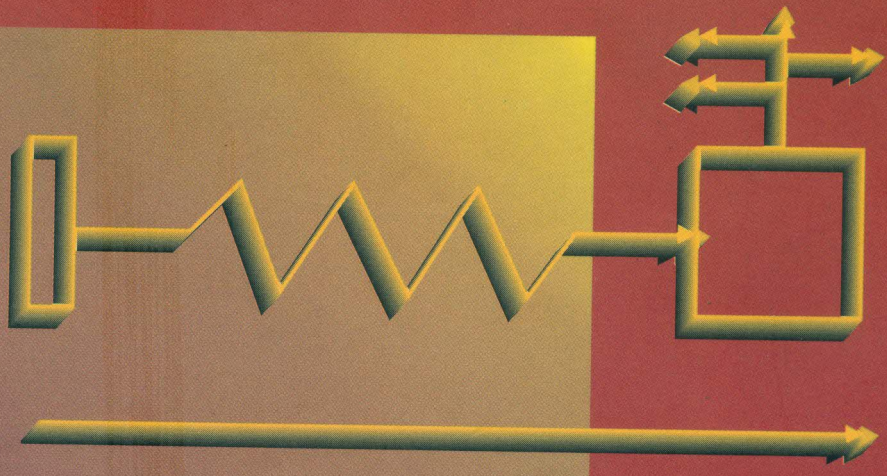
NGUYỄN DOÃN PHƯỚC
PHAN XUÂN MINH

NHẬN DẠNG HỆ THỐNG ĐIỀU KHIỂN



NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT

Nguyễn Doãn Phước & Phan Xuân Minh

NHẬN DẠNG HỆ THỐNG ĐIỀU KHIỂN

NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT
– 2001 –

NHẬN DẠNG HỆ THỐNG ĐIỀU KHIỂN

Nguyễn Doãn Phước & Phan Xuân Minh

Chịu trách nhiệm xuất bản: **PGS.TS. To Đăng Hai**

Biên tập: **Nguyễn Thị Ngọc Khuê**

Trình bày và chế bản: **Tác giả**

Vẽ bìa: **Hương Lan**

NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT

70 Trần Hưng Đạo, Hà Nội

In 1.500 cuốn, khuôn khổ 16x24cm. Tại Xưởng in NXB Văn hóa Dân tộc
Giấy phép xuất bản số: 123-10-6/4/2001
In xong và nộp lưu chiểu tháng 5 năm 2001.

Lời nói đầu

Nhận dạng hệ thống là một trong những công việc đầu tiên phải thực hiện khi giải quyết một bài toán Điều khiển Tự động. Lý do đơn giản chỉ là vì không thể phân tích, tổng hợp hệ thống khi không có mô hình toán học mô tả hệ thống. Trong quá trình xây dựng mô hình hệ thống trên phương diện lý thuyết người ta thường không thể khảo sát được mọi ảnh hưởng của môi trường đến tính động học của hệ thống cũng như những tác động qua lại bên trong hệ thống một cách chính xác tuyệt đối. Rất nhiều yếu tố đã bị bỏ qua hoặc chỉ được xem xét đến như một tác động ngẫu nhiên. Bởi vậy, nếu nói một cách chặt chẽ thì những hiểu biết lý thuyết ban đầu về hệ thống mới chỉ có thể giúp người ta khoanh được vùng lớp các mô hình thích hợp. Để có thể có được một mô hình cụ thể có chất lượng phù hợp với bài toán điều khiển đặt ra trong lớp các mô hình thích hợp đó thì phải sử dụng phương pháp nhận dạng.

Thời điểm ra đời của chuyên ngành Nhận dạng có thể được xem là vào khoảng cuối thập niên 50. Tuy ra đời muộn nhưng Nhận dạng đã phát triển rất nhanh và đã có những thành tựu vượt bậc. Nguyên nhân của sự phát triển vượt bậc đó một phần từ yêu cầu thực tế, song có lẽ phần chính là nhờ có những hỗ trợ tích cực của các ngành khoa học liên quan, đặc biệt là Xử lý tín hiệu và Tin học.

Sự phát triển của Nhận dạng trong lĩnh vực Điều khiển tự động từ năm 1960 đến nay có thể chia ra làm ba giai đoạn phát triển như sau:

- *Giai đoạn một khoảng từ năm 1960 đến 1975 được đánh dấu bằng nhận dạng các mô hình không tham số cho đối tượng điều khiển tuyến tính mà trọng tâm chủ yếu là thiết lập hàm trọng lượng hay hàm đặc tính tần biên–pha dưới dạng một dãy giá trị (phức). Kiến thức lý thuyết cần thiết cho giai đoạn này phần lớn được xây dựng trên cơ sở lý thuyết hàm phức và phân tích phổ tín hiệu.*
- *Giai đoạn hai được đặc trưng bởi sự ra đời của lớp mô hình động liên tục hoặc rời rạc có tham số và được gọi là giai đoạn của nhận dạng tham số mô hình. Thông tin lý thuyết ban đầu về hệ thống ở đây chỉ vừa đủ để người ta có thể lựa chọn được bậc (hay cấu trúc) cho mô hình liên tục hoặc rời rạc. Nhiệm vụ của nhận dạng trong giai đoạn này là xác định giá trị các tham số của mô hình đó với hướng nghiên cứu tập trung là xét tính hội tụ của các phương pháp và ảnh hưởng của nhiễu vào kết quả.*
- *Giai đoạn ba khoảng từ năm 1990 trở lại đây được đánh dấu bằng nhận dạng mô hình động học liên tục phi tuyến và nhận dạng mô hình tham số cho hệ nhiều chiều, trong đó hướng nghiên cứu chính là xét tính nhận dạng được của hệ nhiều chiều. Dần dần, cũng trong giai đoạn này người ta chuyển hướng đi vào nhận dạng các hệ thống suy biến (singular systems).*

Trong vô vàn các phương pháp nhận dạng hệ thống hiện được dùng rộng rãi, chúng tôi chỉ có thể chọn lọc ra và giới thiệu một vài phương pháp đặc trưng làm đại diện. Phương hướng chọn lựa là đi từ mô hình không tham số với công cụ phân tích phổ tín hiệu (chương 2) để làm nền cho công việc nhận dạng tham số mô hình liên tục tuyến tính và mô hình rời rạc tuyến tính sau này (chương 3 và chương 4). Như vậy cuốn sách có nội dung chủ yếu là giới thiệu các phương pháp nhận dạng được hình thành trong giai đoạn 1 và 2. Một phần lý do là những phương pháp này đã trở thành chuẩn mực và đã được cài đặt trong những chương trình tiện dụng của MATLAB giúp bạn đọc có thể sử dụng chúng để kiểm nghiệm lại những điều đã đọc được. Phần nữa là những phương pháp của giai đoạn 3 cho đến nay vẫn chưa có được nhiều sức thuyết phục trong ứng dụng như mong muốn.

Nhằm giúp bạn đọc tiện theo dõi các thuật toán được trình bày, trong cuốn sách này chúng tôi còn giới thiệu một số chương trình viết trên ngôn ngữ lập trình C thể hiện thuật toán để tham khảo.

Cuốn sách được viết với mục đích cung cấp thêm một tài liệu hỗ trợ việc tự học cho sinh viên ngành Điều khiển Tự động đang học môn Lý thuyết Điều khiển nâng cao, sinh viên ngành Điện, cũng như các ngành khác có liên quan tới việc xây dựng mô hình hệ thống. Ngoài ra, cuốn sách còn có mục đích xa hơn là giới thiệu được với những người đang công tác trong lĩnh vực phân tích và tổng hợp hệ thống kỹ thuật một tài liệu tra cứu, tham khảo trong công việc xây dựng mô hình hệ thống.

Để có thể thực hiện được các mục đích đó ngày càng tốt, các tác giả rất mong nhận được những góp ý sửa đổi hay bổ sung thêm từ phía bạn đọc. Thư góp ý xin gửi về:

Trường Đại học Bách khoa Hà Nội
Khoa Điện, Bộ môn Điều khiển Tự động.
Số 1 Đại Cồ Việt. C9/305–306

Hà Nội, ngày 24.4.2001

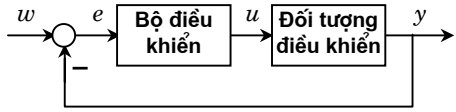
Các tác giả

Mục lục

1 NHẬP MÔN

1.1 Tại sao phải nhận dạng

Xét một bài toán điều khiển theo nguyên tắc phản hồi đầu ra như ở hình 1.1. Muốn tổng hợp được bộ điều khiển cho đối tượng để hệ kín có được chất lượng như mong muốn thì trước tiên cần phải hiểu biết về đối tượng, tức là cần phải có một mô hình toán học mô tả đối tượng. Không thể điều khiển đối tượng khi không hiểu biết hoặc hiểu sai lệch về nó. Kết quả tổng hợp bộ điều khiển phụ thuộc rất nhiều vào mô hình mô tả đối tượng. Mô hình càng chính xác, hiệu suất công việc càng cao.



Hình 1.1: Điều khiển theo nguyên tắc phản hồi đầu ra.

Việc xây dựng mô hình cho đối tượng được gọi là mô hình hóa. Người ta thường phân chia các phương pháp mô hình hóa ra làm hai loại:

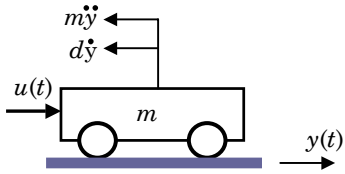
- phương pháp lý thuyết và
- phương pháp thực nghiệm.

Phương pháp lý thuyết là phương pháp thiết lập mô hình dựa trên các định luật có sẵn về quan hệ vật lý bên trong và quan hệ giao tiếp với môi trường bên ngoài của đối tượng. Các quan hệ này được mô tả theo quy luật lý–hóa, quy luật cân bằng, ... dưới dạng những phương trình toán học.

Trong các trường hợp mà ở đó sự hiểu biết về những quy luật giao tiếp bên trong đối tượng cũng về mối quan hệ giữa đối tượng với môi trường bên ngoài không được đầy đủ để có thể xây dựng được một mô hình hoàn chỉnh, nhưng ít nhất từ đó có thể cho biết các thông tin ban đầu về dạng mô hình thì tiếp theo người ta phải áp dụng phương pháp thực nghiệm để hoàn thiện nốt việc xây dựng mô hình đối tượng trên cơ sở *quan sát tín hiệu vào* $u(t)$ *và ra* $y(t)$ của đối tượng sao cho mô hình thu được bằng phương pháp thực nghiệm thỏa mãn các yêu cầu của phương pháp lý thuyết đề ra. Phương pháp thực nghiệm đó được gọi là *nhận dạng* hệ thống điều khiển.

Như vậy, khái niệm nhận dạng hệ thống điều khiển được hiểu là sự bổ sung cho việc mô hình hóa đối tượng mà ở đó lượng *thông tin ban đầu về đối tượng điều khiển không đầy đủ*. Các thông tin ban đầu này có tên gọi chung là thông tin *A–priori*.

Ví dụ 1: Chẳng hạn ta phải xây dựng mô hình cho đối tượng là một chiếc xe chuyển hàng. Tín hiệu đầu vào tác động để đẩy xe là lực $u(t)$. Dưới tác động của lực $u(t)$ xe sẽ đi được quãng đường ký hiệu bởi $y(t)$.



Hình 1.2: Xây dựng mô hình cho đối tượng là một chiếc xe chuyển hàng.

Khi chuyển động sẽ có hai lực cản trở sự chuyển động của xe (bỏ qua ma sát tĩnh). Thứ nhất là lực ma sát động xác định bởi:

$$F_s = d\frac{dy}{dt}, \quad d \text{ là hệ số ma sát động}$$

và thứ hai là lực cản trở sự thay đổi tốc độ

$$F_{gt} = m\frac{d^2y}{dt^2}, m \text{ là khối lượng của xe.}$$

Theo nguyên lý cân bằng lực ta có được mô hình mô tả đối tượng, tức là mô tả quan hệ giữa tín hiệu vào $u(t)$ và tín hiệu ra $y(t)$ như sau:

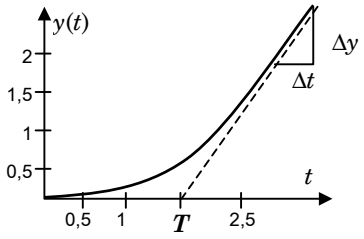
$$m\frac{d^2y}{dt^2} + d\frac{dy}{dt} = u \quad \Rightarrow \quad G(s) = \frac{k}{s(1+Ts)} \tag{1.1a}$$

trong đó $k = \frac{1}{d}$ và $T = \frac{m}{d}$.

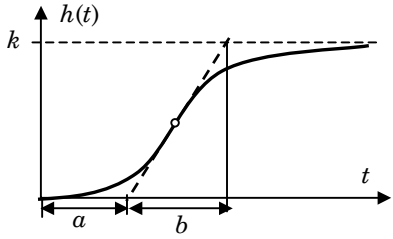
Mô hình (1.1a) được xây dựng từ các hiểu biết ban đầu về đối tượng, nhưng chưa phải là mô mình cụ thể cho chiếc xe chở hàng mà ta đang xét vì các tham số về hệ số ma sát d cũng như khối lượng xe m là chưa có. Nói cách khác mô hình mà ta cần chỉ là một trong các mô hình có dạng (1.1a). Để có được một mô hình hoàn chỉnh thì ta cần phải xác định nốt những tham số k và T còn lại.

Để làm được điều này, người ta áp dụng phương pháp thực nghiệm bằng cách tác động tạm thời vào xe tại thời điểm t=0 một lực cố định, ví dụ như $u(t)$=1 rồi đo tín hiệu ra là quãng đường đi được $y(t)$. Biểu diễn quãng đường đi được $y(t)$ phụ thuộc theo t dưới dạng đồ thị ta có hình 1.3. Từ đồ thị đó ta tính được T là giao điểm của đường tiệm cận

của $y(t)$ với trục hoành và $k \approx \frac{\Delta y}{\Delta t}$. Câu hỏi tại sao ta lại tính được các tham số như vậy sẽ được trả lời sau trong chương 3.



Hình 1.3: Nhận dạng tham số cho mô hình xe chở hàng.



Hình 1.4: Xác định tham số cho mô hình đối tượng động cơ một chiều.

Ví dụ 2: Ta xét thêm ví dụ với đối tượng là động cơ một chiều. Từ những kiến thức lý thuyết chung về động cơ một chiều (thông tin A–priori) người ta mới chỉ có thể xác định được rằng mô hình xấp xỉ tuyến tính của nó có dạng khâu quán tính bậc hai như sau:

$$G(s) = \frac{k}{(1+T_1 s)(1+T_2 s)}, \tag{1.1b}$$

còn lại chi tiết hơn thì ba tham số k, T_1 và T_2 chưa thể xác định được do còn phụ thuộc vào đặc tính riêng của kết cấu từng động cơ. Nói cách khác, từ thông tin A–priori người ta mới chỉ biết được rằng động cơ một chiều thuộc lớp mô hình quán tính bậc hai (1.1b), trong đó k, T_1, T_2 là những phần tử bất kỳ của $\mathbb{R}$.

Để có thể tìm được một mô hình cụ thể cho đối tượng từ lớp các mô hình dạng (1.1b) người ta phải áp dụng phương pháp thực nghiệm (nhận dạng). Nếu như sự tác động của nhiễu là bỏ qua được, các phép đo là chính xác và công việc nhận dạng có thể được thực hiện bằng cách chủ động kích thích đối tượng với một tín hiệu đầu vào thích hợp chọn trước thì phương pháp thường dùng là xác định hàm quá độ thông qua đo tín hiệu ra khi tín hiệu vào là hàm $1(t)$.

Tiếp theo người ta biểu diễn $h(t)$ dưới dạng đồ thị rồi kẻ đường tiếp tuyến với $h(t)$ tại điểm uốn để có a, b và đường tiệm cận tại $t=\infty$ để có k (hình 1.4). Hai tham số T_1 và T_2 còn lại sẽ được xác định từ a và b. Chi tiết thêm về cách xác định T_1, T_2 từ a, b sẽ được trình bày sau trong chương 3. Ở đây chúng tôi chỉ đề cập sơ lược để minh họa cho sự khác biệt giữa phương pháp xây dựng mô hình theo kiểu lý thuyết và thực nghiệm (nhận dạng).

1.1.1 Định nghĩa

Khái niệm về bài toán nhận dạng vừa nêu trên đã được Zadeh thu gọn vào định nghĩa phát biểu năm 1962 với hai nét cơ bản như sau:

1) Nhận dạng là phương pháp thực nghiệm nhằm xác định một mô hình cụ thể trong lớp các mô hình thích hợp đã cho trên cơ sở quan sát các tín hiệu vào ra.

2) Mô hình tìm được phải có sai số với đối tượng là nhỏ nhất.

Theo định nghĩa này thì những bài toán nhận dạng sẽ được phân biệt với nhau ở ba điểm chính. Đó là:

- Lớp mô hình thích hợp. Chẳng hạn lớp các mô hình tuyến tính không có cấu trúc (không biết bậc của mô hình) hoặc có cấu trúc (ví dụ như lớp mô hình (1.1)), lớp các mô hình lưỡng tuyến tính (bilinear), …
- Loại tín hiệu quan sát được (tiền định/ngẫu nhiên).
- Phương thức mô tả sai lệch giữa mô hình và đối tượng thực.

1.1.2 Lớp mô hình thích hợp

Tập hợp tất cả các mô hình có cùng cấu trúc thỏa mãn các yêu cầu về thông tin A–priori mà phương pháp lý thuyết đã đặt ra được gọi là *lớp các mô hình thích hợp*. Ví dụ như tất cả các mô hình dạng (1.1b) với k, T_1 và T_2 là ba phần tử bất kỳ của $\mathbb{R}$ đều có thể là mô hình của động cơ một chiều.

Trong tài liệu này chúng ta sẽ chỉ quan tâm tới các bài toán nhận dạng với lớp những *mô hình tuyến tính* gần đúng của đối tượng. Một mô hình được gọi là tuyến tính nếu ánh xạ T_M mô tả quan hệ giữa r tín hiệu vào $\underline{u}(t)=\begin{pmatrix} u_1(t) \\ \vdots \\ u_r(t) \end{pmatrix}$ và s tín hiệu ra $\underline{y}(t)=\begin{pmatrix} y_1(t) \\ \vdots \\ y_s(t) \end{pmatrix}$ của mô hình thỏa mãn

$$T_M\left(a_1\underline{u}_1(t)+ a_2\,\underline{u}_2(t)\right) = a_1T_M\left(\underline{u}_1(t)\right)+a_2T_M\left(\underline{u}_2(t)\right), \tag{1.2}$$

trong đó $a_1,\ a_2\in\mathbb{R}$. Tính chất trên của mô hình tuyến tính, trong điều khiển, còn được gọi là *nguyên lý xếp chồng*.

Ví dụ: Mô hình trạng thái cho đối tượng không dừng dạng

$$T_M: \quad \frac{d\underline{x}}{dt} = A(t)\underline{x} + B(t)\underline{u}$$
$$\underline{y} = C(t)\underline{x} + D(t)\underline{u}$$

với n biến trạng thái $\underline{x}(t) = \begin{pmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{pmatrix}$ và $A(t)$, $B(t)$, $C(t)$, $D(t)$ là những ma trận phụ thuộc thời gian t (phần tử của chúng là các hàm theo t), là một mô hình tuyến tính. Thật vậy, nếu với kích thích (đầu vào) $\underline{u}_1(t)$ hệ có đáp ứng (đầu ra) $\underline{y}_1(t)$ và với kích thích $\underline{u}_2(t)$ có đáp ứng $\underline{y}_2(t)$, tức là

$$\begin{cases} \dfrac{d\underline{x}_1}{dt} = A(t)\underline{x}_1 + B(t)\underline{u}_1 \\ \underline{y}_1 = C(t)\underline{x}_1 + D(t)\underline{u}_1 \end{cases}, \tag{1.3a}$$

$$\begin{cases} \dfrac{d\underline{x}_2}{dt} = A(t)\underline{x}_2 + B(t)\underline{u}_2 \\ \underline{y}_2 = C(t)\underline{x}_2 + D(t)\underline{u}_2 \end{cases} \tag{1.3b}$$

thì với tín hiệu đầu vào

$$\underline{u}(t) = a_1\underline{u}_1(t) + a_2\underline{u}_2(t), \qquad a_1, a_2 \in \mathbb{R}$$

đầu ra sẽ là

$$\underline{y}(t) = a_1\,\underline{y}_1(t) + a_2\,\underline{y}_2(t),$$

vì từ (1.3) có

$$\begin{cases} a_1\dfrac{d\underline{x}_1}{dt} = a_1A(t)\underline{x}_1 + a_1B(t)\underline{u}_1a_2 \\ a_2\dfrac{d\underline{x}_2}{dt} = a_2A(t)\underline{x}_2 + a_2D(t)\underline{u}_2 \end{cases}$$

$$\Rightarrow \quad \underbrace{a_1\frac{d\underline{x}_1}{dt} + a_2\frac{d\underline{x}_2}{dt}}_{\dfrac{d\underline{x}}{dt}} = A(t)\underbrace{\left[a_1\underline{x}_1 + a_2\underline{x}_2\right]}_{\underline{x}} + B(t)\underbrace{\left[a_1\underline{u}_1 + a_2\underline{u}_2\right]}_{\underline{u}}$$

$$\Rightarrow \quad \underline{y} = C(t)\underline{x} + B(t)\underline{u} = C(t)\cdot\left[a_1\underline{x}_1 + a_2\underline{x}_2\right] + B(t)\cdot\left[a_1\underline{u}_1 + a_2\underline{u}_2\right]$$

$$= a_1\underbrace{\left[C(t)x_1 + B(t)u_1\right]}_{\underline{y}_1} + a_2\underbrace{\left[C(t)x_2 + B(t)u_2\right]}_{\underline{y}_2}. \qquad \square$$

Cũng cần phải nhấn mạnh rằng ba lý do chính cho việc mô hình tuyến tính thường được sử dụng là:

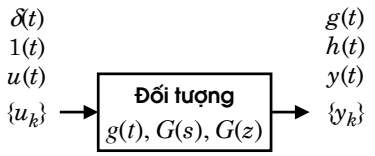
1) Mô hình càng đơn giản, càng tốn ít chi phí. Các tham số mô hình tuyến tính dễ dàng xác định được nhờ nhận dạng mà không cần phải đi từ những phương trình hóa lý phức tạp mô tả đối tượng.

2) Tập các phương pháp nhận dạng tuyến tính rất phong phú và không phải tốn nhiều thời gian để thực hiện.

3) Cấu trúc đơn giản của mô hình cho phép dễ dàng theo dõi được kết quả điều khiển đối tượng và chỉnh định lại mô hình cho phù hợp. Tính chất này đặc biệt rất cần thiết để thực hiện các bài toán điều khiển thích nghi.

Sau đây là các loại mô hình tuyến tính được sử dụng nhiều nhất khi nhận dạng đối tượng SISO không có nhiễu tác động (đối tượng chỉ có một tín hiệu vào $u(t)$ và một tín hiệu ra $y(t)$–*single input*, *single output*):

I.1. Dãy giá trị $\{g_k\}$ của hàm trọng lượng $g(t)$ với $g_k=g(kT_a)$, hoặc $\{h_k\}$ của hàm quá độ $h(t)$ với $h_k=h(kT_a)$, trong đó T_a là chu kỳ trích mẫu tín hiệu. Nhận dạng có nhiệm vụ thông qua việc quan sát (hoặc đo) các tín hiệu vào ra để xác định được $\{g_k\}$ hoặc $\{h_k\}$. Do đặc thù như vậy, dạng bài toán nhận dạng này được xếp vào lớp *bài toán nhận dạng mô hình không tham số* (nonparametric identification).



Hình 1.5: Mô hình đối tượng SISO không có nhiễu tác động.

I.2. Hàm truyền đạt $G(s)$, được hiểu là tỷ số giữa ảnh Laplace của đáp ứng với ảnh Laplace của kích thích và nó chính là ảnh Laplace của hàm trọng lượng $g(t)$:

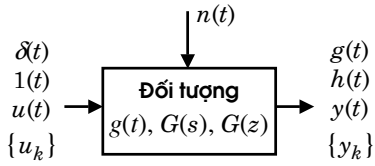
$$G(s)=\frac{Y(s)}{U(s)}=e^{-s\tau}K\frac{1+b_1s+\ \cdots\ +b_{n_b}s^{n_b}}{1+a_1s+\ \cdots\ +a_{n_a}s^{n_a}}, \tag{1.4}$$

trong đó $n_b \leq n_a$ (n_b, n_a gọi là bậc mô hình) là điều kiện để đối tượng có khả năng tồn tại (theo nghĩa causal) và có thể đã biết trước, τ là ký hiệu chỉ thời gian trễ của đối tượng. Nhiệm vụ của nhận dạng là thông qua việc quan sát những tín hiệu vào ra (hoặc qua việc đo dãy giá trị $\{u_k\},\{y_k\}$) để xác định các tham số τ, K, b_1, b_{12}, ... , b_{n_b}, a_1, a_2, ... , a_{n_a} cũng như bậc n_b, n_a (nếu n_b, n_a chưa cho trước) của mô hình. Các dạng bài toán này có tên gọi *nhận dạng mô hình có tham số* (parametric identification).

I.3. Hàm truyền đạt $G(z)$, được hiểu là tỷ số giữa ảnh $\mathcal{Z}$ của dãy giá trị đáp ứng $\{y_k\}$, $y_k=y(kT_a)$, với ảnh $\mathcal{Z}$ của dãy giá trị kích thích $\{u_k\}$, $u_k=u(kT_a)$,

$$G(z)=\frac{Y(z)}{U(z)}=z^{-l}K\frac{1+b_1z^{-1}+\ \cdots\ +b_{n_b}z^{-n_b}}{1+a_1z^{-1}+\ \cdots\ +a_{n_a}z^{-n_a}}, \tag{1.5}$$

trong đó $z = e^{sT_a}$ và T_a là chu kỳ trích mẫu tín hiệu. Khi $l=0$, mô hình (1.5) trên được gọi là mô hình ARMA. Nhận dạng có nhiệm vụ thông qua việc quan sát những tín hiệu vào ra để xác định tham số của mô hình. Bởi vậy bài toán này cũng thuộc lớp *bài toán nhận dạng mô hình có tham số*.



Hình 1.6: Mô hình đối tượng SISO có nhiễu tác động.

Trường hợp đối tượng nhận dạng bị tác động bởi nhiễu thì thông thường có hai biện pháp để giải quyết:

1) *Loại bỏ ảnh hưởng nhiễu* $n(t)$ thông qua cực tiểu hóa phiếm hàm đánh giá sai lệch giữa mô hình và đối tượng.

2) *Mô hình hóa tín hiệu nhiễu*. Mặc dù nhiễu $n(t)$ là tín hiệu không xác định được một cách tổng quát, song phần lớn các nhiễu tồn tại trong tự nhiên lại thuộc lớp hàm có ảnh $\mathcal{Z}$ mô tả được dưới dạng:

$$N(z) = H(z)W(z),$$

trong đó $W(z)$ là ảnh $\mathcal{Z}$ của tín hiệu ồn trắng (white noise) và $H(z)$ là mô hình của nhiễu.

Kết hợp với (1.5) cho các trường hợp $H(z)$ khác nhau ta có:

II.1. Mô hình ARX:

$$G(z) = z^{-l}K\underbrace{\frac{1+b_1z^{-1}+\ \cdots\ +b_{n_b}z^{-n_b}}{1+a_1z^{-1}+\ \cdots\ +a_{n_a}z^{-n_a}}}_{A(z)}, \qquad H(z)=\frac{1}{A(z)}. \tag{1.6}$$

II.2. Mô hình ARMAX:

$$G(z) = z^{-l}K\underbrace{\frac{1+b_1z^{-1}+\ \cdots\ +b_{n_b}z^{-n_b}}{1+a_1z^{-1}+\ \cdots\ +a_{n_a}z^{-n_a}}}_{A(z)}, \qquad H(z)=\frac{C(z)}{A(z)}. \tag{1.7}$$

trong đó

$$C(z) = 1 + c_1z^{-1} + \cdots + c_{n_c}z^{-n_c}$$

II.3. Mô hình Box–Jenkin:

$$G(z) = z^{-l}K\frac{1+b_1z^{-1}+\ \cdots\ +b_{n_b}z^{-n_b}}{1+a_1z^{-1}+\ \cdots\ +a_{n_a}z^{-n_a}}, \qquad H(z)=\frac{C(z)}{F(z)}. \tag{1.8}$$

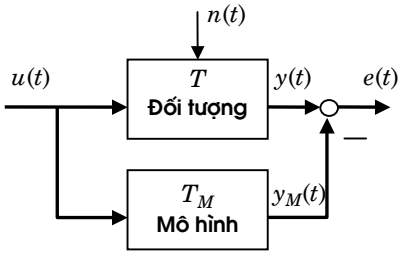
trong đó

$$C(z) = 1 + c_1 z^{-1} + \cdots + c_{n_c} z^{-n_c} \quad \text{và} \quad F(z) = 1 + f_1 z^{-1} + \cdots + f_{n_f} z^{-n_f}$$

1.1.3 Mô tả sai lệch giữa mô hình và đối tượng thực

Trong một bài toán nhận dạng, sai lệch giữa đối tượng thực T và mô hình T_M thường được biểu diễn qua:

1) *Sai lệch đầu ra*. Đây là cách biểu diễn dễ chấp nhận nhất, trực quan, song bị hạn chế do tính phức tạp của mô hình sai lệch và sự phi tuyến giữa các tham số cần nhận dạng với đại lượng sai lệch $e(t)$. Mô hình sai lệch đầu ra thường được sử dụng cho các bài toán nhận dạng có mô hình tĩnh, bài toán xác định điểm lấy mẫu của chuỗi Voltera hay bài toán quan sát điểm trạng thái, ...



Hình 1.7: Sai lệch đầu ra.

Bài toán nhận dạng bây giờ được phát biểu cụ thể hơn là thông qua việc quan sát các tín hiệu vào ra, hãy xác định mô hình T_M sao cho:

a) Bình phương năng lượng của sai lệch nhỏ nhất:

$$Q = \int_{-\infty}^{\infty} [y(t) - y_M(t)]^2 dt \to \min!, \tag{1.9a}$$

b) Giá trị trung bình của bình phương năng lượng sai lệch nhỏ nhất:

$$Q = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} [y(t) - y_M(t)]^2 dt \to \min!, \tag{1.9b}$$

Nếu việc quan sát tín hiệu được thực hiện bằng cách đo rời rạc dãy giá trị các tín hiệu vào/ra thì hai công thức trên được cải biên một cách phù hợp thành

a) $$Q = \sum_{k=-\infty}^{\infty} [y(kT_a) - y_M(kT_a)]^2 \to \min!, \tag{1.9c}$$

b) $$Q = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{k=-N}^{N} [y(kT_a) - y_M(kT_a)]^2 \to \min!, \tag{1.9d}$$

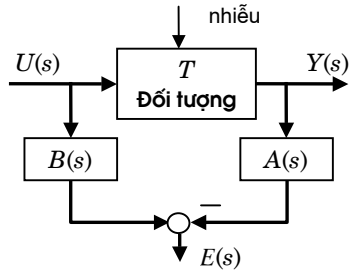
trong đó T_a là chu kỳ trích mẫu tín hiệu.

2) *Sai lệch tổng quát* $e(t)$. Đây là loại sai lệch rất được ưa dùng trong các bài toán nhận dạng tham số với mô hình tuyến tính động vì loại sai lệch này biểu diễn được quan hệ tuyến tính giữa các tham số cần xác định với những giá trị đo được $\{y_k\}$, $\{u_k\}$

như hình 1.8 mô tả, trong đó $A(s)$, $B(s)$ là hai đa thức của mô hình tham số kiểu (1.4)

$$G(s) = \frac{B(s,\underline{b})}{A(s,\underline{a})} = \frac{b_0 + b_1 s + \cdots + b_{n_b} s^{n_b}}{a_0 + a_1 s + \cdots + a_{n_a} s^{n_a}},$$

với $\underline{a} = \begin{pmatrix} a_0 \\ \vdots \\ a_{n_a} \end{pmatrix}$, $\underline{b} = \begin{pmatrix} b_0 \\ \vdots \\ b_{n_b} \end{pmatrix}$. Sai lệch $e(t)$ khi đó sẽ được biểu diễn thông qua ảnh Laplace của nó là $E(s)$ thành



Hình 1.8: Sai lệch tổng quát.

$$E(s) = U(s)B(s,\underline{b}) - Y(s)A(s,\underline{a}).$$

Trong nhiều tài liệu, sai lệch $e(t)$ còn được gọi là *sai lệch dự báo tuyến tính*.

Bài toán đặt ra là qua việc quan sát các tín hiệu vào ra, xác định những vector tham số $\underline{a}$, $\underline{b}$ sao cho

a) Bình phương năng lượng của sai lệch là nhỏ nhất:

$$Q = \int_{-\infty}^{\infty} |e(t)|^2 dt \to \min!, \tag{1.10a}$$

và nếu áp dụng công thức Paserval

$$\int_{-\infty}^{\infty} |e(t)|^2 dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |E(j\omega)|^2 d\omega$$

thì (1.9a) còn được tính trực tiếp trong miền phức bằng

$$Q = \frac{1}{2\pi} \int_{-\infty}^{\infty} |E(j\omega)|^2 d\omega \to \min!. \tag{1.10b}$$

trong đó $E(j\omega)$ là ảnh Fourier của $e(t)$.

b) Giá trị trung bình của bình phương năng lượng sai lệch là nhỏ nhất:

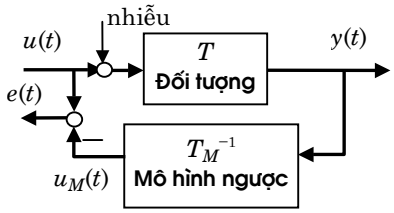
$$Q = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} |e(t)|^2 dt = \lim_{T\to\infty} \frac{1}{4\pi T} \int_{-T}^{T} |E(j\omega)|^2 dt \to \min!, \tag{1.10c}$$

Cũng tương tự như ở trường hợp 1), khi việc quan sát tín hiệu được thực hiện bằng cách đo rời rạc dãy giá trị các tín hiệu vào/ra thì những công thức trên sẽ được sửa đổi thành

a) $$Q = \sum_{k=-\infty}^{\infty} [e(kT_a)]^2 \to \min!, \tag{1.10d}$$

b) $$Q = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{k=-N}^{N} [e(kT_a)]^2 \to \min!, \tag{1.10e}$$

3) *Sai lệch đầu vào*. Là loại sai lệch thường được dùng cho lớp các bài toán nhận dạng không có nhiễu đầu ra. Loại sai lệch đầu vào, do phải xác định mô hình ngược T_M^{-1} thay vì T_M nên có những hạn chế của nó và cho tới giữa thập niên 90 ít được sử dụng trong thực tế. Khoảng từ năm 1992 trở lại đây, với sự ra đời của kỹ thuật đại số điều khiển vi phân, sự hạn chế này đã dần có phần được cải thiện.



Hình 1.9: Sai lệch đầu vào.

1.2 Phân lớp các bài toán nhận dạng

Theo định nghĩa của Zadeh về nhận dạng thì có ba tiêu chuẩn phân loại một bài toán nhận dạng như sau:

- phân theo loại các tín hiệu đã quan sát được,
- phân theo lớp các mô hình thích hợp,
- phân theo dạng sai số giữa đối tượng thực và mô hình.

Thêm vào đó khi tiến hành nhận dạng một đối tượng còn cần phải chú ý tới các điều kiện khách quan do yêu cầu kỹ thuật như:

- thời gian quan sát tín hiệu không thể lớn tùy ý,
- tín hiệu quan sát được thường bị chặn.

Để cụ thể hoá những khái niệm trên của Zadeh, hãy xét một ví dụ. Chẳng hạn có một đối tượng T cần được nhận dạng. Đối tượng T được giả thiết, hoặc từ phương pháp lý thuyết xác định được (thông tin A–priori) là SISO (single input single output / một vào một ra), tham số hằng và ổn định. Nhiệm vụ của nhận dạng là trong lớp các mô hình thích hợp $\mathcal{M}_1$ (lớp các mô hình động học có tham số hằng và ổn định), chỉ nhờ vào quan sát các tín hiệu vào ra $u(t)$ và $y(t)$, xác định một mô hình $T_M \in \mathcal{M}_1$ cho đối tượng sao cho sai số giữa mô hình T_M và đối tượng thật T, được ký hiệu bởi $S(T,T_M)$, là nhỏ nhất.

Ta có bài toán nhận dạng thứ nhất như sau:

1) Qua quan sát tín hiệu vào ra $u(t)$ và $y(t)$, tìm $T_M \in \mathcal{M}_1$ để có $S(T,T_M) \rightarrow \min!$.

Nếu như ngoài các tín hiệu vào ra, tác động tới đối tượng còn có nhiễu $n(t)$ làm cho tín hiệu thu được đầu ra $y(t)$ có sai lệch so với tín hiệu thật $y_0(t)$ thì bài toán nhận dạng này còn có thêm nhiệm vụ không đơn giản chút nào là tách sự ảnh hưởng của nhiễu $n(t)$ vào $y_0(t)$. Ta có bài toán nhận dạng thứ hai:

2) Qua quan sát tín hiệu vào ra $u(t)$, $y(t)$ để lọc ra $y_0(t)$ hãy tìm $T_M \in \mathcal{M}_1$ theo $u(t)$ và $y_0(t)$ sao cho $S(T,T_M) \to \min!$.

Thông thường, ở những bài toán nhận dạng có nhiễu như bài toán 2, mà ở đó $y_0(t)$ không tách được ra khỏi $y(t)$ thì bắt buộc phải xác định $T_M \in \mathcal{M}_1$ phụ thuộc vào $u(t)$, $y(t)$ và sau đó mới đánh giá sự ảnh hưởng của nhiễu $n(t)$ vào kết quả.

Với giả thiết thêm rằng từ thông tin A–priori của phương pháp lý thuyết người ta còn được biết thêm là đối tượng *tuyến tính*, thì lớp các mô hình thích hợp bây giờ là tập con $\mathcal{M}_2 \subset \mathcal{M}_1$ chỉ gồm các *mô hình động học tuyến tính* có tham số hằng và ổn định. Bài toán nhận dạng ban đầu được đơn giản thành:

3) Qua quan sát tín hiệu vào ra $u(t)$, $y(t)$ để lọc ra $y_0(t)$, xác định $T_M \in \mathcal{M}_2$ theo $u(t)$ và $y_0(t)$ sao cho $S(T,T_M) \to \min!$.

Tiếp tục, nếu như sai số $S(T,T_M)$ được cho cụ thể là sai lệch đầu ra với phương trình biểu diễn (1.8) thì sẽ có được bài toán số 4 như sau:

4) Qua quan sát tín hiệu vào ra $u(t)$, $y(t)$ để lọc ra $y_0(t)$, hãy tìm $T_M \in \mathcal{M}_2$ theo $u(t)$ và $y_0(t)$ sao cho $Q = \int_0^\infty [y_0(t) - y_M(t)]^2 dt \to \min!$.

Giả thiết thêm rằng từ thông tin A–priori có được mô hình thích hợp là mô hình tham số hằng, chẳng hạn như T_M có đặc tính tần là hàm hữu tỷ phức với vector tham số $\underline{a}$, $\underline{b}$ thì lớp các mô hình thích hợp bây giờ sẽ là tập con $\mathcal{M}_3 \subset \mathcal{M}_2$ chỉ gồm các hàm hữu tỷ phức $G(j\omega, \underline{a}, \underline{b})$.

Nếu ký hiệu $Y_0(j\omega)$ cho ảnh Fourier của $y_0(t)$, $U(j\omega)$ là ảnh của $u(t)$ thì bài toán 4 trở thành bài toán nhận dạng mô hình tham số được phát biểu như sau:

5) Qua quan sát tín hiệu vào ra $u(t)$, $y(t)$ để lọc ra $y_0(t)$, xác định vector tham số $\underline{a}$, $\underline{b}$ để có $\frac{1}{2\pi} \int_{-\infty}^{\infty} |E(j\omega)|^2 d\omega \to \min!$.

Như vậy, qua ví dụ với năm bài toán trên có thể nhận thấy, từ một vấn đề xây dựng mô hình động học cho đối tượng T, với những thông tin A–priori khác nhau là những bài toán nhận dạng khác nhau.

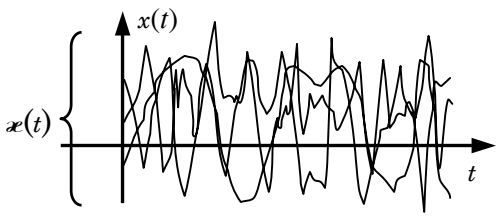
Trong cả năm bài toán được nêu trên, khi nhận dạng, ta đều phải đo cả tín hiệu vào và tín hiệu ra. Bởi vậy những bài toán đó rất phù hợp với các điều kiện *nhận dạng bị động* (*passive*), hay còn gọi *nhận dạng trực tuyến* (on–line) của điều khiển thích nghi mà ở đó đối tượng nhận dạng không thể tách riêng ra khỏi hệ thống cũng như quá trình nhận dạng phải được thực hiện song song cùng với quá trình làm việc của toàn bộ hệ thống.

Nếu như điều kiện cho phép tách đối tượng ra khỏi hệ thống khi nhận dạng thì để tránh việc phải đo tín hiệu vào (và do đó bớt đi một sai số đo) ta có thể chủ động kích thích đối tượng bằng một tín hiệu vào thích hợp và chỉ phải đo tín hiệu ra. Những dạng bài toán nhận dạng như vậy được gọi là kiểu *nhận dạng chủ động* (*active*) hay *nhận dạng không trực tuyến* (off–line). Một trong những tín hiệu đầu vào thường hay được sử dụng khi nhận dạng chủ động là tín hiệu ồn trắng, tức là loại tín hiệu có mật độ phổ là hằng số ở mọi giá trị tần số.

1.3 Quá trình ngẫu nhiên

1.3.1 Khái niệm

Khi đo tín hiệu vào/ra, trạng thái để nhận dạng đối tượng hay hệ thống, kết quả nhận dạng sẽ phụ thuộc rất nhiều vào tính chính xác của các phép đo này. Khác với loại tín hiệu tiền định là với những điều kiện đo như nhau các phép đo sẽ cho ra cùng một kết quả thì khi đo tín hiệu ngẫu nhiên, mặc dù các phép đo đều được thực hiện trong cùng một điều kiện, các kết quả đo sẽ rất khác nhau. Ví dụ để đo được tín hiệu $x(t)$ người ta có thể nhận được rất nhiều (thậm chí không đếm được) các hàm thời gian khác nhau. Điều này gây không ít khó khăn cho việc mô tả và xử lý chúng.



Hình 1.10: Quá trình ngẫu nhiên là một tập hợp các hàm ngẫu nhiên cùng tính chất..

Tuy nhiên, nếu biết được thêm rằng các hàm thời gian nhận được này có cùng một tính chất E nào đó đặc trưng cho tín hiệu $x(t)$ thì việc mô tả tín hiệu $x(t)$ có thể được thay bằng việc mô tả tập hợp $\mathscr{x}(t)$ của tất cả các hàm thời gian có cùng tính chất E trên. Tập $\mathscr{x}(t)$ được gọi là một *quá trình ngẫu nhiên*, trong đó tín hiệu $x(t)$ nhận được chỉ là một phần tử (hình 1.10).

1.3.2 Các tham số của quá trình ngẫu nhiên

Một quá trình ngẫu nhiên $\mathscr{x}(t)$ được mô tả một cách đầy đủ bởi *các hàm phân bố*.

1) Hàm phân bố bậc một

$$F(x, t) = P\left(\mathscr{x}(t) \le x\right) \tag{1.11}$$

xác định xác suất xuất hiện hàm thời gian mà tại thời điểm t có giá trị không lớn hơn giá trị x cho trước.

2) Hàm phân bố bậc cao

$$F(x_1, x_2, \dots, x_n, t_1, t_2, \dots, t_n) = P\left(\mathcal{x}(t_1) \le x_1, \mathcal{x}(t_2) \le x_2, \dots, \mathcal{x}(t_n) \le x_n\right)$$

xác định xác suất xuất hiện hàm thời gian mà tại thời điểm t_k có giá trị không lớn hơn giá trị x_k, cho trước $k = 1, 2, \dots, n$.

Đạo hàm của các hàm phân bố

$$f(x, t) = \frac{\partial F(x,t)}{\partial t} \tag{1.12a}$$

$$f(x_1, x_2, \dots, x_n, t_1, t_2, \dots, t_n) = \frac{\partial^n F}{\partial x_1 \cdots \partial x_n} \tag{1.12b}$$

được gọi là *mật độ phân bố*. Đối với $f(x, t)$ thì từ một giá trị $\Delta x>0$ cho trước, tích $f(x,t)\Delta x$ sẽ cho biết xác suất xuất hiện hàm thời gian nhận được trong khi đo tín hiệu mà tại thời điểm t có giá trị nằm trong khoảng $[x, x+\Delta x]$.

Cho hai quá trình ngẫu nhiên $\mathcal{x}(t)$ và $\mathcal{y}(t)$. Cũng tương tự như với một quá trình, hàm phân bố cho hai quá trình ngẫu nhiên

$$F(x, y, t_1, t_2) = P\left(\mathcal{x}(t_1) \le x, \mathcal{y}(t_2) \le y\right)$$

được hiểu là xác suất xuất hiện hàm thời gian của $\mathcal{x}(t)$ mà tại thời điểm t_1 có giá trị không lớn hơn giá trị x và của $\mathcal{y}(t)$ mà tại thời điểm t_2 có giá trị không lớn hơn giá trị y.

Mặc dù các hàm phân bố đã có thể mô tả được đầy đủ tập $\mathcal{x}(t)$, song nó vẫn còn quá phức tạp. Bởi vậy, thay vì phải xác định cụ thể các hàm phân bố người ta thường hay xác định các tham số ngẫu nhiên đặc trưng của nó. Với một lớp các hàm phân bố đặc biệt (ví dụ hàm Gauss) hoàn toàn có thể từ các tham số này xác định được chính xác các hàm phân bố.

Những tham số ngẫu nhiên của những hàm phân bố bao gồm:

1) Giá trị trung bình:

$$m_x(t) = M[\mathcal{x}(t)] = \int_{-\infty}^{\infty} x \cdot f(x,t)dx \tag{1.13}$$

2) Hàm tự tương quan:

$$r_x(t_1, t_2) = M\left[\mathcal{x}(t_1)\mathcal{x}(t_2)\right] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left[x_1 x_2 f(x_1, x_2, t_1, t_2)\right]dx_1 dx_2 . \tag{1.14}$$

Hàm tự tương quan chính là giá trị trung bình của mối tương quan giữa $\mathcal{x}(t)$ tại thời điểm t_1 với $\mathcal{x}(t)$ tại thời điểm t_2.

3) Hàm phương sai:

$$c_x(t_1, t_2) \quad = M\left[\left(\mathcal{x}(t_1) - m_x(t_1)\right)\left(\mathcal{x}(t_2) - m_x(t_2)\right)\right] \tag{1.15}$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left[(x_1 - m_x(t_1))\cdot(x_1 - m_x(t_1))\cdot f(x_1, x_2, t_1, t_2)\right]dx_1 dx_2$$

4) Giá trị tản mát: $\sigma_x^2(t) = r_x(t, t)$. (1.16)

5) Hàm hỗ tương quan:

$$r_{xy}(t_1, t_2) = M\left[\boldsymbol{x}(t_1)\boldsymbol{y}(t_2)\right] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left[xy\cdot f(x, y, t_1, t_2)\right]dxdy \quad (1.17)$$

6) Hàm hiệp phương sai:

$$c_{xy}(t_1, t_2) = M\left[\left(\boldsymbol{x}(t_1) - m_x(t_1)\right)\left(\boldsymbol{y}(t_2) - m_x(t_2)\right)\right] =$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left[(x - m_x(t_1))\cdot\left(y - m_y(t_1)\right)\cdot f(x, y, t_1, t_2)\right]dxdy \quad (1.18)$$

Có thể kiểm chứng được ngay rằng

$$c_x(t_1, t_2) = r_x(t_1, t_2) - m_x(t_1)\cdot m_x(t_2) \quad (1.19)$$

$$c_{xy}(t_1, t_2) = r_{xy}(t_1, t_2) - m_x(t_1)\cdot m_{xy}(t_2) \quad (1.20)$$

Hai quá trình ngẫu nhiên $\boldsymbol{x}(t)$ và $\boldsymbol{y}(t)$ được gọi là *không tương quan*, nếu

$$c_{xy}(t_1, t_2) = 0, \text{ tức là } \quad r_{xy}(t_1, t_2) = m_x(t_1)\cdot m_y(t_2). \quad (1.21)$$

Một quá trình ngẫu nhiên $\boldsymbol{x}(t)$, nếu có các tham số ngẫu nhiên không phụ thuộc vào điểm gốc thời gian, tức là không thay đổi giá trị khi trục thời gian được tịnh tiến một khoảng τ bất kỳ, thì quá trình đó được gọi là *quá trình ngẫu nhiên dừng*.

Một quá trình ngẫu nhiên dừng $\boldsymbol{x}(t)$ có các tính chất sau:

a) $f(x, t) = f(x, t+\tau)$ với mọi $\tau \in \mathbb{R}$. (1.22)

b) $m_x(t) =$ hằng số $=: m_x$, trong đó ký hiệu $=:$ chỉ phép gán. (1.23)

c) $r_x(t_1, t_2) = r_x(0, t_2 - t_1) =: r_x(\tau)$. (1.24)

d) $c_x(t_1, t_2) =: c_x(\tau) = r_x(\tau) - m_x^2$. (1.25)

e) $\sigma_x^2(t) = r_x(0) - m_x^2 =$ hằng số $=: \sigma_x^2$. (1.26)

Hai quá trình ngẫu nhiên $\boldsymbol{x}(t)$ và $\boldsymbol{y}(t)$ được gọi là *cùng nhau dừng*, nếu chúng là những quá trình dừng và hàm hỗ tương quan $r_{xy}(t_1,t_2)$ không thay đổi giá trị khi tịnh tiến trục thời gian một khoảng τ bất kỳ, tức là

$$r_{xy}(t_1, t_2) = r_{xy}(0, t_2 - t_1) =: r_{xy}(\tau) = M\left[\boldsymbol{x}(t)\boldsymbol{y}(t+\tau)\right] \quad (1.27)$$

Có thể thấy ngay được rằng, với hai quá trình cùng nhau dừng $\mathcal{x}(t)$, $\mathcal{y}(t)$ có:

$$c_{xy}(t_1, t_2) =: c_{xy}(\tau) = r_{xy}(\tau) - m_x m_y \ . \tag{1.28}$$

Một quá trình ngẫu nhiên $\mathcal{x}(t)$, nếu các tham số ngẫu nhiên thay vì phải xác định từ toàn bộ tập hợp $\mathcal{x}(t)$ có thể được xác định chỉ với một phần tử đại diện $x(t)$ bất kỳ của tập, được gọi là *quá trình ngẫu nhiên egodic*. Những quá trình ngẫu nhiên egodic phải là các quá trình dừng (điều ngược lại không đúng) và có các tính chất sau:

a) $$m_x = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} x(t)dt \ . \tag{1.29}$$

b) $$r_x(\tau) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} x(t)x(t+\tau)dt \ . \tag{1.30}$$

c) $r_x(\tau)$ là hàm chẵn và $r_x(0) \geq |r_x(\tau)|$. (1.31)

d) $$\lim_{\tau\to\infty} r_x(\tau) = m_x^2 \tag{1.32}$$

e) $$r_{xy}(\tau) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} x(t)y(t+\tau)dt \ . \tag{1.33}$$

f) $$r_{xy}(-\tau) = r_{yx}(\tau) \tag{1.34}$$

g) $$|r_{xy}(\tau)| \leq \frac{1}{2}\left[r_x(0) + r_y(0)\right]. \tag{1.35}$$

h) $\lim_{\tau\to\infty} r_{xy}(\tau) = m_x m_y$, nếu $x(t)$ và $y(t+\tau)$ khi $\tau \to \infty$ không tương quan.

Các quá trình ngẫu nhiên được xét trong kỹ thuật thường được giả thiết là các quá trình egodic và từ nay về sau, mọi quá trình ngẫu nhiên trong quyển sách này, nếu không nói một cách chi tiết sẽ được hiểu là quá trình egodic.

Ảnh Fourier $S_x(j\omega)$ của hàm tự tương quan $r_x(\tau)$ của quá trình ngẫu nhiên egodic $\mathcal{x}(t)$ được gọi là *mật độ phổ hợp của tín hiệu*. Do $r_x(\tau)$ là một hàm chẵn nên $S_x(j\omega)$ là một hàm thực (xem phần bài tập trong chương sau). Bởi vậy thay vì $S_x(j\omega)$ người ta thường chỉ viết $S_x(\omega)$.

Ảnh Fourier $S_{xy}(j\omega)$ của hàm hỗ tương quan $r_{xy}(\tau)$ giữa hai quá trình ngẫu nhiên egodic $\mathcal{x}(t)$, $\mathcal{y}(t)$ được gọi là *mật độ phổ chéo của tín hiệu*. Chú ý rằng khác với mật độ phổ hợp $S_x(\omega)$, mật độ phổ chéo $S_{xy}(j\omega)$ nói chung là một số phức. Định nghĩa về ảnh Fourier của một hàm thời gian cũng như tính chất của nó sẽ được trình bày trong chương tiếp theo.

Một quá trình ngẫu nhiên egodic $\mathcal{x}(t)$ có hàm tự tương quan dạng "hàm" dirac

$$r_x(\tau) = k\delta(\tau),$$

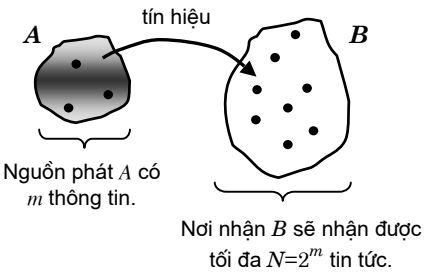
nói cách khác nó có mật độ phổ hợp là một hằng số

$$S_x(j\omega) = k,$$

thì quá trình ngẫu nhiên đó được gọi là *quá trình ồn trắng*. Mỗi phần tử của một quá trình ồn trắng có tên là *tín hiệu ồn trắng*.

1.3.3 Đại lượng đánh giá lượng thông tin có trong nguồn phát tín hiệu ngẫu nhiên

Một tín hiệu có khả năng truyền tải được nhiều thông tin cùng một lúc. Để tránh nhầm lẫn, ta sẽ gọi những thông tin được phát cùng một lúc trên đường tín hiệu là *một tin tức*. Qua đường tín hiệu, tin tức được truyền từ nơi phát (nguồn thông tin) đến nơi nhận. Vậy làm thế nào mà chỉ từ nơi nhận tin tức và cũng chỉ thông qua tín hiệu nhận được ta có thể đánh giá được lượng thông tin của nguồn phát?



Hình 1.11: Đánh giá lượng thông tin của nguồn phát từ số tin tức nhận được tại nơi nhận.

Hãy xem minh họa ở hình 1.11 làm ví dụ. Giả sử nguồn phát A có m thông tin (tập A có m phần tử). Tin tức sẽ được "chế biến" từ m thông tin này và gửi sang nơi nhận B. Mỗi tin tức là một tập con. Số các tin tức (hay tập con của A) có i thông tin (phần tử) chính là C_m^i , i=0,1, … , m. Bởi vậy số tin tức tối đa mà B có thể nhận được bằng

$$N = \sum_{i=0}^{m} C_m^i = 2^m$$

Đặt ngược lại vấn đề. Nếu B nhận được N tin tức thì lượng thông tin của nguồn phát A ít nhất là $\log_2 N$. Người ta nói lượng thông tin mà B nhận được từ A qua N tin tức là

$$\widetilde{H}_1 = \log_2 N \tag{1.36}$$

Như vậy đại lượng $\widetilde{H}_1$ là một độ đo cho lượng thông tin của nguồn phát A thông qua tập các tin tức mà B nhận được. Đại lượng này có tên là *Entropie loại* 1 với đơn vị đo là bit.

Ký hiệu N các tin tức mà B nhận được là x_1, … , x_N và có để ý đến xác suất $p(x_i)$ mà tin tức thứ i được truyền, năm 1947 Shannon đã mở rộng công thức (1.36) để xác định lượng thông tin nguồn phát có tính đến tính ngẫu nhiên phát tin như sau

$$\widetilde{H}_2 = -\sum_{i=1}^{N} p(x_i)\log_2 p(x_i) \tag{1.37}$$

và được gọi là *Entropie loại* 2. Trường hợp tất cả N tin tức $x_1, \dots, x_N$ được phát với cùng xác suất như nhau

$$p(x_1) = \cdots = p(x_N) = \frac{1}{N}$$

thì với (1.37) có

$$\widetilde{H}_2 = -\sum_{i=1}^{N} p(x_i)\log_2 p(x_i) = -\sum_{i=1}^{N}\frac{1}{N}\log_2\frac{1}{N} = \log_2 N = \widetilde{H}_1$$

và do đó (1.37) không mâu thuẫn với (1.36).

Xuất xứ, hai công thức (1.36), (1.37) được định nghĩa như là một độ đo cho lượng thông tin của một nguồn phát. Thực tế, chúng lại có ý nghĩa sử dụng nhiều hơn trong việc so sánh và xác định xem nguồn phát nào nhiều thông tin hơn trong số các nguồn phát đã có. Mặt khác việc tính $\ln N$ lại thông dụng hơn $\log_2 N$. Bởi vậy ý nghĩa sử dụng của (1.36), (1.37) sẽ không thay đổi nếu chúng được nhân với hằng số $\ln 2$ để xuất hiện $\ln N$. Ta đi đến công thức "cải biên" có tính phổ thông hơn như sau:

$$H_1 = \ln N \tag{1.38}$$

và
$$H_2 = -\sum_{i=1}^{N} p(x_i)\ln p(x_i) \tag{1.39}$$

Tương tự, khái niệm Entropie loại 1 đã được Smylie định nghĩa thành đại lượng đo lượng thông tin có trong nguồn phát một quá trình ngẫu nhiên (egodic) $\mathscr{x}(t)$, mà ở đó mỗi một tin tức $x(t)$ là một hàm liên tục theo t, như sau

$$H_1 = \int_{-\infty}^{\infty} \ln S_x(\omega)\, d\omega\,. \tag{1.40}$$

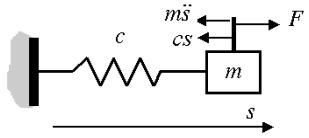
Cũng như vậy, khái niệm Entropie loại 2 cho lượng thông tin của nguồn phát quá trình ngẫu nhiên $\mathscr{x}(t)$ là:

$$H_2 = -\int_{-\infty}^{\infty} S_x(\omega)\ln S_x(\omega)\, d\omega\,. \tag{1.41}$$

Câu hỏi ôn tập và bài tập

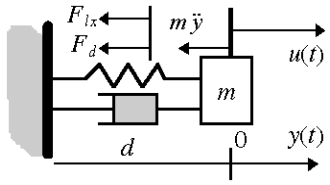
1. Thế nào là một bài toán nhận dạng bị động mô hình có tham số, một bài toán nhận dạng chủ động mô hình không có tham số?. Nêu ví dụ minh họa.
2. Xét đối tượng điều khiển là một hệ cơ gồm lò xo c và một vật khối lượng m. Vật sẽ chuyển động trên trục nằm ngang dưới tác động của lực F (hình 1.12). Lực F được xem như là tín hiệu vào và quãng đường s mà vật đi được là tín hiệu ra (đáp ứng

của đối tượng). Hãy xác định lớp các mô hình thích hợp mô tả đối tượng. Để có được một mô hình cụ thể tương đối chính xác cho đối tượng thì người ta phải làm gì thêm?.



Hình 1.12: Cho bài tập 2.

3. Hãy xác định lớp các mô hình thích hợp cho hệ cơ ở hình 1.13 gồm 1 lò xo, một vật có khối lượng m và khâu suy giảm vận tốc d. Tín hiệu vào của hệ là lực $u(t)$ tác động vào vật, tín hiệu ra là quãng đường $y(t)$ mà vật đi được. Để có được một mô hình cụ thể tương đối chính xác cho đối tượng thì người ta phải làm gì thêm và như thế nào?.

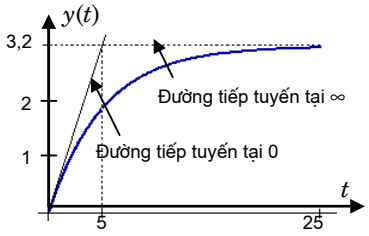


Hình 1.13: Cho bài tập 3.

4. Giả sử rằng từ các thông tin A–priori ban đầu người ta đã xác định được rằng một đối tượng tuyến tính sẽ có mô hình dạng

$$G(s) = \frac{k}{1+Ts}$$

Kích thích đối tượng bằng một tín hiệu $1(t)$ tại đầu vào người ta thu được đáp ứng $y(t)$ như ở hình 1.14. Hãy tìm các tham số k và T cho mô hình trên.



Hình 1.14: Cho bài tập 4.

5. Hãy chứng minh các công thức (1.22)÷(1.26) về tính chất của quá trình ngẫu nhiên dừng và (1.29)÷(1.35) về tính chất của quá trình ngẫu nhiên egodic.
6. Có thể nói một quá trình ngẫu nhiên egodic cũng là một quá trình ngẫu nhiên dừng được không và tại sao?
7. Một quá trình ngẫu nhiên dừng có phải là một quá trình ngẫu nhiên egodic không và tại sao?
8. Một người đi xe máy có 4 số s_1=0, s_2=1, s_3=2 và s_4=3. Biết rằng xác suất sử dụng các số của người đó là $p(s_1)$=0,08; $p(s_2)$=0,12 ; $p(s_3)$=0,3 và $p(s_4)$=0,5. Hãy tính lượng thông tin (*Entropie*) đánh giá việc sử dụng số của người lái xe này.

2 NHẬN DẠNG MÔ HÌNH KHÔNG THAM SỐ NHỜ PHÂN TÍCH PHỔ TÍN HIỆU

Ký hiệu $h(t)$ là hàm quá độ của đối tượng, tức là đáp ứng của đối tượng khi được kích thích bởi tín hiệu Heaviside tại đầu vào

$$1(t) = \begin{cases} 1 \text{ khi } t \geq 0 \\ 0 \text{ khi } t < 0 \end{cases}$$

Theo định nghĩa nêu ngay từ chương mở đầu thì việc mô hình hóa đối tượng chính là việc mô tả ánh xạ giữa tín hiệu vào $u(t)$ và tín hiệu ra $y(t)$:

$$T_M: u(t) \mapsto y(t)$$

Với đối tượng tuyến tính, ánh xạ trên sẽ được mô tả bởi tích phân Duhamel:

$$y(t) = \frac{d}{dt}\int_0^t h(t-\tau)u(\tau)d\tau ,$$

tức là thông qua $h(t)$ ta luôn xác định được $y(t)$ từ tín hiệu đầu vào $u(t)$. Bởi vậy hàm quá độ $h(t)$ đã mô tả đầy đủ đối tượng và do đó nó có thể được xem như là một mô hình (không tham số) của đối tượng.

Cũng như vậy, nếu gọi $g(t)$ là hàm trọng lượng, tức là đáp ứng của đối tượng khi được kích thích bởi tín hiệu dirac $\delta(t)$ tại đầu vào

$$\delta(t) = \frac{dh(t)}{dt} \tag{2.1}$$

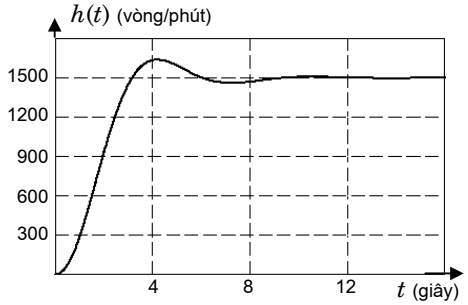
thì ánh xạ $T_M : u(t) \mapsto y(t)$ mô tả quan hệ vào/ra sẽ là

$$y(t) = u(t)h(0) + \int_0^t u(\tau)g(t-\tau)d\tau .$$

Nói cách khác, giống như $h(t)$, thông qua $g(t)$ ta luôn có được $y(t)$ từ $u(t)$ và do đó $g(t)$ cũng có thể được xem như là một mô hình không tham số của đối tượng.

Như vậy, việc nhận dạng mô hình không tham số sẽ đồng nghĩa với việc nhận dạng hàm quá độ $h(t)$ hay hàm trọng lượng $g(t)$. Một trong những phương pháp nhận dạng mô hình không tham số đơn giản nhất cho đối tượng tuyến tính là phương pháp chủ động (*active*) xác định hàm quá độ $h(t)$ bằng cách kích thích đối tượng với tín hiệu Heaviside

tại đầu vào rồi đo tín hiệu đầu ra. Hình 2.1 là một ví dụ minh họa kết quả nhận dạng chủ động mô hình không tham số $h(t)$ của động cơ xoay chiều ba pha theo cách thức vừa trình bày trong trường hợp lý tưởng rằng không có nhiễu tác động vào đối tượng và các phép đo là chính xác 100%.



Hình 2.1: Kết quả nhận dạng chủ động mô hình không tham số cho động cơ xoay chiều ba pha.

Tuy nhiên, không phải lúc nào cũng có các điều kiện lý tưởng để áp dụng được phương pháp nhận dạng chủ động. Trong bài toán nhận dạng thực tế người ta luôn phải tính tới khả năng có nhiễu tác động lên đối tượng, cũng như lỗi hoặc nhiễu có lẫn trong giá trị của phép đo tín hiệu dẫn tới kết quả thu được có sai số ảnh hưởng tới sự ứng dụng sau này của mô hình. Mặt khác bài toán nhận dạng chủ động luôn yêu cầu đối tượng phải được tách rời khỏi hệ thống và phải được kích thích bằng một tín hiệu chọn trước mà điều này không phải lúc nào cũng thực hiện được. Từ lý do đó, ở đây chúng ta sẽ không đi sâu thêm vào những phương pháp chủ động nhận dạng mô hình không tham số với những tín hiệu mẫu đặt trước ở đầu vào, mà thay vào đó là các phương pháp nhận dạng bị động (*passive*) có khả năng loại bỏ ảnh hưởng của nhiễu trong quá trình nhận dạng và một trong các phương pháp như vậy là nhận dạng bằng phân tích phổ tín hiệu.

Với (2.1) thì từ $g(t)$ ta cũng suy ra được $h(t)$ và ngược lại nên bài toán nhận dạng mô hình không tham số sẽ được gọi là đã giải quyết xong nếu như đã xác định được $g(t)$ hay ảnh Fourier $G(j\omega)$ của nó. Chương này sẽ trình bày các phương pháp nhận dạng đường đặc tính tần $G(j\omega)$ cũng như dãy các giá trị $\{G_n\}$ với $G_n = G(jn\Omega)$ cho một đối tượng tuyến tính cần nhận dạng trên cơ sở quan sát cả hai tín hiệu vào và ra (nhận dạng *passive*/ hay on–line), bao gồm các nội dung sau:

– Nhận dạng mật độ phổ tín hiệu bằng kỹ thuật DFT. Đánh giá sai số và các kỹ thuật làm giảm sai số.
– Xác định hàm trọng lượng, hàm quá độ từ phổ tín hiệu theo thuật toán cực tiểu sai lệch.

Sau khi đã có đường đặc tính tần $G_n = G(jn\Omega)$ mô tả đối tượng, trong chương 3 chúng ta sẽ làm quen tiếp các phương pháp khác nhằm xác định cấu trúc mô hình cũng như tham số của nó từ mô hình không tham số đã thu được.

2.1 Toán tử Fourier rời rạc (DFT)

Toán tử Fourier rời rạc (*discret Fourier transformation* – DFT) là một trường hợp đặc biệt của toán tử Fourier liên tục (CFT), được sử dụng để phân tích tín hiệu rời rạc có thời gian sống hữu hạn, ví dụ như những tín hiệu chỉ tồn tại trong một miền thời gian giới nội $[0, T)$, ngoài miền này thì được coi là bằng 0.

Những tín hiệu rời rạc, có thời gian sống hữu hạn $x(t)$ đều có thể biểu diễn được dưới dạng một dãy số hữu hạn $\{x_k\}$, $k = 0, 1, \ldots, N-1$ rất tiện cho việc xử lý chúng trên máy tính. Sự bùng nổ ứng dụng của máy tính trong các ngành kỹ thuật hiện nay đã đưa DFT lên vị trí số một trong tự động hóa phân tích và xử lý thông tin, tín hiệu.

Tuy nhiên, phần lớn các tín hiệu có trong tự nhiên lại đều tồn tại dưới dạng liên tục, và chỉ vì để tiện cho khâu tự động xử lý chúng mà người ta đã phải rời rạc hóa cũng như hữu hạn hóa thời gian sống của nó, biến nó thành một tín hiệu rời rạc, có miền xác định giới nội. Điều đó đã làm cho ảnh Fourier, hoặc phổ của tín hiệu thu được nhờ toán tử DFT, có chứa sai số so với phổ tính bằng toán tử Fourier liên tục. Việc phân tích sai số của DFT và đề ra các phương pháp làm giảm sai số đó là nhiệm vụ chính của sự ứng dụng toán tử DFT.

Sau đây, một số kết quả chính của việc nghiên cứu này sẽ được trình bầy bao gồm:

- DFT là dạng toán tử Fourier liên tục (CFT) theo quan điểm hàm mở rộng,
- sai số của DFT so với CFT do việc rời rạc hóa tín hiệu sinh ra – hiệu ứng trùng phổ (*aliasing*),
- sai số của DFT so với CFT do việc hữu hạn hóa thời gian sống sinh ra – hiệu ứng rò rỉ (*leakage*),

trong đó DFT sẽ được xây dựng bằng cách xem nó là toán tử Fourier liên tục trên quan điểm lý thuyết hàm mở rộng dirac $\delta(t)$.

2.1.1 Hàm mở rộng dirac

Có thể nói rằng, nhờ có lý thuyết hàm mở rộng (*distributions*) mà bản chất toán học của "hàm số" dirac $\delta(t)$ mới được hiểu một cách chặt chẽ và tổng quát. Thế tại sao không có khái niệm hàm mở rộng thì bản chất hàm dirac lại chưa được hiểu chặt chẽ?. Ta hãy xét một ví dụ minh họa.

Ví dụ: Từ tính chất

$$\mathcal{L}\{\frac{dx}{dt}\} = sX(s) - x(+0)$$

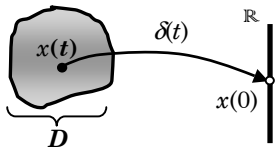
của toán tử Laplace, trong đó $\mathcal{L}$ là ký hiệu chỉ phép biến đổi Laplace và $X(s)$ là ảnh Laplace của $x(t)$, ta sẽ có với $x(t)=1(t)$ điều phi lý sau:

$$\mathcal{L}\{\delta(t)\} = s\frac{1}{s} - 1 \quad \Leftrightarrow \quad 1 = 0.$$

Điều phi lý trên đã dẫn người ta tới suy nghĩ rằng công thức chính xác phải là:

$$\mathcal{L}\{\frac{dx}{dt}\} = sX(s) - x(-0) = sX(s) \qquad (\text{vì } x(t) \equiv 0 \text{ khi } t < 0)$$

và công thức này có thể được chứng minh chặt chẽ với khái niệm hàm mở rộng. ❑



Hình 2.2: Hàm là ánh xạ từ $\boldsymbol{D}$ vào $\mathbb{R}$.

Khái niệm "hàm mở rộng" của $\delta(t)$ xuất phát từ bản chất "không hàm số" của nó, chẳng hạn nó không đúng như định nghĩa toán học kinh điển là ánh xạ từ $\mathbb{R}$ vào $\mathbb{R}$. Thực chất nó là một phép tính chuyển đổi hàm liên tục $x(t)$ thành số thực.

Nếu gọi $\boldsymbol{D}$ là tập tất cả các hàm $x(t)$ liên tục, có

$$\text{supp } x(t) = \{\, t \in \mathbb{R} \mid x(t) \neq 0 \,\}$$

giới nội trên $\mathbb{R}$ (hàm có thời gian sống hữu hạn) thì hàm mở rộng dirac $\delta(t)$ được hiểu là một ánh xạ liên tục, tuyến tính, từ $\boldsymbol{D}$ vào $\mathbb{R}$ (hình 2.2) như sau

$$x(t) \xrightarrow{\delta(t)} x(0) := \int_{-\infty}^{\infty} \delta(t)x(t)dt\,, \tag{2.2}$$

trong đó dấu tích phân không có ý nghĩa toán học như tích phân *Riemann* hay *Lebegues* mà thuần túy nó chỉ là một biểu tượng nói rằng phép tính $\delta(t)$:$\boldsymbol{D} \to \mathbb{R}$ có các phép biến đổi trong tính toán (cộng, trừ, nhân, chia, đạo hàm, …) giống như một tích phân. Chẳng hạn như các phép tính sau:

1) $$\int_{-\infty}^{\infty} \delta(t-\tau)x(t)dt = \int_{-\infty}^{\infty} \delta(t)x(t+\tau)dt \tag{2.3a}$$

2) $$\int_{-\infty}^{\infty} \dot{\delta}(t)x(t)dt = -\int_{-\infty}^{\infty} \delta(t)\dot{x}(t)dt = -\dot{x}(0) \tag{2.3b}$$

3) $$\int_{-\infty}^{\infty} \delta(at)x(t)dt = \frac{1}{|a|}\int_{-\infty}^{\infty} \delta(t)x\left(\frac{t}{a}\right)dt \quad \text{hay} \quad \delta(at) = \frac{1}{|a|}\,\delta(t) \tag{2.3c}$$

Cũng từ các tính chất này mà công thức (2.2) định nghĩa hàm dirac $\delta(t)$ còn được viết thành

$$x_k = x(kT_a) = \int_{-\infty}^{\infty} \delta(t - kT_a)x(t)dt = \int_{-\infty}^{\infty} \delta(t)x(t + kT_a)dt\,. \tag{2.4}$$

Những khái niệm tổng quát về hàm mở rộng nói chung (không riêng hàm dirac $\delta(t)$) bạn đọc có thể tìm thấy trong chương 5 của quyển sách này.

2.1.2 Mô hình hóa quá trình rời rạc tín hiệu

Công thức định nghĩa (2.4) được gọi là công thức trích mẫu tín hiệu $x(t)$ tại thời điểm $t=kT_a$. Nhằm thống nhất ý nghĩa về ký hiệu trích mẫu tín hiệu $x(kT_a)$ cho cả hai cách viết với hàm dirac $\delta(t)$ và với hàm Kronecker $k(t)$

$$k(t) = \begin{cases} 1 & \text{khi } t=0 \\ 0 & \text{khi } t \neq 0 \end{cases}$$

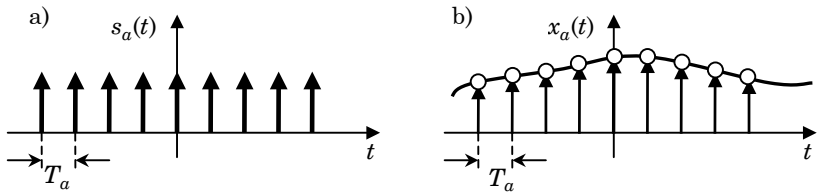
tức là

$$x_k = x(kT_a) = k(t-kT_a)x(t) = k(t-kT_a)x(kT_a),$$

từ nay về sau công thức định nghĩa (2.4) sẽ được viết lại thành

$$x_k = x(kT_a) = \delta(t-kT_a)x(t) = \delta(t-kT_a)x(kT_a), \tag{2.5}$$

trong đó phép "nhân" giữa hàm dirac $\delta(t)$ với một hàm thường $x(t)$ phải hiểu là phép tích phân (2.2).



Hình 2.3: a) Đồ thị hàm răng lược.
b) Mô hình hóa quá trình trích mẫu tín hiệu.

Để thu được dãy các giá trị $\{x_k\}$ với $x_k = x(kT_a)$, tức là dãy biểu diễn các giá trị của tín hiệu $x(t)$ tại những thời điểm … , $-T_a$, 0 , T_a , … , từ (2.5) ta có

$$\{x_k\} := \sum_{k=-\infty}^{\infty} x(t)\delta(t - kT_a) = x(t) \sum_{k=-\infty}^{\infty} \delta(t - kT_a)\,. \tag{2.6}$$

Nếu ký hiệu $x_a(t)$ là dãy $\{x_k\}$ và $s_a(t)$ là hàm mở rộng

$$s_a(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT_a)\,, \tag{2.7}$$

thì việc rời rạc hóa $x(t)$ thành dãy giá trị $\{x_k\}$ biểu diễn được qua tích

$$\{x_k\} := x_a(t) = x(t)s_a(t) \tag{2.8}$$

và do đó $x_a(t)$ cũng là một hàm mở rộng (tích của hàm mở rộng với hàm thường là một hàm mở rộng). Hàm $s_a(t)$ còn được gọi là *hàm răng lược* (hay *hàm rời rạc hóa*).

2.1.3 Ảnh Fourier của hàm mở rộng

Cho đến lúc này, khái niệm hàm dirac $\delta(t)$ đã được tóm tắt tương đối đầy đủ, về định nghĩa, tính chất, và quan trọng hơn cả là về các phép biến đổi được xây dựng trên ý nghĩa hàm mở rộng. Tuy nhiên, để có thể áp dụng được chúng vào việc mô tả DFT như một toán tử Fourier liên tục trên D thì vẫn còn thiếu khái niệm về ảnh Fourier của hàm mở rộng mà trong một vài tài liệu khác nhau còn được gọi là *toán tử Fourier mở rộng*.

Từ nay về sau ta sẽ thống nhất với nhau rằng, khi nói đến tín hiệu $x(t)$, thì $x(t)$ được hiểu là một *hàm thường và liên tục từng khúc*. Nếu tín hiệu $x(t)$ liên tục tại T_a thì giá trị $x(T_a)$ của tín hiệu $x(t)$ tại thời điểm T_a theo nghĩa hàm mở rộng sẽ là

$$x(T_a) := \delta(t-T_a)x(t) = x(T_a)\,\delta(t-T_a)\,.$$

Bây giờ ta xét toán tử Fourier. Cho tín hiệu $x(t)$. Nếu $x(t)$ thỏa mãn:

1) $\int_{-\infty}^{\infty} |x(t)|dt < \infty$ (tức là một số hữu hạn),

2) trong một khoảng giới nội bất kỳ liên tục từng khúc chỉ có hữu hạn các điểm cực trị,

3) tại điểm không liên tục t_0 thỏa mãn

$$x(t_0) = \frac{1}{2}\,[x(t_0-0) + x(t_0+0)]$$

thì $x(t)$ có ảnh Fourier $X(j\omega)$ xác định bởi

$$X(j\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt \tag{2.9a}$$

và ngược lại $x(t)$ cũng được suy ra từ $X(j\omega)$ theo

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(j\omega)e^{j\omega t}d\omega\,. \tag{2.9b}$$

Có thể nhận thấy ngay rằng toán tử Fourier là một toán tử tuyến tính, tức là

$$a \cdot x(t) + b \cdot y(t) \mapsto a \cdot X(j\omega) + b \cdot Y(j\omega).$$

Cũng tương tự như đối với hàm thường, ảnh Fourier $X_a(j\omega)$ của hàm mở rộng $x_a(t)$ thu được nhờ tính chất tuyến tính của toán tử Fourier:

$$X_a(j\omega) = \int_{-\infty}^{\infty} x_a(t) e^{-j\omega t} dt = \int_{-\infty}^{\infty} \left(\sum_{k=-\infty}^{\infty} x(t)\delta(t - kT_a) \right) e^{-j\omega t} dt$$
$$= \sum_{k=-\infty}^{\infty} \left(\int_{-\infty}^{\infty} x(t)\delta(t - kT_a) e^{-j\omega t} dt \right),$$

và cùng với công thức định nghĩa (2.2) được:

$$X_a(j\omega) = \sum_{k=-\infty}^{\infty} x(kT_a) \underbrace{\int_{-\infty}^{\infty} \delta(t - kT_a) e^{-j\omega t} dt}_{e^{-j\omega kT_a}} = \sum_{k=-\infty}^{\infty} x_k e^{-j\omega kT_a} . \tag{2.10}$$

Theo (2.10), $X_a(j\omega)$ là một hàm tuần hoàn với chu kỳ $\Omega_a = \frac{2\pi}{T_a}$, tức là

$$X_a(j\omega) = X_a[j(\omega + n\Omega_a)],$$

trong đó n là số nguyên ($n \in \mathbb{Z}$).

2.1.4 Quan hệ giữa $X(j\omega)$ và $X_a(j\omega)$

Như vậy, thay cho kết quả là tìm ảnh Fourier $X(j\omega)$ của $x(t)$, sau khi thực hiện việc rời rạc hóa ta lại chỉ thu được $X_a(j\omega)$. Để có thể từ $X_a(j\omega)$ tính ngược ra được $X(j\omega)$ thì cần thiết phải xác định được mối quan hệ giữa $X(j\omega)$ và $X_a(j\omega)$.

Trước hết ta chứng minh hai định lý sau.

Định lý 2.1:

$$\delta(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \cos(\omega t) d\omega = \lim_{a \to \infty} \frac{\sin(at)}{\pi t} . \tag{2.11}$$

Chứng minh:

Xuất phát từ hai công thức định nghĩa (2.9b) và (2.9a) về toán tử Fourier ta có

$$x(0) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\omega) d\omega = \frac{1}{2\pi} \int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt \right) d\omega = \int_{-\infty}^{\infty} \left(\frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-j\omega t} d\omega \right) x(t) dt$$

và sau khi so sánh với (2.2) sẽ được

$$\delta(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-j\omega t}d\omega .$$

Nhưng vì

$$e^{-j\omega t} = \cos(\omega t) - j\sin(\omega t) \quad \text{và} \quad \int_{-\infty}^{\infty}\sin(\omega t)d\omega = 0$$

do hàm sin là hàm lẻ, nên cuối cùng ta có điều phải chứng minh thứ nhất:

$$\delta(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\cos(\omega t)d\omega . \tag{2.12}$$

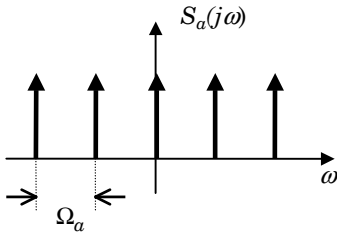
Tiếp tục, nếu viết (2.11) thành

$$\delta(t) = \lim_{a\to\infty}\frac{1}{2\pi}\int_{-a}^{a}\cos(\omega t)d\omega = \lim_{a\to\infty}\frac{\sin(\omega t)}{2\pi t}\Big|_{-a}^{a}$$

sẽ có được điều phải chứng minh thứ hai. ❑

Định lý 2.2 (Papoulis)**:** Ảnh Fourier $S_a(j\omega)$ của hàm răng lược $s_a(t)$ định nghĩa bởi (2.7) cũng là một hàm răng lược và có dạng (hình 2.4)

$$S_a(j\omega) = \Omega_a \sum_{n=-\infty}^{\infty}\delta(\omega - n\Omega_a)$$



Hình 2.4: Ảnh Fourier của hàm răng lược cũng là hàm răng lược.

Chứng minh:

Cũng lại xuất phát từ công thức định nghĩa toán tử Fourier thì

$$S_a(j\omega) = \sum_{n=-\infty}^{\infty} e^{-j\omega nT_a} \tag{2.13}$$

và do đó $S_a(j\omega)$ tuần hoàn với chu kỳ $\Omega_a = \frac{2\pi}{T_a}$. Bởi vậy sẽ đủ nếu ta chứng minh được rằng trong lân cận điểm 0 hàm $S_a(j\omega)$ có giá trị

$$S_a(j\omega) = \Omega_a\delta(\omega). \tag{2.14}$$

Áp dụng công thức tính tổng của một cấp số nhân cho (2.13)

$$S_a(j\omega) = \sum_{n=-\infty}^{\infty} e^{-j\omega nT_a} = \lim_{N\to\infty} \sum_{n=-N}^{N} e^{-j\omega nT_a}$$

$$= \lim_{N\to\infty} \frac{\sin(N+\frac{1}{2})\omega T_a}{\sin\frac{\omega T_a}{2}} = \underbrace{\left(\lim_{N\to\infty} \frac{\sin(N+\frac{1}{2})\omega T_a}{\pi\omega T_a} \right)}_{\delta(\omega T_a)} \frac{\pi\omega T_a}{\sin\frac{\omega T_a}{2}}$$

và theo công thức trích mẫu (2.5) thì điều này tương đương với

$$S_a(j\omega) = \delta(\omega T_a) \lim_{\omega T_a\to 0} \frac{\pi\omega T_a}{\sin\frac{\omega T_a}{2}}.$$

Kết hợp cùng tính chất (2.3c), suy ra được

$$S_a(j\omega) = 2\pi\cdot\delta(\omega T_a) = \frac{2\pi}{T_a}\,\delta(\omega).$$

và đó chính là công thức phải chứng minh. ❑

Trở về vấn đề chính là xác định mối quan hệ giữa $X(j\omega)$ và $X_a(j\omega)$. Nhớ lại một tính chất của toán tử Fourier liên tục là ảnh của tích hai hàm trong miền thời gian chính là tích chập của hai ảnh trong miền phức $x(t)y(t) \mapsto X(j\omega)*Y(j\omega)$, trong đó tích chập trong miền phức được định nghĩa bởi

$$X(j\omega)*Y(j\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\zeta)Y\left[j(\omega-\zeta)\right]d\zeta\,,$$

thì từ công thức (2.8) mô tả quá trình trích mẫu, được

$$X_a(j\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\zeta)S_a\left[j(\omega-\zeta)\right]d\zeta\,. \tag{2.15}$$

Thay công thức của định lý 2.2 vào (2.15) có

$$X_a(j\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\zeta)\left(\frac{2\pi}{T_a} \sum_{n=-\infty}^{\infty} \delta(\omega-\zeta-n\Omega_a)\right)d\zeta$$

$$= \frac{1}{T_a} \sum_{n=-\infty}^{\infty} \underbrace{\int_{-\infty}^{\infty} X(j\zeta)\delta(\omega-\zeta-n\Omega_a)d\zeta}_{X\left[j(\omega-n\Omega_a)\right]}$$

và cuối cùng ta đi đến định lý 2.3:

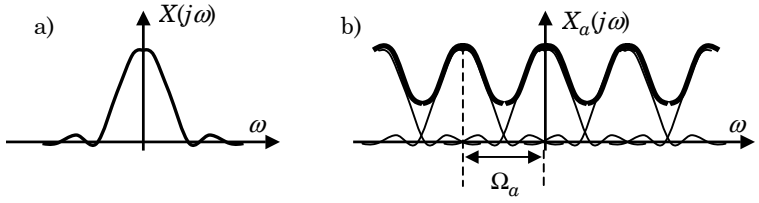
Định lý 2.3: Giữa ảnh Fourier $X_a(j\omega)$ của $x_a(t)$ và $X(j\omega)$ của $x(t)$ có mối quan hệ

$$X_a(j\omega) = \frac{1}{T_a} \sum_{n=-\infty}^{\infty} X\left[j(\omega-n\Omega_a)\right]\,. \tag{2.16}$$

(2.16) là công thức rất đẹp, mô tả quan hệ giữa hai ảnh $X_a(j\omega)$, $X(j\omega)$. Chỉ riêng công thức này đã nói lên được hiệu ứng trùng phổ của DFT và bản chất định lý Shannon sẽ được đề cập tới ngay trong mục tiếp theo.

2.1.5 Hiệu ứng trùng phổ và định lý Shannon

Hiệu ứng trùng phổ (aliasing) được trực tiếp nhìn thấy trong định lý 2.3 và hình 2.5 biểu diễn công thức (2.16). Đó là hiện tượng mà các thành phần thừa của $X(j\omega)$ có lẫn trong $X_a(j\omega)$ khi dịch trục tọa độ để thực hiện phép cộng (2.16).



Hình 2.5: a) Phổ $X(j\omega)$ của tín hiệu $x(t)$.
b) Phổ $X_a(j\omega)$ của tín hiệu $x_a(t)$, tức là của dãy $\{x_k\}$.

Để loại bỏ các thành phần thừa này thì từ công thức (2.16) của định lý 2.3 có

$$T_a \cdot X_a(j\omega) = X(j\omega) + \underbrace{\sum_{\substack{n=-\infty \\ n\neq 0}}^{\infty} X\left[j(\omega - n\Omega_a)\right]}_{\text{Bản chất hiệu ứng trùng phổ}}$$

và do đó hiện tượng trùng phổ sẽ mất nếu (hình 2.6)

$$\sum_{\substack{n=-\infty \\ n\neq 0}}^{\infty} X\left[j(\omega - n\Omega_a)\right] = 0 \qquad , \qquad -\frac{1}{2}\Omega_a \le \omega < \frac{1}{2}\Omega_a. \tag{2.17a}$$

Trong trường hợp $X(j\omega)=0$ khi $|\omega| > \frac{\Omega_a}{2}$ thì để có (2.17a) ta chỉ cần chọn

$$T_a \le \frac{2\pi}{\Omega_a}$$

và lúc này, phổ $X(j\omega)$ sẽ được suy ra từ $X_a(j\omega)$ một cách đơn giản như sau

$$X_a(j\omega) = \frac{1}{T_a} X(j\omega) \qquad \text{khi} \qquad -\frac{1}{2}\Omega_a \le \omega < \frac{1}{2}\Omega_a. \tag{2.17b}$$

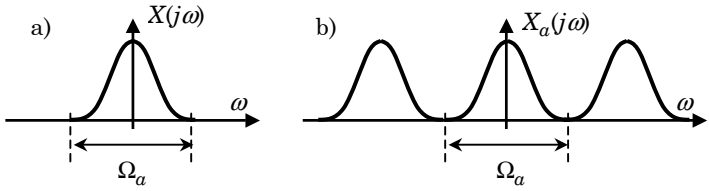
Từ những kết luận trên ta suy ra được:

Định lý 2.4 (Shannon): Nếu phổ $X(j\omega)$ của tín hiệu $x(t)$ đồng nhất bằng 0 ngoài miền giới nội $|\omega| \geq \left|\frac{\Omega_a}{2}\right|$, thì với chu kỳ trích mẫu

$$T_a \leq \frac{2\pi}{\Omega_a}. \tag{2.18}$$

ảnh $X(j\omega)$ của $x(t)$ sẽ suy ra được $X_a(j\omega)$ của $x_a(t)=\{x_k\}$ theo công thức (2.17b).

Công thức (2.18) được gọi là chu kỳ trích mẫu Nyquist và định lý 2.4 trên, ngoài tên gọi định lý Shannon, cũng còn có một gọi khác là Katelnikov, bởi lẽ Shannon và Katelnikov đã độc lập với nhau tìm ra nó.



Hình 2.6: Giải thích định lý Shannon–Khatelnikov và chu kỳ trích mẫu của Nyquist.

2.1.6 Hiệu ứng rò rỉ (leakage) và kỹ thuật hàm cửa sổ

Mặc dù đã rời rạc hóa tín hiệu $x(t)$ thành dãy $\{x_k\}= x_a(t)$, việc tính phổ $X_a(j\omega)$ theo công thức (2.10)

$$X_a(j\omega) = \sum_{k=-\infty}^{\infty} x_k e^{-j\omega k T_a} \tag{2.19}$$

vẫn chưa thể thực hiện được trên máy tính chỉ vì một lý do là dãy $\{x_k\}$ có vô số số hạng (k chạy từ $-\infty$ đến ∞) và điều này đồng nghĩa với việc thời gian quan sát (đo) tín hiệu T phải lớn vô cùng.

Để có thể cài đặt (2.19) thành thuật toán tính $X_a(j\omega)$ thì cần thiết phải cắt bớt những số hạng x_k khi k nằm ngoài khoảng quan sát $[0, T)$, hay nói cách khác phải hữu hạn hóa thời gian sống của tín hiệu $x(t)$, $x_a(t)$:

$$x(t) \mapsto \tilde{x}(t) = \begin{cases} x(t) & \text{khi } 0 \leq t < T \\ 0 & \text{khi } t \geq T \text{ hoặc } t < 0 \end{cases}$$

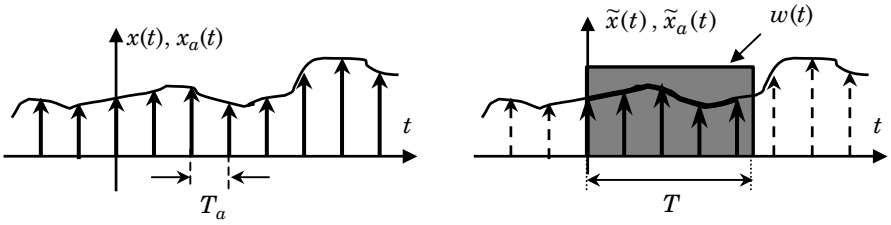
$$\Rightarrow \quad x_a(t) \mapsto \tilde{x}_a(t) = \begin{cases} x_a(t) & \text{khi } 0 \leq t < T \\ 0 & \text{khi } t \notin [0, T) \end{cases}$$

Quá trình hữu hạn hóa thời gian sống của tín hiệu $x(t)$, $x_a(t)$ thành $\tilde{x}(t)$, $\tilde{x}_a(t)$ theo các công thức trên có thể mô tả được một cách rất đơn giản bằng cách nhân chúng với hàm cửa sổ $w(t)$ định nghĩa như sau (hình 2.7)

$$w(t) = \begin{cases} 1 \text{ khi } 0 \le t < T \\ 0 \quad \text{khi } t \notin [0, T) \end{cases} \tag{2.20}$$

tức là

$$\tilde{x}(t) = w(t)x(t) \qquad \text{và} \qquad \tilde{x}_a(t) = w(t)x_a(t). \tag{2.21}$$



Hình 2.7: Mô tả quá trình hữu hạn hóa thời gian sống của tín hiệu.

Sau khi hữu hạn hóa thời gian sống của tín hiệu theo (2.21) và áp dụng công thức (2.19) để xác định ảnh Fourier $\tilde{X}_a(j\omega)$ của $\tilde{x}_a(t)$ ta có

$$\tilde{X}_a(j\omega) = \sum_{k=0}^{N} x_k e^{-j\omega k T_a}, \tag{2.22}$$

với N là số nguyên nhỏ nhất nhưng không nhỏ hơn $\frac{T}{T_a}$.

Như vậy, khi hữu hạn hóa thời gian sống của $x_a(t)$ với khoảng $[0, T)$ thành $\tilde{x}_a(t) = w(t)x_a(t)$, ta sẽ chỉ nhận được ảnh Fourier $\tilde{X}_a(j\omega)$

$$\tilde{X}_a(j\omega) = W(j\omega) * X_a(j\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} W(j\zeta) X_a[j(\omega - \zeta)] d\zeta$$

của $\tilde{x}_a(t)$ chứ không phải $X_a(j\omega)$ của $x_a(t)$ như mong đợi và giữa hai ảnh $\tilde{X}_a(j\omega)$, $X_a(j\omega)$ này tồn tại một sai lệch được gọi là sai lệch rò rỉ (leakage). Sai lệch này càng nhỏ nếu T càng lớn. Tuy nhiên không thể áp dụng phương pháp tăng thời gian qua sát T để giảm sai lệch rò rỉ, vì như thế ta sẽ quay trở lại bài toán đầu là thời gian quan sát tín hiệu vô cùng lớn.

Một giải pháp khác được sử dụng để giảm sai số rò rỉ là kỹ thuật hàm cửa sổ. Tư tưởng của phương pháp này nằm ở mô hình hữu hạn hóa thời gian sống tín hiệu theo công thức (2.21) và vấn đề đặt ra là phải tìm được hàm cửa sổ thích hợp $w(t)$ sao cho bình phương sai lệch

$$Q = \int_{-\infty}^{\infty} \left| X_a(j\omega) - \widetilde{X}_a(j\omega) \right|^2 d\omega$$

đạt giá trị nhỏ nhất. Đây chính là một bài toán tối ưu phi tuyến [12], [13]. Lời giải tổng thể cho bài toán tối ưu trên chưa có, song Geckinli, Yavus đã đưa ra trong [5] những tính chất cần có của nghiệm tối ưu $w^*(t)$ như sau:

1) $w^*(t+\frac{T}{2})$ là hàm chẵn, không âm,

2) $w^*(\frac{T}{2}) = 1$,

3) khi đạo hàm bậc m của $w^*(t)$ không liên tục tại một điểm $t \in [0,T]$ nào đó thì các điểm cực đại của đường đặc tính tần logarit của $w^*(t-\frac{T}{2})$ trong miền phức phải tiến đến 0 với vận tốc lớn hơn $6(m+1)$dB/dec.

Căn cứ theo ba tính chất trên người ta đã đưa ra bảng các hàm cửa sổ làm giảm sai lệch rò rỉ một cách hiệu quả thường được sử dụng như sau:

a) Xung vuông $w_0(t) = \begin{cases} 1 & \text{khi } 0 \le t < T \\ 0 & \text{khi } t \notin [0,T) \end{cases}$

b) Cosinus $w_1(t) = \begin{cases} \cos\dfrac{\pi(t-\frac{T}{2})}{T} & \text{khi } 0 \le t < T \\ 0 & \text{khi } t \notin [0,T) \end{cases}$

c) Bartlett $w_2(t) = \begin{cases} 1-2\left|\dfrac{t}{T}-\dfrac{1}{2}\right| & \text{khi } 0 \le t < T \\ 0 & \text{khi } t \notin [0,T) \end{cases}$

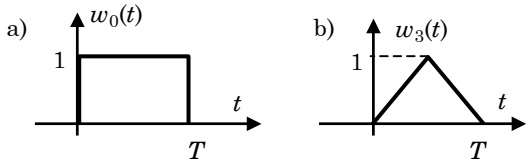
d) Hanning $w_3(t) = \begin{cases} 0{,}5+0{,}5\cos\dfrac{\pi(2t-T)}{T} & \text{khi } 0 \le t < T \\ 0 & \text{khi } t \notin [0,T) \end{cases}$

e) Hamming $w_4(t) = \begin{cases} 0{,}54+0{,}46\cos\dfrac{\pi(2t-T)}{T} & \text{khi } 0 \le t < T \\ 0 & \text{khi } t \notin [0,T) \end{cases}$

f) Blackman $w_5(t) = \begin{cases} 0{,}42+0{,}5\cos\dfrac{\pi(2t-T)}{T}+0{,}08\cos\dfrac{2\pi(2t-T)}{T} & \text{khi } 0 \le t < T \\ 0 & \text{khi } t \notin [0,T) \end{cases}$

g) Papoulis $w_6(t) = \begin{cases} \dfrac{\left|\sin\dfrac{\pi(2t-T)}{T}\right|}{\pi} + 1 - \dfrac{|2t-T|}{T}\cos\dfrac{\pi(2t-T)}{T} & \text{khi } 0 \le t < T \\ 0 \qquad \text{khi } t \notin [0,T) \end{cases}$

h) Parzen $w_7(t) = \begin{cases} 1 - 6\dfrac{(2t-T)^2}{T^2}\left(1 - \dfrac{|2t-T|}{T}\right) & \text{khi } 0 \le t < \dfrac{T}{4} \text{ hoặc } \dfrac{3T}{4} \le t < T \\ 2\left(1 - \dfrac{|2t-T|}{T}\right)^3 & \text{khi } \dfrac{T}{4} \le t < \dfrac{3T}{2} \\ 0 \qquad \text{khi } t \notin [0,T) \end{cases}$



Hình 2.8.: a) Xung vuông.
b) Hàm Bartlett

Sau khi đã chọn được một hàm cửa sổ thích hợp và nhân với dãy giá trị tín hiệu đo được là $\{x_k\}=x_a(t)$, giá trị tín hiệu cũ x_k trở thành

$$\tilde{x}_k = x_k\, w_i(kT_a) \qquad \text{với} \qquad i=0,1,\ \ldots\ ,7 \tag{2.23}$$

và khi đó phổ tìm được sẽ là

$$\tilde{X}_a(j\omega) = \sum_{k=0}^{N-1} \tilde{x}_k e^{-j\omega kT_a} \ . \tag{2.24}$$

trong đó N là số nguyên nhỏ nhất không nhỏ hơn $\dfrac{T}{T_a}$.

So sánh với hiện tượng trùng phổ mà ở đó sai lệch trùng phổ có thể được loại bỏ hoàn toàn cho lớp tín hiệu $x(t)$ có dải băng bị chặn bằng cách chọn chu kỳ trích mẫu T_a thỏa mãn công thức Nyquist (2.18), một câu hỏi sẽ được đặt ra ở đây là với lớp tín hiệu như thế nào thì không xuất hiện sai số rò rỉ?.

Để trả lời câu hỏi này, trước hết ta chứng minh định lý sau:

Định lý 2.5: Nếu $x(t)$ là một hàm tuần hoàn với chu kỳ T thì $x(t)$ biểu diễn được dưới dạng

$$x(t) = x_T(t) * s_T(t), \tag{2.25}$$

trong đó $x_T(t)$ là hàm xác định trong miền $[0, T)$ và tại đó $x_T(t) = x(t)$, $s_T(t)$ là hàm răng lược với khoảng cách bước răng là T, nói cách khác

$$x_T(t) = \begin{cases} x(t) & \text{khi} \quad t \in [0,T) \\ 0 & \text{khi} \quad t \notin [0,T) \end{cases}$$

và $$s_T(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT)$$

Chứng minh:

Xuất phát ngay từ công thức định nghĩa hàm $x_T(t)$ ta có

$$x(t) = \sum_{k=-\infty}^{\infty} x_T(t - kT) = \sum_{k=-\infty}^{\infty} \int_{-\infty}^{\infty} x_T(t-\tau)\delta(\tau - kT)d\tau$$

$$= \int_{-\infty}^{\infty} [x_T(t-\tau) \underbrace{\sum_{k=-\infty}^{\infty} \delta(t - kT)}_{S_T(\tau)}]d\tau = \int_{-\infty}^{\infty} x_T(t-\tau) S_T(\tau) d\tau$$

và đó chính là điều phải chứng minh. ❑

Nếu $x(t)$ là tín hiệu tuần hoàn với chu kỳ T thì khi rời rạc hóa thành $x_a(t)$, với chu kỳ lấy mẫu T_a được chọn sao cho $T = NT_a$ để trong một chu kỳ lấy được đúng N mẫu, thì $x_a(t)$ cũng tuần hoàn với chu kỳ T. Bởi vậy

$$x_a(t) = x_{T_a}(t) * s_T(t)$$

với $$x_{T_a}(t) = \begin{cases} x_a(t) \text{ khi } t \in [0,T) \\ 0 \text{ khi } t \notin [0,T) \end{cases}$$

hay $x_{T_a}(t)$ là dãy hữu hạn $x_{T_a}(t) = \{x_0, x_1, \ldots, x_{N-1}\}$ có N phần tử.

Suy ra

$$X_a(j\omega) = X_{T_a}(j\omega) S_T(j\omega) = \underbrace{\left(\sum_{k=0}^{N-1} x_k e^{-j\omega k T_a} \right)}_{X_{T_a}(j\omega)} \underbrace{\left(\Omega \sum_{k=0}^{N-1} \delta(t - n\Omega) \right)}_{S_T(j\omega)}$$

$$\Leftrightarrow \quad X_a(j\omega) = \Omega \sum_{n=-\infty}^{\infty} X_{T_a}(jn\Omega)\delta(\omega) .$$

Nói cách khác, $X_a(j\omega)$ là một hàm rời rạc, có giá trị tại những điểm tần số $n\Omega$, $n\in\mathbb{Z}$. Song vì $X_a(j\omega)$ tuần hoàn với chu kỳ

$$\Omega_a=\frac{2\pi}{T_a}=N\Omega$$

nên cuối cùng ta cũng chỉ cần xác định các giá trị của $X_a(j\omega)$ tại $\omega=n\Omega$ với $n=0, \cdots, N-1$ (trong vòng một chu kỳ) là đủ và những giá trị đó là

$$X_a(jn\Omega) = \Omega \sum_{k=0}^{N-1} x_k e^{-jkn\Omega T_a}, \qquad n=0, \cdots, N-1 \tag{2.26}$$

do đó không cần thiết phải hữu hạn hóa thời gian sống cho $x_a(t)$ vì tổng trên chỉ gồm có hữu hạn N số hạng.

Tổng kết lại, ta đi đến kết luận:

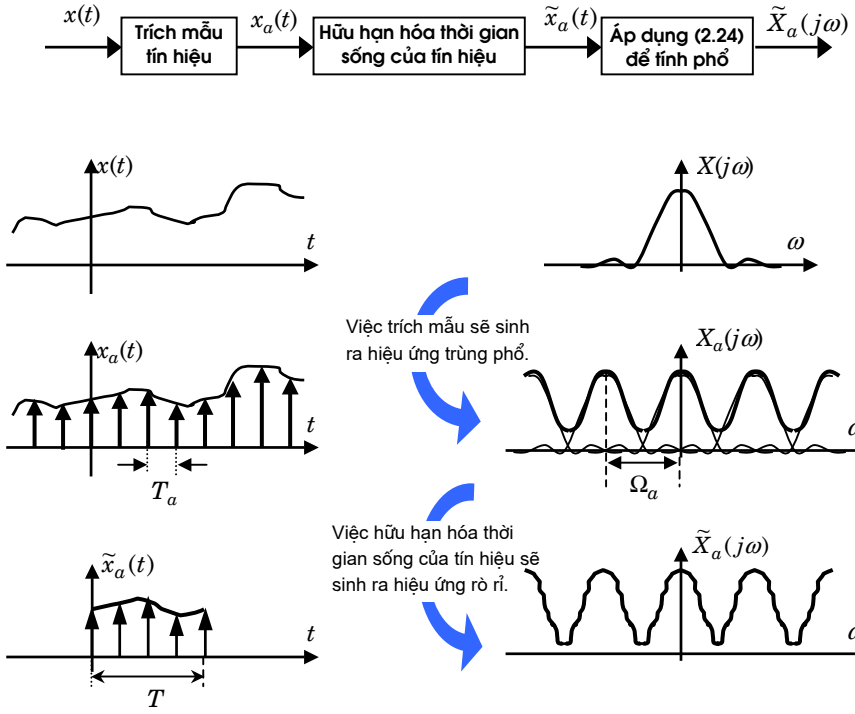
Định lý 2.6: Nếu $x(t)$ là một hàm tuần hoàn với chu kỳ T và $x(t)$ được rời rạc hóa thành $x_a(t)$ với chu kỳ lấy mẫu $T_a=\frac{T}{N}$ thì ảnh Fourier $X_a(j\omega)$ của $x_a(t)$ là tổng của hữu hạn các số hạng (nên không cần phải áp dụng kỹ thuật hàm của sổ và do đó không xuất hiện sai số rò rỉ). Ngoài ra $X_a(j\omega)$ là một hàm tuần hoàn với chu kỳ $\Omega_a=N\Omega$, có giá trị rời rạc tại những điểm $\omega=n\Omega$.

2.1.7 Kết luận về DFT và thuật toán FFT

Việc tính phổ $X(j\omega)$ của tín hiệu $x(t)$ theo thuật toán DFT đã cho ra kết quả $\widetilde{X}_a(j\omega)$ mà giữa chúng tồn tại hai loại sai lệch sinh ra bởi hiệu ứng trùng phổ (aliasing) và hiệu ứng rò rỉ (*leakage*). Hai sai lệch này đã được phân tích trong những mục trên thông qua hàm mở rộng $\delta(t)$.

Đối với những tín hiệu $x(t)$ thỏa mãn giả thiết của định lý Shannon, tức là các tín hiệu có dải băng bị chặn $X(j\omega)\equiv 0$ khi $|\omega|\geq\left|\frac{\Omega_a}{2}\right|$, ta có thể loại bỏ được sai lệch của hiệu ứng trùng phổ bằng cách chọn chu kỳ trích mẫu T_a theo công thức (2.18) của định lý Shannon.

Với tín hiệu không tuần hoàn, để giảm sai lệch của hiệu ứng rò rỉ ta áp dụng kỹ thuật hàm cửa sổ bằng cách chọn một hàm cửa sổ thích hợp trong số các hàm đã được liệt kê tại mục 2.1.6 thay vì chỉ đơn giản là cắt bớt những những giá trị $x_k = x(kT_a)$ nằm ngoài khoảng thời gian quan sát tín hiệu $[0, T)$.



Hình 2.9: Các bước thực hiện toán tử DFT và nguyên nhân của các sai số.

Sau khi đã chọn được chu kỳ trích mẫu và hàm cửa sổ, ta biến đổi tín hiệu $x(t)$ thành $\tilde{x}_a(t)$. Tiếp theo, áp dụng công thức (2.24) để xác định $\tilde{X}_a(j\omega)$ và tích $T_a\tilde{X}_a(j\omega)$ sẽ được xem như là phổ gần đúng của $x(t)$.

Các bước tính phổ gần đúng $\tilde{X}_a(j\omega)$ của tín hiệu liên tục $x(t)$ được tóm tắt trong sơ đồ ở hình 2.9.

Để dễ nhớ công thức tính $T_a\tilde{X}_a(j\omega)$ được xem như là kết quả gần đúng của $X(j\omega)$, có thể xuất phát từ công thức định nghĩa toán tử Fourier (2.9a), trong đó dấu $\int$ được thay bởi Σ, dt thay bởi T_a, đổi $x(t)$ thành tích $x_k\, w_k$ và có

$$X(j\omega) \approx T_a\tilde{X}_a(j\omega) = T_a \sum_{k=0}^{N-1} x_k w_k e^{-j\omega k T_a} \ .$$

Bây giờ ta đi vào vấn đề cài đặt thuật toán DFT với công việc chính là cài đặt công thức (2.24). Có thể thấy ngay rằng việc cài đặt trực tiếp (2.24) chưa thể được vì đầu ra

(kết quả) $\widetilde{X}_a(j\omega)$ của nó vẫn còn là một hàm liên tục. Nhằm khắc phục khó khăn đó, ta sẽ không tính trực tiếp $\widetilde{X}_a(j\omega)$ mà thay vào đó là $\widetilde{N}$ các giá trị rời rạc $\widetilde{X}_a(jn\Omega)$, $n=0,1,\ \dots\ ,\widetilde{N}-1$ của nó với Ω là chu kỳ trích mẫu trong miền phức. Công thức (2.24) trở thành

$$\widetilde{X}_a(jn\Omega) = \sum_{k=0}^{N-1} \widetilde{x}_k e^{-jn\Omega kT_a}\ , \qquad n=0,1,\ \dots\ ,\ \widetilde{N}-1 \tag{2.27}$$

Vậy với Ω như thế nào và với $\widetilde{N}$ lớn bao nhiêu bao nhiêu thì việc cài đặt sẽ thuận lợi mà không ảnh hưởng đến kết quả tính toán?. Trước hết, có thể thấy ngay rằng $\widetilde{N}$ càng lớn và Ω càng nhỏ thì càng tốt, song do có kết luận rằng $\widetilde{X}_a(j\omega)$ tuần hoàn với chu kỳ Ω_a nên thực chất chỉ cần các giá trị $\widetilde{X}_a(jn\Omega)$ trong một chu kỳ là đủ và sẽ là đủ để có $\widetilde{N}$ giá trị của $\widetilde{X}_a(jn\Omega)$ trong một chu kỳ khi

$$\Omega = \frac{\Omega_a}{\widetilde{N}} = \frac{2\pi}{\widetilde{N}T_a}\ . \tag{2.28}$$

Bởi vậy hiệu quả công việc tính toán chỉ phụ thuộc vào một mình $\widetilde{N}$ và với $\widetilde{N}$ càng lớn, kết quả càng tốt.

Thuật toán Fourier nhanh (được viết tắt thành FFT) là một thuật toán đặc biệt để thực hiện nhanh công thức (2.27). Thuật toán FFT do Cooley và Tukey tìm ra năm 1965. Từ đó tới nay thuật toán đã được bổ sung, phát triển thêm rất nhiều và đa dạng.

Về ý nghĩa thực tế của thuật toán FFT, Bergland đã nhận xét rất "kinh tế" trong [2] như sau: "*The fast Fourier transform algorithm can reduce the time involved in finding a discret Fourier transform from several minutes to less than a second and lower the cost from several dolars to several cents*".

Sau đây, ta sẽ làm quen với thuật toán FFT thời khai sinh do Cooley và Tukey tìm ra cho trường hợp

$$N= \widetilde{N} =2^p, \qquad \text{với} \qquad p\in \mathbb{N},$$

hay $N= \widetilde{N}$ là một số nguyên lũy thừa của 2. Các dạng khác của FFT cho những giá trị N, $\widetilde{N}$ bất kỳ, đọc giả quan tâm có thể tìm thấy ở tài liệu [3] của Brigham.

Khi có $N= \widetilde{N}$ thì $\widetilde{N}T_a = NT_a = T$ và do đó với (2.28) công thức (2.27) của DFT sẽ là

$$\widetilde{X}_a(jn\Omega) = \sum_{k=0}^{N-1} \widetilde{x}_k e^{-j\frac{2\pi}{N}nk}\ , n=0,1,\ \dots\ ,N-1 \tag{2.29}$$

Hãy xét (2.29) như là một đa thức $P_0(q_0)$ bậc N–1 của biến $q_0 = e^{-j\frac{2\pi}{N}}$ và các hệ số $\tilde{x}_k$, $k = 0, 1, \ldots, N$–1:

$$P_0(q_0) = \sum_{k=0}^{N-1} \tilde{x}_k q_0^k , \tag{2.30}$$

vậy N các giá trị $\tilde{X}_a(jn\Omega)$, $n = 0, 1, \ldots, N$–1 của cho $\tilde{X}_a(j\omega)$ trong một chu kỳ đương nhiên được xác định từ $P_0(q_0)$ qua

$$\tilde{X}_a(jn\Omega) = P_0(q_0^n) . \tag{2.31}$$

Định lý 2.5: Biến số q_0 trong công thức (2.31) là nghiệm phức của phương trình x^N=1 và được gọi là nghiệm bậc N của đường tròn đơn vị, thỏa mãn các tính chất sau:

a) $q_0 \neq 0$; $q_0^N = 1$ và $\sum_{k=0}^{N-1} q_0^{nk} = 0$; $\forall n \in \mathbb{Z}$.

b) nếu N=2m với $m \in \mathbb{N}$ thì $-q_0^n = q_0^{n+m}$ và q_0^2; cũng là nghiệm phức bậc m của đường tròn đơn vị.

Chứng minh:

Do $q_0^N = e^{-j2\pi} = \cos(2\pi) - j\sin(2\pi) = 1$ nên hiển nhiên q_0 là nghiệm phức bậc N của đường tròn đơn vị và $q_0 \neq 0$. Thay công thức của q_0 vào tổng trên ta được điều phải chứng minh thứ nhất

$$\sum_{k=0}^{N-1} q_0^{nk} = \frac{q_0^{nN} - 1}{q_0 - 1} = \frac{e^{-2jn\pi} - 1}{q_0^n - 1} = 0 .$$

Mặt khác, do

$$q_0^m = e^{-j\frac{2m\pi}{N}} = e^{-j\pi} = \cos(\pi) - j\sin(\pi) = -1$$

nên

$$-q_0^n = q_0^{n+m}$$

Bởi vậy với

$$q_0^{2nm} = q_0^{Nn} = 1$$

thì q_0^{2n} cũng là nghiệm phức bậc m của đường tròn đơn vị và đó chính là điều phải chứng minh thứ hai. ❑

Dựa vào định lý trên, với điều kiện $N=2m$; $m\in\mathbb{N}$, ta sẽ tách $P_0(q_0)$ (được gọi là đa thức mẹ) của (2.30) thành hai đa thức (gọi là các đa thức con) $P_{00}(q_1)$ bậc $m-1$ gồm các biến bậc chẵn và $P_{01}(q_1)$ cũng có bậc $m-1$ gồm các biến bậc lẻ như sau:

$$\begin{aligned} P_0(q_0) &= \sum_{k=0}^{m-1} \widetilde{x}_{2k} q_0^{2k} + \sum_{n=0}^{m-1} \widetilde{x}_{2k+1} q_0^{2k+1} \\ &= \sum_{k=0}^{m-1} \widetilde{x}_{2k} q_1^{k} + q_0 \sum_{k=0}^{m-1} \widetilde{x}_{2k+1} q_1^{k} \qquad (q_1 = q_0{}^2) \\ &= P_{00}(q_1) + q_0 P_{01}(q_1). \end{aligned} \tag{2.32}$$

Số 0 đầu trong chỉ số 00 và 01 của $P_{00}(q_1)$ và $P_{01}(q_1)$ xác định rằng $P_{00}(q_1)$, $P_{01}(q_1)$ cùng được tách ra từ đa thức gốc $P_0(q_0)$. Các số 0 và 1 sau đó trong chỉ số 00 và 01 được dùng để phân biệt hàm đa thức tách ra là chẵn hay lẻ.

Định lý 2.6: Những giá trị $P_0(q_0^n)$, $n=0,1,\ \dots\ ,N-1$ của đa thức mẹ $P_0(q_0)$ được xác định từ hai đa thức con $P_{00}(q_1)$ và $P_{01}(q_1)$ theo công thức có cấu trúc "hình bướm" (*Butterfly*) như sau:

$$P_0(q_0^n) = P_{00}(q_1^n) + q_0^n P_{01}(q_1^n) \tag{2.33a}$$

$$P_0(q_0^{n+m}) = P_{00}(q_1^n) - q_0^n P_{01}(q_1^n),\ n=0,1,\ \dots\ ,m-1. \tag{2.33b}$$

Chứng minh:

Từ (2.32) có ngay được đẳng thức thứ nhất.

Đẳng thức thứ hai được suy ra từ tính chất $-q_0^n = q_0^{n+m}$ như sau:

$$P_0(q_0^{n+m}) = P_0(-q_0^n) = P_{00}(q_1^n) - q_0^n P_{01}(q_1^n)$$

vì $q_1 = q_0^2$

và đó chính là điều phải chứng minh. ❑

Dễ thấy rằng khi đã có $P_{00}(q_1^n)$ và $P_{01}(q_1^n)$, $n=0,1,\ \dots\ ,m-1$ thì với (2.33) số các phép tính nhân phức phải thực hiện để xác định $P_0(q_0^n)=\widetilde{X}_a(jn\Omega)$, $n=0,1,\ \dots\ ,N-1$ sẽ chỉ bằng một nửa so với khi áp dụng trực tiếp (2.31). Cụ thể ở đây thì số các phép nhân phức phải thực hiện để tính $\widetilde{X}_a(jn\Omega)$ chính là $q_0^n P_{01}(q_1^n)$, $n=0,1,\ \dots\ ,m-1$. Thông qua việc giảm một nửa số các phép tính nhân phức mà tốc độ tính toán được tăng lên gấp đôi.

Tiếp tục, nếu như $m=2s$, hay $N=4s$ thì lại có thể phân tích hai đa thức $P_{00}(q_1)$ và $P_{01}(q_1)$ thành tổng các đa thức bậc thấp hơn, cụ thể là bậc $s-1$, theo đúng nguyên lý đã làm với $P_0(q_0)$ như sau:

$$P_{00}(q_1) = P_{000}(q_2) + q_1 P_{001}(q_2)$$
$$P_{01}(q_1) = P_{010}(q_2) + q_1 P_{011}(q_2), \qquad (q_2 = q_1^{\,2} = q_0^{\,4}).$$

Các giá trị $P_{00}(q_1^n)$, $P_{01}(q_1^n)$, n=0, 1, … ,s–1 cũng sẽ được xác định một cách tương tự nhờ các đa thức con $P_{000}(q_2^n)$, $P_{001}(q_2^n)$, $P_{000}(q_2^n)$ và $P_{001}(q_2^n)$ theo công thức *Butterfly*

$$P_{00}(q_1^n) = P_{000}(q_2^n) + q_1^n P_{001}(q_2^n) \tag{2.34a}$$
$$P_{00}(q_1^{n+s}) = P_{000}(q_2^n) - q_1^n P_{001}(q_2^n) \tag{2.34b}$$

và

$$P_{01}(q_1^n) = P_{010}(q_2^n) + q_1^n P_{011}(q_2^n) \tag{2.34c}$$
$$P_{01}(q_1^{n+s}) = P_{010}(q_2^n) - q_1^n P_{011}(q_2^n). \tag{2.34d}$$

Tổng quát lên cho trường hợp N= $2^l L$, L>1 thì tại bước tách đa thức thứ l từ đa thức mẹ $P_{\ldots b}(q_l)$, trong đó b=0 hoặc 1, sẽ có các đa thức con $P_{\ldots b0}(q_{l+1})$ và $P_{\ldots b1}(q_{l+1})$, $q_{l+1} = q_l^2$ có bậc L–1. Những giá trị $P_{\ldots b}(q_l^n)$, của các đa thức mẹ cũng được xác định từ các giá trị tương ứng của những đa thức con theo công thức *Butterfly*:

$$P_{\ldots b}(q_l^n) = P_{\ldots b0}(q_{l+1}^n) + q_l^n P_{\ldots b1}(q_{l+1}^n), \tag{2.35a}$$

$$P_{\ldots b}(q_l^{n+L}) = P_{\ldots b0}(q_{l+1}^n) - q_l^n P_{\ldots b1}(q_{l+1}^n), \qquad n=0, \ldots, L-1 \tag{2.35b}$$

Khi $N = 2^p$, ta thay L=2 và l=p–1 vào (2.35) và thu được công thức *Butterfly* cho bước tách đa thức thứ p–1. Đây cũng là bước tách đa thức cuối cùng vì các đa thức con $P_{\ldots b0}(q_p)$ và $P_{\ldots b1}(q_p)$ lúc này đều là những đa thức bậc nhất.

Sau khi đã có được những giá trị $P_{\ldots b0}(q_p^n)$, $P_{\ldots b1}(q_p^n)$, n=0, 1 của 2^{p-1} đa thức con tại bước tách đa thức cuối cùng, ta áp dụng (2.35) để tìm những giá trị $P_{\ldots b}(q_{p-1}^n)$ của 2^{p-2} đa thức mẹ $P_{\ldots b}(q_{p-1})$. Tiếp tục, lại sử dụng công thức *Butterfly* để tìm giá trị các đa thức mẹ của các đa thức $P_{\ldots b}(q_{p-1})$. Quá trình trên sẽ được thực hiện ngược lại với quy trình tách đa thức cho tới khi xác định được các giá trị của đa thức gốc $P_0(q_0)$ ứng với bước tách đa thức đầu tiên. Theo (2.31) thì đó chính là N giá trị $\widetilde{X}_a(jn\Omega)$, n=0, 1, … , N–1 cần tìm. Số các phép tính nhân phức phải thực hiện để xác định $\widetilde{X}_a(jn\Omega)$ giảm xuống còn 2pN thay vì $N \cdot \widetilde{N} = N^2$ so với khi tính trực tiếp từ (2.29).

Trên đây là nội dung thuật toán FFT. Thuật toán FFT đã được cài đặt thành những hàm chuẩn trong các chương trình tiện dụng. Ví dụ như trong Toolbox Signalprocessing hay Identification của MatLab. Cách sử dụng chúng cũng rất đơn giản, ta chỉ cần gọi

hàm đó với giá trị đầu vào là dãy $\{\ \widetilde{x}_k\ \}$, k=0,1, … , N–1. Đầu ra của hàm sẽ là dãy kết quả $\{\ \widetilde{X}_a(jn\Omega)\ \}$, n=0, 0,1, … , N–1.

Ví dụ 1: Tìm ảnh Fourier của dãy $\{\ \widetilde{x}_k\ \}=\{1,2,3,4,5,6,7,8\}$ theo công thức (2.29).

Đặt

$$P_0(q_0) = \sum_{k=0}^{7} \widetilde{x}_k q_0^k = 1 + 2q_0 + 3q_0^2 + 4q_0^3 + 5q_0^4 + 6q_0^5 + 7q_0^6 + 8q_0^7,$$

trong đó $q_0 = e^{-j\frac{\pi}{4}} = \frac{\sqrt{2}}{2} - j\frac{\sqrt{2}}{2}$.

Bước 1: **Tách đa thức**

$$P_{00}(q_1) = 1 + 3q_1 + 5q_1^2 + 7q_1^3, \qquad P_{01}(q_1) = 2 + 4q_1 + 6q_1^2 + 8q_1^3$$

với $q_1 = {q_0}^2 = e^{-j\frac{\pi}{2}} = -j$.

Tiếp tục tách $P_{00}(q_1)$ và $P_{01}(q_1)$ thành

$P_{000}(q_2) = 1+5q_2$, $P_{001}(q_2) = 3+7q_2$, $P_{010}(q_2) = 2+6\,q_2$, $P_{011}(q_2) = 4+8\,q_2$,

với $q_2 = {q_1}^2 = {q_0}^4 = e^{-j\pi} = -1$

Bước 2: **Tính giá trị**

$P_{000}({q_2}^0) = 6$, $P_{000}(q_2) = -4$, $P_{001}({q_2}^0) = 10$, $P_{001}(q_2) = -4$,

$P_{010}({q_2}^0) = 8$, $P_{010}(q_2) = -4$, $P_{011}({q_2}^0) = 12$, $P_{011}(q_2) = -4$.

Áp dụng công thức *Butterfly*:

$P_{00}({q_1}^0) = P_{000}({q_2}^0) + {q_1}^0 P_{001}({q_2}^0) = 16$, $P_{00}({q_1}^2) = P_{000}({q_2}^0) - {q_1}^0 P_{001}({q_2}^0) = -4$,

$P_{00}(q_1) = P_{000}(q_2) + q_1 P_{001}(q_2) = -4+4j$, $P_{00}({q_1}^3) = P_{000}(q_2) - q_1 P_{001}(q_2) = -4-4j$,

$P_{01}({q_1}^0) = P_{010}({q_2}^0) + q_1{}^0 P_{011}({q_2}^0) = 20$, $P_{01}({q_1}^2) = P_{010}({q_2}^0) - {q_1}^0 P_{011}({q_2}^0) = -4$,

$P_{01}(q_1) = P_{010}(q_2) + q_1 P_{011}(q_2) = -4+4j$, $P_{01}({q_1}^3) = P_{010}(q_2) - q_1 P_{011}(q_2) = -4-4j$.

Áp dụng tiếp công thức *Butterfly* lần nữa ta sẽ thu được kết quả

$\widetilde{X}_a(0) = P_0({q_0}^0) = P_{00}({q_1}^0) + q_0^0 P_{01}({q_1}^0) = 36$

$\widetilde{X}_a(4\Omega) = P_0({q_0}^4) = P_{00}({q_1}^0) - q_0^0 P_{01}({q_1}^0) = -4$

$\widetilde{X}_a(\Omega) = P_0(q_0) = P_{00}(q_1) + q_0 P_{01}(q_1) = -4+(4+4\sqrt{2})j$

$\widetilde{X}_a(5\Omega) = P_0({q_0}^5) = P_{00}(q_1) - q_0 P_{01}(q_1) = -4+(4-4\sqrt{2})j$

$\widetilde{X}_a(2\Omega) = P_0({q_0}^2) = P_{00}({q_1}^2) + q_0^2 P_{01}({q_1}^2) = -4+4j$

$\widetilde{X}_a(6\Omega) = P_0({q_0}^6) = P_{00}({q_1}^2) - q_0^2 P_{01}({q_1}^2) = -4-4j$

$\widetilde{X}_a(3\Omega) = P_0(q_0^3) = P_{00}(q_1^3) + q_0^3 P_{01}(q_1^3) = -4 + (4\sqrt{2} - 4)\,j$

$\widetilde{X}_a(7\Omega) = P_0(q_0^7) = P_{00}(q_1^3) - q_0^3 P_{01}(q_1^3) = -4 - (4 + 4\sqrt{2})j.$ ❑

Dưới đây là chương trình con **fft()** viết bằng ngôn ngữ C mô tả việc cài đặt thuật toán trên để tham khảo. Chương trình con **fft()** có các biến hình thức là

a) **Nexp** chứa số mũ lũy thừa 2 của N, với N là độ dài của dãy tín hiệu vào $\{\widetilde{x}_k\}$, $k = 0, 1, \ldots, N-1$, tức là **Nexp** $= \log_2 N$.

b) Con trỏ **x** chỉ vào đầu mảng **x[]** chứa dãy giá trị đầu vào $\{\widetilde{x}_k\}$, $k = 0, 1, \ldots, N-1$, theo thứ tự **x[0]** $= \widetilde{x}_0, \ldots,$ **x[N-1]** $= \widetilde{x}_{N-1}$. Như vậy mảng **x[]** phải có các phần tử phức và có độ dài ít nhất là $2^{\mathbf{Nexp}}$.

Các giá trị $\widetilde{X}_a(jn\Omega)$, $n = 0, 1, \ldots, N-1$ tính được sẽ được chương trình con **fft()** ghi lại vào mảng **x[]** cũng theo thứ tự **x[0]** $= \widetilde{X}_a(0), \ldots,$ **x[N-1]** $= \widetilde{X}_a(j(N-1)\Omega)$. Như vậy sau khi thực hiện xong dãy $\{\widetilde{x}_k\}$ sẽ bị xóa và thay vào đó là kết quả $\{\widetilde{X}_a(jn\Omega)\}$. Mảng **x[]** của hàm **fft()** vừa chứa các giá trị đầu vào $\{\widetilde{x}_k\}$, vừa chứa các giá trị đầu ra $\{\widetilde{X}_a(jn\Omega)\}$ của thuật toán nên nó phải là mảng các số phức.

```
void fft(int Nexp,complex *x)
{
   int N=pow(2.,Nexp),N1=Nexp,N2,k,l,p,m,i;
   complex s,wp;
   N2=N;
   for (l=0;l<Nexp;l++)
   {
      N1--;
      N2=N2/2;
      for (k=0;k<N;k+=N2)
      {
         m=k/pow(2.,(double)N1);
         p=ibr(m,Nexp);
         wp=exp(complex(0.,-M_PI*2*p/N));
         for (i=0;i<N2;i++)
         {  s=x[k+N2]*wp;
            x[k+N2]=x[k]-s;
            x[k++]=x[k]+s;
         }
      }
   }
   for (k=0;k<N;k++)
   {
```

```
        i=ibr(k,Nexp);
        if (i>k){ s=x[k]; x[k]=x[i]; x[i]=s;}
    }
}
```

Chương trình **fft()** trên sử dụng một hàm con **ibr()** có nhiệm vụ tách đa thức và sau đó sắp xếp lại mảng giá trị $\{ \widetilde{X}_a(jn\Omega) \}$ tính được. Hàm con **ibr()** có hai biến vào:

a) **index** chứa chỉ số cũ của giá trị $\widetilde{x}_k$ là k trong mảng **x[]**.

b) **Nexp** chứa số mũ lũy thừa 2 của N, tức là **Nexp** $=\log_2 N$.

Hàm **ibr()** trả về chỉ số mới phải có của giá trị $\widetilde{x}_k$ trong mảng **x[]**.

```
int ibr(int index, int Nexp)
{  int k=0,i,j=index;
   for (i=0;i<Nexp;i++)
   {
      k=2*k+(j%2);
      j=j/2;
   }
   return (k);
}
```

Ví dụ 2: Ta sẽ thực hiện lại ví dụ 1, nhưng lần này sử dụng hàm **fft()**. Trước tiên viết chương trình tạo dãy đầu vào gồm 8 giá trị $\{ \widetilde{x}_k \}=\{1,2,3,4,5,6,7,8\}$ nhưng phải là các giá trị phức cho mảng **x[]** và sau đó gọi hàm **fft()**:

```
void main()
{
   complex *x;
   int i,Nexp=3,N=8;
   x=new complex [N];
   for (i=0;i<N;i++) x[i]=complex(double (i+1));
   fft(Nexp,x);
   for (i=0;i<N;i++)
      printf("\nx[%d]=(%f,%f)",i,real(x[i]),imag(x[i]));
   delete [] x;
   getch();
}
```

Chương trình sẽ cho ra kết quả:

```
x[0]=(36.000000,0.000000)
x[1]=(-4.000000,9.656854)
x[2]=(-4.000000,4.000000)
x[3]=(-4.000000,1.656854)
x[4]=(-4.000000,0.000000)
```

```
x[5]=(-4.000000,-1.656854)
x[6]=(-4.000000,-4.000000)
x[7]=(-4.000000,-9.656854) .
```
❑

Kết hợp **fft()** cùng với kỹ thuật hàm cửa số nhằm làm giảm sai số rò rỉ và quan hệ (2.17b) ta sẽ có được hàm **dft()** viết trên C cho sau đây để phục vụ việc tính ảnh Fourier $X(jn\Omega_\lambda)$ của một tín hiệu $x(t)$ từ dãy các giá trị tín hiệu $x_k = x(kT_a)$, k = 0, 1, … , N–1 của nó. Hàm **dft()** có các biến hình thức sau:

a) Con trỏ **x** chỉ đầu mảng số thực **x[]** chứa các giá trị tín hiệu đo được $x_k = x(kT_a)$, với k = 0, 1, … , N–1.

b) **Ta** chứa hằng số thời gian trích mẫu tín hiệu T_a .

c) **N** là số các giá trị tín hiệu $x_k = x(kT_a)$ đo được.

d) **w** là chỉ số hàm cửa sổ được sử dụng để làm giảm sai số rò rỉ. Đây chính là chỉ số i của $w_i(t)$ cho trong mục 2.1.6, bởi vậy i chỉ có nghĩa khi nằm trong khoảng $0 \le i \le 7$. Nếu giá trị của **w** nằm ngoài khoảng [0,7] hàm **dft()** sẽ sử dụng hàm cửa sổ xung vuông $w_0(t)$.

e) Con trỏ **X** chỉ đầu mảng số phức **X[]** chứa các giá trị ảnh Fourier $X(jn\Omega_\lambda)$, n=0,1, … , λ tính được với $\Omega_\lambda = \dfrac{2\pi}{\lambda T_a}$ trong đó λ phải là số nguyên dạng lũy thừa của 2 không nhỏ hơn N để phù hợp với chương trình con **fft()**. Hàm **dft()** tính ảnh Fourier của dãy $\{x_k\}$ gồm λ phần tử và các phần tử x_k có chỉ số k từ N đến λ–1 được gán bằng 0 sau đó cất kết quả vào mảng **X[]** có độ dài ít nhất là λ.

Hàm trả về giá trị λ là độ dài của mảng **X[]** chứa kết quả $X(jn\Omega_\lambda)$. Nội dung của mảng đầu vào **x[]** không bị thay đổi.

```
int dft(double *x, double Ta, int N, int w, complex *X)
{
   int i,k,p,Nexp=0;
   double T=Ta*N,s,t;
   k=N/2;
   for (i=0;i<=k;i++)
   {
      t=Ta*(double)(i-k);
      s=fw(w,t,T);
      X[i]=complex(x[i]*s*Ta);
      if((i>0)&&(i!=(p=(N-i)))) X[p]=complex(x[p]*s*Ta);
   }
   while ((k=pow(2.,Nexp))<N) Nexp++;
```

```
    for (i=N;i<k;i++) X[i]=complex(0.);
    fft(Nexp,X);
    return(k);
}
```

Hàm **dft()** trên sử dụng thêm hàm con **fw()** có nhiệm vụ phụ giúp việc tạo dãy $\{\widetilde{x}_k\}$ từ dãy giá trị các tín hiệu $\{x_k\}$ đã cho trong mảng **x[]** theo công thức

$$\widetilde{x}_k = x_k w_i(kT_a)$$

bằng cách xác định $w_i(t+\frac{T}{2})$ khi đã biết chỉ số i, thời gian t và khoảng thời gian sống T của tín hiệu. Hàm có các biến hình thức

a) **w** chứa chỉ số hàm cửa sổ được sử dụng. Nội dung của **w** là một số trong khoảng [0,7]. Nếu giá trị của **w** lớn hơn 7, hàm **fw()** sẽ sử dụng hàm cửa sổ $w_0(t)$.

b) **t** là giá trị thời gian.

c) **T** chứa giá trị T xác định supp$w_i(t)$=(0, T) –xem thêm hàm cửa sổ ở mục 2.1.7.

Hàm **fw()** trả về giá trị $s = w_i(t+\frac{T}{2})$ tính được.

```
double fw(int w, double t, double T)
{
   double s;
   switch(w)
   {
      case 1:   s=cos(t*M_PI/T); break;
      case 2:   s=1.-2.*fabs(t)/T; break;
      case 3:   s=0.5+0.5*cos(t*2*M_PI/T); break;
      case 4:   s=0.54+0.46*cos(t*2*M_PI/T); break;
      case 5:   s=0.42+0.5*cos(t*2*M_PI/T)+0.08*cos(t*M_PI*4/T);
                break;
      case 6:   s=fabs(sin(t*M_PI*2/T)/M_PI)+1.-
                   2.*fabs(t/T)*cos(t*M_PI*2/T);
                break;
      case 7:   if(fabs(t)<(T/4)) s=1.-24.*t*t*(1.-
                   2.*fabs(t/T)/(T*T));
                else s=2.*pow(1.-2.*fabs(t/T),3.); break;
      default:  s=1.;
   }
   return(s);
}
```

2.1.8 Toán tử DFT ngược

Mục 2.1.7 đã trình bày thuật toán FFT được dùng để tính nhanh các giá trị $\{ \widetilde{X}_a(jn\Omega) \}$ từ dãy $\{ \widetilde{x}_k \}$, $k=0,1, \ \dots \ , N-1$ theo công thức

$$\widetilde{X}_a(jn\Omega) = \sum_{k=0}^{N-1} \widetilde{x}_k e^{-j\frac{2\pi}{N}nk} \ , n=0,1, \ \dots \ , N-1,$$

trong đó $\widetilde{x}_k$ là mẫu tín hiệu $x(t)$ tại thời điểm $t= kT_a$ đã được nhân với giá trị hàm cửa sổ $w(kT_a)$ cũng tại thời điểm đó theo (2.23) nhằm làm giảm sai số rò rỉ:

$$\widetilde{x}_k = x_k w(kT_a)$$

Kết hợp với thuật toán FFT mà cụ thể là hàm **`fft()`**, hàm **`dft()`** cho trong mục 2.1.7 có nhiệm vụ tính

$$X(jn\Omega) \approx T_a \widetilde{X}_a(jn\Omega)$$

và nó được xem như là công thức tính gần đúng của toán tử Fourier

$$X(j\omega) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt \tag{2.36}$$

Bài toán ngược được xét ở đây là tìm $x(t)$ từ các giá trị phổ đã có của nó $X(jn\Omega)$, $n=0,1, \ \dots \ , N-1$, trong đó $X(jn\Omega)$ là giá trị của $X(j\omega)$ tại điểm tần số $\omega=n\Omega=\frac{2\pi n}{NT_a}$, tức là xác định

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\omega) e^{j\omega t} d\omega \tag{2.37}$$

Thay $\omega'=-\omega$ vào (2.37)

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(-j\omega') e^{-j\omega' t} d\omega' \tag{2.38}$$

rồi so sánh với (2.36) thì thấy ngoài hằng số $\frac{1}{2\pi}$ chúng có cấu trúc hoàn toàn tương tự. Cụ thể là vai trò của $x(t)$ trong (2.36) thì nay trong (2.38) được thay bởi

$$X(-j\omega') = \overline{X(j\omega')}$$

còn $X(j\omega)$ trong (2.36) thì lại được thay bằng $x(t)$. Do đó, nếu lý luận như đã làm với DFT ta sẽ đi đến công thức tính xấp xỉ cho $x(t)$ như sau

$$x(kT_a) \approx \frac{\Omega}{2\pi} \sum_{n=0}^{N-1} w(n\Omega) \overline{X(jn\Omega)} e^{-j\frac{2\pi}{N}kn} = \frac{1}{NT_a} \sum_{n=0}^{N-1} \widetilde{X}_n e^{-j\frac{2\pi}{N}kn} \tag{2.39}$$

trong đó $w(\omega)$ là hàm cửa sổ trong miền phức nhằm làm giảm sai số rò rỉ của phép tính Fourier ngược và $\widetilde{X}_n = w(n\Omega)\overline{X(jn\Omega)}$.

Không kể hằng số $\frac{1}{NT_a}$ thì (2.39) chính là công thức của DFT (2.29) với dãy các giá trị đầu vào:

$$\{\, w(n\Omega)\overline{X(jn\Omega)}\,\}, \quad n=0,1,\ldots,\ N-1.$$

Nói cách khác, thuật toán FFT hay hàm **`fft()`** không những có tác dụng để tính $\{\, \widetilde{X}_a(jn\Omega)\,\}$ từ dãy $\{\, \tilde{x}_k\,\}$ mà còn có thể được dùng để xác định ngược $\tilde{x}_k$ từ dãy các giá trị liên hợp

$$\widetilde{X}_n = w(n\Omega)\overline{X(jn\Omega)}$$

của ảnh Fourier của nó.

Áp dụng kết quả thu được ở trên, ta đi đến thuật toán sử dụng FFT cho việc tính ngược giá trị $x(kT_a)$ của tín hiệu $x(t)$ như sau:

1) Tạo dãy $\{\, w(n\Omega)\overline{X(jn\Omega)}\,\}$, n=0,1, … , N–1 từ các giá trị phổ $X(jn\Omega)$ của tín hiệu.
2) Gọi hàm **`fft()`** với đầu vào vừa tạo. Kết quả đầu ra sẽ là dãy N–1 các giá trị $\tilde{x}_k$, k=0,1, … , N–1.
3) Tính $x(kT_a) \approx \frac{\tilde{x}_k}{NT_a}$, k=0,1, … , N–1 với $T_a = \frac{2\pi}{N\Omega}$

Thuật toán trên đã được cài đặt thành chương trình con **`invdft()`** viết trên C cho dưới đây để tham khảo. Chương trình con này có các biến hình thức sau:

a) **`Ta`** là chu kỳ lấy mẫu tín hiệu. Tức là dãy giá trị đầu vào $\{X(jn\Omega)\}$, $n=0,1,\ \ldots\ ,$ N–1, sẽ có $\Omega = \frac{2\pi}{NT_a}$.

b) **`Nexp`** chứa số mũ lũy thừa 2 của N. Như vậy dãy tín hiệu vào $\{X(jn\Omega)\}$ của chương trình sẽ có độ dài là $N=2^{\texttt{Nexp}}$.

c) **`w`** xác định hàm cửa sổ được sử dụng nhằm giảm sai số rò rỉ của kỹ thuật DFT. Nội dung của **`w`** là chỉ số $0 \le i \le 7$ của $w_i(t)$ cho trong mục 2.1.6. Nếu giá trị của **`w`** nằm ngoài khoảng [0,7] chương trình sẽ sử dụng hàm cửa sổ xung vuông $w_0(t)$.

d) Con trỏ **`x`** chỉ đầu mảng số phức **`x[]`** chứa dãy giá trị đầu vào $\{X(jn\Omega)\}$, $n=0,1,$ … , N–1, theo thứ tự **`x[0]`**=$X(0)$, …, **`x[N-1]`**= $X(j(N-1)\Omega)$. Mảng **`x[]`** phải có độ dài ít nhất là $2^{\texttt{Nexp}}$.

Dãy kết quả $\{x(kT_a)\}$, k=0,1, … , N–1 sẽ được chương trình **invdft()** ghi lại vào mảng **x[]**. Nói cách khác, sau khi thực hiện chương trình thực hiện xong, dãy $\{X(jn\Omega)\}$ sẽ bị xóa và thay vào đó là dãy kết quả $\{x(kT_a)\}$.

```
void invdft(double Ta, int Nexp, int w, complex *x)
{
    int i,k,p,N=pow(2,Nexp);
    double T=Ta*N,s,t;
    k=N/2;
    for (i=0;i<=k;i++)
    {
        t=Ta*(double)(i-k);
        s=fw(w,t,T);
        x[i]=conj(x[i])*s/T;
        if((i>0)&&(i!=(p=(N-i)))) x[p]=conj(x[p])*s/T;
    }
    fft(Nexp,x);
}
```

2.2 Nhận dạng mật độ phổ tín hiệu

Cho hai quá trình ngẫu nhiên $\boldsymbol{u}(t)$ và $\boldsymbol{y}(t)$. Như đã quy ước ở chương 1, nếu không có chỉ dẫn gì thêm, hai quá trình ngẫu nhiên này sẽ được hiểu là những quá trình ngẫu nhiên egodic. Quá trình $\boldsymbol{u}(t)$ được xem như là tập hợp của tất cả các tín hiệu đầu vào $u(t)$ có thể có của đối tượng và cũng như vậy $\boldsymbol{y}(t)$ được xem là tập hợp tất cả các tín hiệu đầu ra $y(t)$ có thể có từ đối tượng.

Nhận dạng mật độ phổ $S_u(\omega)$, $S_{uy}(j\omega)$ tức là phải xác định ảnh Fourier của hàm tương quan $r_u(\tau)$, $r_{uy}(\tau)$ theo công thức Wiener–Chitchin:

$$S_u(\omega) = \int_{-\infty}^{\infty} r_u(t)e^{-j\omega t}dt \qquad \text{và} \qquad S_{uy}(j\omega) = \int_{-\infty}^{\infty} r_{uy}(t)e^{-j\omega t}dt \tag{2.40}$$

trong đó $S_u(\omega)$ thuần thực vì $r_u(\tau)$ là hàm chẵn, bởi vậy nó được viết đơn giản là $S_u(\omega)$ thay cho $S_u(j\omega)$.

Với định nghĩa trên thì để có $S_u(\omega)$, $S_{uy}(j\omega)$ trước hết ta phải tính được $r_u(\tau)$, $r_{uy}(\tau)$ trên cơ sở quan sát các tín hiệu vào ra $u(t)$, $y(t)$. Nói cách khác việc nhận dạng $S_u(\omega)$, $S_{uy}(j\omega)$ theo nguyên tắc phải gồm hai bước:

1) Xác định $r_u(\tau)$, $r_{uy}(\tau)$ từ dãy giá trị tín hiệu $\{u_k\}$, $\{u_k\}$, k=0,1, … , N–1.

2) Tính $S_u(\omega)$, $S_{uy}(j\omega)$ từ $r_u(\tau)$, $r_{uy}(\tau)$ theo công thức Wiener–Chitchin (2.40) nhờ kỹ thuật DFT, cụ thể là hàm **dft()** đã trình bày ở mục 2.1.

Vấn đề đặt ra ở đây cũng giống như trong mục trước là việc quan sát tín hiệu chỉ có thể được thực hiện trong khoảng $[0,T)$, nên cũng chỉ có thể cung cấp được dãy $\{u_k\}$, $\{u_k\}$, $k=0,1,\ \ldots\ ,N-1$ hữu hạn các giá trị tín hiệu $u(kT_a)$, $y(kT_a)$, trong đó N là số nguyên nhỏ nhất không nhỏ hơn $\frac{T}{T_a}$. Điều này dẫn tới $S_u(\omega)$, $S_{uy}(j\omega)$ nhận được không phải mật độ phổ chính xác của tín hiệu mà chỉ là những giá trị gần đúng.

Sau đây, trong phần này ta sẽ làm quen với những phương pháp xác định xấp xỉ $S_u(\omega)$, $S_{uy}(j\omega)$ thông qua việc quan sát tín hiệu trong khoảng thời gian hữu hạn $[0,T)$ cũng như kỹ thuật làm tăng độ chính xác cho kết quả.

2.2.1 Nhận dạng hàm tương quan

Từ nay về sau, khái niệm hàm tương quan sẽ được hiểu chung là hàm tự tương quan $r_u(\tau)$ và hàm hỗ tương quan $r_{uy}(\tau)$.

Do khoảng thời gian quan sát $[0,T)$ các tín hiệu vào ra $u(t)$, $y(t)$ là hữu hạn nên để tường minh ta sẽ thay $u(t)$, $y(t)$ bởi $\tilde{u}(t)$, $\tilde{y}(t)$ bằng cách nhân $u(t)$, $y(t)$ với hàm cửa sổ xung vuông $w_0(t)$ như sau

$$\tilde{u}(t)=u(t)w_0(t)=\begin{cases}u(t) & \text{khi}\quad 0\le t<T\\ 0 & \text{khi}\quad t\notin[0,T)\end{cases}$$

$$\tilde{y}(t)=y(t)w_0(t)=\begin{cases}y(t) & \text{khi}\quad 0\le t<T\\ 0 & \text{khi}\quad t\notin[0,T)\end{cases}$$

Khi đó hai công thức tính hàm tương quan $r_u(\tau)$, $r_{uy}(\tau)$ cho trong (1.30), (1.33) được sửa lại thành

$$\tilde{r}_u(\tau)=\frac{1}{T}\int_0^T\tilde{u}(t)\tilde{u}(t+\tau)dt \tag{2.41a}$$

$$\tilde{r}_{uy}(\tau)=\frac{1}{T}\int_0^T\tilde{u}(t)\tilde{y}(t+\tau)dt\ , \tag{2.41b}$$

và tất nhiên $\tilde{u}(t)$, $\tilde{y}(t)$ cũng là những phần tử bất kỳ của tập hợp $\boldsymbol{u}(t)$, $\boldsymbol{y}(t)$. Lý do việc sửa hệ số $\frac{1}{2T}$ trong (1.30), (1.33) thành $\frac{1}{T}$ nằm ở bản chất khái niệm *giá trị trung bình mối tương quan* của $r_u(\tau)$ và $r_{uy}(\tau)$, tức là phải chia giá trị tích phân (tổng) cho đúng khoảng thời gian đã quan sát tín hiệu (hay số các số hạng nếu dấu tích phân được thay bằng dấu tổng).

Trước hết ta chứng minh định lý sau:

Định lý 2.7: Hàm tương quan $\tilde{r}_u(\tau)$ và $\tilde{r}_{uy}(\tau)$ xác định theo (2.41) sẽ đồng nhất bằng 0 khi τ nằm ngoài khoảng $(-T, T)$.

Chứng minh: Khi $\tau \notin (-T, T)$ ta luôn có $\tilde{u}(t+\tau)=0$ và $\tilde{y}(t+\tau)=0$ với mọi $t\in[0,T)$. Do đó cũng có

$$\tilde{u}(t)\tilde{u}(t+\tau)=\tilde{u}(t)\tilde{y}(t+\tau)=0, \qquad \forall t\in[0,T),\ \tau\notin(-T,T)$$

và đó chính là điều phải chứng minh. ❑

Vấn đề đặt ra ở đây là kết quả quan sát tín hiệu $u(t)$, $y(t)$ trong khoảng thời gian $t\in[0,T)$ không phải là $\tilde{u}(t), \tilde{y}(t)$ mà chỉ là những giá trị $\tilde{u}_k=\tilde{u}(kT_a)$, $\tilde{y}_k=\tilde{y}(kT_a)$, k=0,1, … , N–1 của chúng, trong đó T_a là khoảng thời gian trích mẫu, N là số nguyên nhỏ nhất không nhỏ hơn $\dfrac{T}{T_a}$. Nếu áp dụng kỹ thuật hàm mở rộng $\delta(t)$ để mô hình hóa quá trình trích mẫu này, ta sẽ có được công thức tính gần đúng $\tilde{r}_u(\tau)$ và $\tilde{r}_{uy}(\tau)$ như sau:

$$\tilde{r}_u(mT_a)\approx \frac{1}{N}\sum_{k=0}^{N-1}\tilde{u}_k\tilde{u}_{k+m} \tag{2.42a}$$

$$\tilde{r}_{uy}(mT_a)\approx \frac{1}{N}\sum_{k=0}^{N-1}\tilde{u}_k\tilde{y}_{k+m} \tag{2.42b}$$

bằng cách thay $\{\tilde{u}_k\}$, $\{\tilde{y}_k\}$ vào (2.41), đổi dấu $\int$ thành $\boldsymbol{\Sigma}$, thay dt bằng T_a.

Do $\{\tilde{u}_k\}$, $\{\tilde{y}_k\}$ là những dãy hữu hạn với k=0,1, … , N–1, tức là khi $k\le 0$ hoặc $k\ge N$ có $\tilde{u}_k=\tilde{y}_k=0$, nên ngoài khoảng $0\le k< N-|m|$ thì $\tilde{u}_{k+m}=\tilde{y}_{k+m}=0$. Bởi vậy hai công thức (2.42a) và (2.42b) có thể sửa lại được thành:

$$\tilde{r}_u(mT_a)\approx \frac{1}{N}\sum_{k=0}^{N-|m|-1}\tilde{u}_k\tilde{u}_{k+m} \tag{2.43a}$$

$$\tilde{r}_{uy}(mT_a)\approx \frac{1}{N}\sum_{k=0}^{N-|m|-1}\tilde{u}_k\tilde{y}_{k+m} \tag{2.43b}$$

Mặt khác, muốn tính giá trị trung bình $\tilde{r}_u(mT_a), \tilde{r}_{uy}(mT_a)$ một cách chính xác, ta chia tổng có được cho đúng số các số hạng có trong tổng và đi đến dạng khác của (2.43) như sau

$$\tilde{r}_u(mT_a)\approx \frac{1}{N-|m|}\sum_{k=0}^{N-|m|-1}\tilde{u}_k\tilde{u}_{k+m} \tag{2.44a}$$

$$\tilde{r}_{uy}(mT_a)\approx \frac{1}{N-|m|}\sum_{k=0}^{N-|m|-1}\tilde{u}_k\tilde{y}_{k+m} \tag{2.44b}$$

Cả hai công thức (2.43), (2.44) đều được sử dụng để tính gần đúng các giá trị $\tilde{r}_u(mT_a)$, $\tilde{r}_{uy}(mT_a)$ của hàm tương quan, trong đó (2.44) được gọi là công thức *unbias* (tức là sẽ không còn sai lệch tĩnh khi $N \to \infty$) và (2.42) là công thức *bias* (vẫn còn sai lệch ngay cả khi $N \to \infty$).

Theo định lý 2.7, thì với $|m| \geq N$ có

$$\tilde{r}_u(mT_a) = \tilde{r}_{uy}(mT_a) = 0,$$

do đó trong công thức tính xấp xỉ (2.42), (2.44) chỉ số m chỉ cần chạy từ $-N+1$ đến $N-1$ là đủ.

Ngoài ra, kết hợp thêm với tính chất (1.31), (1.34) về hàm tương quan, ta cũng không cần phải tính $\tilde{r}_u(mT_a)$, $\tilde{r}_{uy}(mT_a)$ cho tất cả chỉ số m từ $-N+1$ đến $N-1$, mà chỉ cần tính cho m từ 0 đến $N-1$, vì khi $m<0$ giá trị của $\tilde{r}_u(mT_a)$, $\tilde{r}_{uy}(mT_a)$ sẽ suy được ra từ các giá trị đối xứng $\tilde{r}_u(-mT_a)$, $\tilde{r}_{yu}(-mT_a)$:

$$\tilde{r}_u(mT_a) = \tilde{r}_u(-mT_a), \qquad \tilde{r}_{uy}(mT_a) = \tilde{r}_{yu}(-mT_a).$$

Tóm lại, có hai cách tính gần đúng hàm tương quan như sau:

1) Công thức *bias*

$$\tilde{r}_u(mT_a) \approx \begin{cases} \dfrac{1}{N}\displaystyle\sum_{k=0}^{N-m-1} \tilde{u}_k \tilde{u}_{k+m} & \text{với} \quad m = 0,1,\ldots,N-1 \\ \tilde{r}_u(-mT_a) & \text{với} \quad m = -N+1,\ldots,-1 \end{cases} \tag{2.45}$$

$$\tilde{r}_{uy}(mT_a) \approx \begin{cases} \dfrac{1}{N}\displaystyle\sum_{k=0}^{N-m-1} \tilde{u}_k \tilde{y}_{k+m} & \text{với} \quad m = 0,1,\ldots,N-1 \\ \tilde{r}_{yu}(-mT_a) & \text{với} \quad m = -N+1,\ldots,-1 \end{cases} \tag{2.46}$$

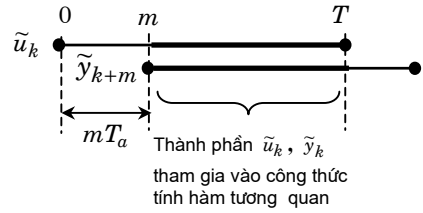
2) Công thức *unbias*

$$\tilde{r}_u(mT_a) \approx \begin{cases} \dfrac{1}{N-m}\displaystyle\sum_{k=0}^{N-m-1} \tilde{u}_k \tilde{u}_{k+m} & \text{với} \quad m = 0,1,\ldots,N-1 \\ \tilde{r}_u(-mT_a) & \text{với} \quad m = -N+1,\ldots,-1 \end{cases} \tag{2.47}$$

$$\tilde{r}_{uy}(mT_a) \approx \begin{cases} \dfrac{1}{N-m}\displaystyle\sum_{k=0}^{N-m-1} \tilde{u}_k \tilde{y}_{k+m} & \text{với} \quad m = 0,1,\ldots,N-1 \\ \tilde{r}_{yu}(-mT_a) & \text{với} \quad m = -N+1,\ldots,-1 \end{cases} \tag{2.48}$$

Một điều cần chú ý thêm là mặc dù những công thức trên cho phép tính được gần đúng $2N-1$ giá trị $\tilde{r}_u(mT_a)$, $\tilde{r}_{uy}(mT_a)$, $m=-N+1,\ldots,-1,0,1,\ \ldots\ ,N-1$ của hàm tương quan từ N giá trị đo được $\tilde{u}_k$, $k=0,1,\ \ldots\ ,N-1$ của tín hiệu vào $u(t)$ và N giá trị $\tilde{y}_k$, $k=0,1,\ \ldots\ ,N-1$ đo được của tín hiệu ra $y(t)$, nhưng vì thời gian đo là hữu hạn nên với $|m|$ càng lớn,

sẽ càng có ít giá trị $\tilde{u}_k$, $\tilde{y}_k$ tham gia vào công thức (2.45)÷(2.48), (hình 2.10), do đó sai số sẽ càng lớn. Bởi vậy, trong số 2*N*–1 giá trị $\tilde{r}_u(mT_a)$, $\tilde{r}_{uy}(mT_a)$ tính được chỉ có một số ít các giá trị xung quanh điểm 0 là tương đối chính xác.



Hình 2.10: Mô tả sai số phụ thuộc vào ***m*** của các giá trị hàm tương quan $\tilde{r}_{uy}(mT_a)$.

Vậy lấy bao nhiêu thì đủ?. Để trả lời câu hỏi này, người ta đã lập bài toán thống kê và đi đến kết luận trong [10], [12] rằng con số đó là khoảng 5÷20% của 2*N*–1, tức là chỉ lấy thực sự các giá trị

$$\tilde{r}_u(mT_a), \tilde{r}_{uy}(mT_a) \quad \text{có} \quad |m| \le M \quad \text{với} \quad \frac{(2N-1)}{20} \le M \le \frac{(2N-1)}{5},$$

trong đó *M* được gọi là chỉ số Lag (cắt bớt).

Một hàm **`cor()`** viết trên ngôn ngữ lập trình C thực hiện việc cài đặt thuật toán nhận dạng hàm hỗ tương quan $\tilde{r}_{uy}(mT_a)$ theo các công thức (2.45)÷(2.48) cho dưới đây để tham khảo.

Hàm **`cor()`** có 6 biến hình thức, bao gồm:

a) Con trỏ **`u`** chỉ đầu mảng số thực **`u[]`** chứa các giá trị $\tilde{u}_k$, k=0,1, … , N–1.

b) Con trỏ **`y`** chỉ đầu mảng số thực **`y[]`** chứa các giá trị $\tilde{y}_k$, k=0,1, … , N–1.

c) Số nguyên **`N`** chứa độ dài hai dãy { $\tilde{u}_k$ }, { $\tilde{y}_k$ }, tức là chứa chỉ số N.

d) Số nguyên **`M`** chứa chỉ số Lag, tức là độ dài cần phải có của dãy kết quả **`r[]`**.

e) Số nguyên **`bias`** xác định giá trị hàm tương quan sẽ được tính theo công thức bias (**`bias`**=1) hay unbias (**`bias`**≠1).

f) Con trỏ **`r`** chỉ đầu mảng số thực **`r[]`** chứa các giá trị $\tilde{r}_{uy}(mT_a)$, m = 0,1, … , M tính được theo thứ tự **`r[0]`**=$\tilde{r}_{uy}(0)$, **`r[1]`**=$\tilde{r}_{uy}(T_a)$, … , **`r[M]`**=$\tilde{r}_{uy}(MT_a)$.

Hàm trả về giá trị báo lỗi bằng 0 nếu $M<N$ hoặc bằng 1 trong trường hợp ngược lại. Hàm không làm thay đổi nội dung các mảng **`u[]`**, **`y[]`**.

```
int cor(double *u, double *y, int N, int M, int bias, double *r)
{
   int m,k,err=0;
   if (M<N)
      for (m=0;m<=M;m++)
      {
         r[m]=0.;
         for(k=0;k<N-m;k++) r[m]=r[m]+u[k]*y[k+m];
         if  (bias==1) r[m]=r[m]/N;
         else r[m]=r[m]/(N-m);
      }
   else err=1;
   return (err);
}
```

Muốn tính giá trị hàm tự tương quan $\widetilde{r}_u(mT_a)$, m=0,1, … , M ta gọi hàm trên với hai dãy đầu vào **u[k]**, **y[k]** chứa cùng một giá trị là **u[k]**= **y[k]**=$\widetilde{u}_k$, k=0,1, … , N–1. Cũng như vậy, để tính những giá trị còn lại $\widetilde{r}_{uy}(mT_a)$, m=–M,…,–1,0 của hàm hỗ tương quan ta gọi hàm **cor()** với các giá trị đầu vào **u[k]**=$\widetilde{y}_k$, **y[k]**=$\widetilde{u}_k$, k=0,1, … , N–1. Kết quả sẽ là **r[0]**=$\widetilde{r}_{uy}(0)$, **r[1]**=$\widetilde{r}_{uy}(-T_a)$, … , **r[M]**=$\widetilde{r}_{uy}(-MT_a)$.

Ví dụ: Viết chương trình nhập dữ liệu và gọi hàm **cor()**

```
void main()
{
   double *u,*y, *r;
   int N,M,i,k,bias;
   printf ("N= "); scanf ("%d",&N);
   printf ("M= "); scanf ("%d",&M);
   printf ("bias= "); scanf ("%d",&bias);
   u=new double [N+1]; y=new double [N+1]; r=new double [M+1];
   for (i=0;i<N;i++){printf ("u[%d]=",i); scanf ("%lf",u+i);}
   for (i=0;i<N;i++){printf ("y[%d]=",i); scanf ("%lf",y+i);}
   printf("err=%d\n",i=cor(u,y,N,M,bias,r));
   if (i==0) for (k=0;k<=M;k++) printf("r[%d]=\t%1.4f\n",k,r[k]);
   delete [] u;
   delete [] y;
   delete [] r;
   getch();
}
```

sau đó thực hiện với các giá trị sau cho từ bàn phím

N=4, **M**=2, **bias**=1, **u[0]**=**y[0]**=1, **u[1]**=**y[1]**=2, **u[2]**=**y[2]**=3, **u[3]**=**y[3]**=4,

ta sẽ nhận được

err=0, **r[0]**=7,5; **r[1]**=5,0 và **r[2]**=2,75. ❑

2.2.2 Nhận dạng mật độ phổ

Khái niệm nhận dạng gián tiếp mật độ phổ được hiểu là xác định các giá trị mật độ phổ thông qua việc nhận dạng hàm tương quan, tức là nhận dạng hàm tương quan trước rồi sau đó sử dụng **`dft()`** để tính giá trị mật độ phổ.

Để xác định mật độ $S_u(\omega)$, $S_{uy}(j\omega)$ ta đi từ công thức Wiener–Chitchin (2.40) cho $\tilde{r}_u(\tau)$ và $\tilde{r}_{uy}(\tau)$ sẽ có

$$\tilde{S}_u(\omega) = \int_{-T}^{T} \tilde{r}_u(\tau) e^{-j\omega\tau} d\tau \qquad \text{và} \qquad \tilde{S}_{uy}(j\omega) = \int_{-T}^{T} \tilde{r}_{uy}(\tau) e^{-j\omega\tau} d\tau \tag{2.40}$$

Giống như đã làm với tín hiệu $x(t)$ ở mục 2.2, ở đây ta lại sử dụng hàm $\delta(t)$ mô tả quá trình trích mẫu $\tilde{r}_u(\tau), \tilde{r}_{uy}(\tau)$ thành $\tilde{r}_u(mT_a), \tilde{r}_{uy}(mT_a)$, m=–N+1, … ,–1,0,1, … , N–1 và đi đến công thức tương đương với (2.29) như sau

$$\tilde{S}_u(n\Omega_N) = T_a \sum_{m=-N+1}^{N-1} \tilde{r}_u(mT_a) e^{-jmn\frac{2\pi}{2N-1}} \tag{2.49a}$$

$$\tilde{S}_{uy}(jn\Omega_N) = T_a \sum_{m=-N+1}^{N-1} \tilde{r}_{uy}(mT_a) e^{-jmn\frac{2\pi}{2N-1}} \tag{2.49b}$$

trong đó $\Omega_N = \dfrac{2\pi}{(2N-1)T_a}$ là khoảng thời gian lấy mẫu giá trị mật độ phổ của các hàm $\tilde{S}_u(\omega)$, $\tilde{S}_{uy}(j\omega)$ trong miền phức và như vậy ta có thể sử dụng kỹ thuật DFT để thực hiện hai công thức trên.

Tuy nhiên, như đã đề cập ở mục trước, do chỉ có các giá trị $\tilde{r}_u(mT_a), \tilde{r}_{uy}(mT_a)$ với chỉ số $|m|$ nhỏ là còn tương đối chính xác, nên ta sẽ không lấy tất cả 2N–1 giá trị $\tilde{r}_u(mT_a), \tilde{r}_{uy}(mT_a)$ đã có vào việc tính mật độ phổ mà chỉ lấy một số ít nằm xung quanh điểm 0, giới hạn bởi chỉ số Lag M

$$\tilde{r}_u(mT_a), \tilde{r}_{uy}(mT_a) \quad \text{có} \qquad |m| \le M.$$

Các giá trị khác không bỏ đi mà đơn thuần được xem là bằng 0. Nói cách khác ta sẽ xác định ảnh Fourier của dãy giá trị gồm 2N–1 phần tử như sau

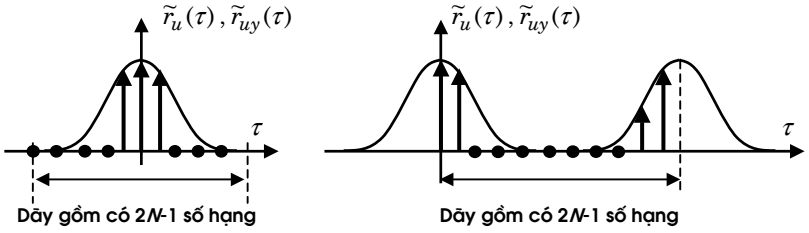
$$0, \cdots, 0, \tilde{r}_u(-MT_a), \cdots, \tilde{r}_u(-T_a), \tilde{r}_u(0), \tilde{r}_u(T_a), \cdots, \tilde{r}_u(MT_a), 0, \cdots, 0 \tag{2.50a}$$

$$0, \cdots, 0, \tilde{r}_{uy}(-MT_a), \cdots, \tilde{r}_{uy}(-T_a), \tilde{r}_{uy}(0), \tilde{r}_{uy}(T_a), \cdots, \tilde{r}_{uy}(MT_a), 0, \cdots, 0 \tag{2.50b}$$

Với việc áp dụng chỉ số Lag cắt bớt những giá trị $\tilde{r}_u(mT_a), \tilde{r}_{uy}(mT_a)$ có sai số lớn đó, hai công thức (2.49) trở thành:

$$\widetilde{S}_u(n\Omega_N) = T_a \sum_{m=-M}^{M} \widetilde{r}_u(mT_a) e^{-jmn\frac{2\pi}{2N-1}}$$

$$\widetilde{S}_{uy}(jn\Omega_N) = T_a \sum_{m=-M}^{M} \widetilde{r}_{uy}(mT_a) e^{-jmn\frac{2\pi}{2N-1}}$$



Hình 2.11: Áp dụng thuật toán FFT cho dãy giá trị (2.50).

Công việc cuối cùng để có thể áp dụng được hàm **`dft()`** một cách trực tiếp cho dãy (2.50) để tính ảnh Fourier $\widetilde{S}_u(n\Omega_N), \widetilde{S}_{uy}(jn\Omega_N)$ của nó là phải chuyển hai công thức trên về dạng $\{\widetilde{x}_k\}$ như (2.29) yêu cầu, tức là chỉ số m của các số hạng trong dãy không chạy từ $-N+1$ đến $N-1$ như trong (2.50) mà phải đi từ 0 đến $2N-2$.

Để làm được điều này, ta áp dụng định lý 2.6 cho công thức (2.29) sẽ thấy giá trị mật độ phổ nhận được $\widetilde{S}_u(n\Omega_N), \widetilde{S}_{uy}(jn\Omega_N)$ trong (2.49) không chỉ đơn thuần là ảnh Fourier của dãy (2.50) mà là của một dãy tuần hoàn trong đó (2.50) là các giá trị trong một chu kỳ. Bởi vậy để có $\widetilde{S}_u(n\Omega_N), \widetilde{S}_{uy}(jn\Omega_N)$ đúng như dạng mà (2.29) yêu cầu, ta có thể áp dụng DFT cho những giá trị trong một chu kỳ khác bắt đầu từ điểm 0 (hình 2.11) và đi đến dạng dãy thích hợp cho FFT như sau:

$$\widetilde{r}_u(0), \widetilde{r}_u(T_a), \cdots, \widetilde{r}_u(MT_a), \underbrace{0, \cdots, 0, \cdots, 0}_{2(N-M-1) \text{ số hạng } 0}, \widetilde{r}_u(-MT_a), \cdots, \widetilde{r}_u(-T_a), \tag{2.51a}$$

$$\widetilde{r}_{uy}(0), \widetilde{r}_{uy}(T_a), \cdots, \widetilde{r}_u(MT_a), \underbrace{0, \cdots, 0, \cdots, 0}_{2(N-M-1) \text{ số hạng } 0}, \widetilde{r}_u(-MT_a), \cdots, \widetilde{r}_u(-T_a), \tag{2.51b}$$

Tổng kết lại tất cả những kết quả nêu trên về vấn đề nhận dạng mật độ phổ tín hiệu $\widetilde{S}_u(n\Omega_N), \widetilde{S}_{uy}(jn\Omega_N)$ từ các giá trị tín hiệu vào/ra $\{\widetilde{u}_k\}, \{\widetilde{y}_k\}$, $k=0,1, \ldots, N-1$ đã đo được của tín hiệu $u(t), y(t)$ ta đi đến thuật toán:

1) Chọn một số M trong khoảng 5÷20% của $2N-1$ làm chỉ số *Lag*.

2) Sử dụng hàm **cor()** để tính $2M+1$ các giá trị hàm tương quan $\tilde{r}_u(mT_a)$, $\tilde{r}_{uy}(mT_a)$, $m=-M, \dots, -1, 0, 1, \dots, M$ từ các giá trị tín hiệu $\{\tilde{u}_k\}, \{\tilde{y}_k\}$, $k=0,1,\ \dots\ , N-1$.

3) Chọn hàm cửa sổ $w(t)$ xác định trong khoảng $|t| \le MT_a$ nhằm làm giảm sai số rò rỉ khi thực hiện công thức (2.49) bằng cách lấy một trong số 8 hàm cửa sổ $w_i(t)$, $i=0,1, \dots, 7$ đã cho tại mục 2.1.6 với $T=2MT_a$ rồi dịch sang trái một khoảng MT_a sao cho hàm có giá trị cực đại tại $t=0$ (nằm đối xứng qua trục tung).

4) Nhân các giá trị $\tilde{r}_u(mT_a)$, $\tilde{r}_{uy}(mT_a)$ với $w(mT_a)$ để có

$$\hat{r}_u(mT_a) = \tilde{r}_u(mT_a)w(mT_a)$$
$$\hat{r}_{uy}(mT_a) = \tilde{r}_{uy}(mT_a)w(mT_a)$$

5) Chọn số nguyên λ là lũy thừa của 2 không nhỏ hơn $2N-1$. Sắp xếp lại dãy $\{\hat{r}_u(mT_a)\}$, $\{\hat{r}_{uy}(mT_a)\}$, $m=-M, \dots, 0, \dots, M$ thành

$$\hat{r}_u(0), \hat{r}_u(T_a), \cdots, \hat{r}_u(MT_a), 0, \cdots, 0, \hat{r}_u(-MT_a), \cdots, \hat{r}_u(-T_a)$$
$$\hat{r}_{uy}(0), \hat{r}_{uy}(T_a), \cdots, \hat{r}_{uy}(MT_a), 0, \cdots, 0, \hat{r}_{uy}(-MT_a), \cdots, \hat{r}_{uy}(-T_a)$$

bằng cách thêm $\lambda-2M-1$ số 0 vào giữa, sao cho mỗi dãy có đúng λ phần tử.

6) Nhân từng phần tử của dãy trên với chu kỳ trích mẫu T_a rồi sử dụng **fft()** để tính ảnh $\tilde{S}_u(n\Omega_\lambda)$, $\tilde{S}_{uy}(jn\Omega_\lambda)$, $n=0,1, \dots, \lambda$, trong đó $\Omega\lambda = \dfrac{2\pi}{\lambda T_a}$.

Chú ý: Hiệu ứng aliasing và leakage của toán tử DFT cũng tác động lên việc nhận dạng $\tilde{S}_u(n\Omega_\lambda)$, $\tilde{S}_{uy}(jn\Omega_\lambda)$ vì đầu vào của chương trình chỉ là dãy giá trị $\tilde{r}_u(mT_a)$, $\tilde{r}_{uy}(mT_a)$, $m=-M, \dots, 0, \dots, M$ chứ không phải bản thân hàm tương quan. Hơn nữa, do ta đã sử dụng số Lag M để hạn chế sự tham gia các giá trị $\tilde{r}_u(mT_a)$, $\tilde{r}_{uy}(mT_a)$ có sai số lớn vào việc tính mật độ phổ, nên toàn bộ $2N-2$ giá trị phổ tính được $\tilde{S}_u(n\Omega_\lambda)$, $\tilde{S}_{uy}(jn\Omega_\lambda)$, $n= 0,1, \dots, 2N-2$ không có độ chính xác như nhau. Cụ thể là khi chỉ số n càng cao, sai số của $\tilde{S}_u(n\Omega_\lambda)$, $\tilde{S}_{uy}(jn\Omega_\lambda)$ càng lớn. Theo kết quả thống kê trong [12] thì chỉ có những giá trị $\tilde{S}_u(n\Omega_\lambda)$, $\tilde{S}_{uy}(jn\Omega_\lambda)$ trong khoảng $0 \le n \le 2M$ là có sai lệch không đáng kể.

Thuật toán trên đã được cài đặt thành hàm và chương trình chuẩn trong các phần mềm tiện dụng như hàm **spa()** trong Toolbox Identification của MatLab. Một chương trình khác có tên **spec()** được cài đặt trên C phục vụ việc tính $\tilde{S}_u(n\Omega_\lambda)$, $\tilde{S}_{uy}(jn\Omega_\lambda)$ từ $\{\tilde{u}_k\}$, $\{\tilde{y}_k\}$, $k=0,1,\ \dots\ , N-1$ dưới dạng hàm cho dưới đây để tham khảo.

Hàm **spec()** này sử dụng thêm hàm con **cor()** phục vụ việc tính $\widetilde{r}_u(mT_a)$, $\widetilde{r}_{uy}(mT_a)$ theo công thức bias (2.45), (2.46) hoặc unbias (2.47), (2.48).

Hàm **spec()** có các biến hình thức sau:

a) Con trỏ **u** chỉ đầu mảng số thực **u[]** chứa các giá trị $\widetilde{u}_k$, k=0,1, … , N–1.

b) Con trỏ **y** chỉ đầu mảng số thực **y[]** chứa các giá trị $\widetilde{y}_k$, k=0,1, … , N–1.

c) Số thực **Ta** chứa hằng số thời gian trích mẫu.

d) Số nguyên **N** chứa độ dài hai dãy $\{\widetilde{u}_k\}$, $\{\widetilde{y}_k\}$, tức là chứa chỉ số N.

e) Số nguyên **Sexp** chứa số mũ lũy thừa 2 của độ dài của dãy kết quả $\{\widetilde{S}_{uy}(jn\Omega_\lambda)\}$, n = 0,1, … , $2^{\mathbf{Sexp}}$–1.

f) Số nguyên **M** chứa chỉ số Lag, tức là độ dài cần phải có của dãy giá trị hàm tương quan $\{\widetilde{r}_{uy}(mT_a)\}$.

g) Số nguyên **bias** xác định giá trị hàm tương quan sẽ được tính theo công thức bias (**bias**=1) hay unbias (**bias**≠1).

h) Số nguyên **w** xác định chỉ số $0 \le i \le 7$ của hàm cửa sổ sẽ được sử dụng nhằm làm giảm sai số rò rỉ (mục 2.1.6) khi sử dụng kỹ thuật DFT. Nếu nội dung của **w** là một số ngoài khoảng [0,7] thì hàm sẽ sử dụng hàm cửa sổ $w_0(t)$.

i) Con trỏ **S** chỉ đầu mảng số phức **S[]** chứa kết quả $\{\widetilde{S}_{uy}(jn\Omega_\lambda)\}$ theo thứ tự **S[0]**$=\widetilde{S}_{uy}(0)$, **S[1]**$=\widetilde{S}_{uy}(j\Omega_\lambda)$, … , **S[n]**$=\widetilde{S}_{uy}(jn\Omega_\lambda)$, …. Như vậy mảng **S[]** phải có độ dài ít nhất là $2^{\mathbf{Sexp}}$.

Hàm **spec()** không làm thay đổi nội dung của hai mảng **u[]** và **y[]**.

Hàm trả về giá trị –1 nếu $M \ge N$. Trường hợp $M < N$ hàm sẽ trả về giá trị là độ dài thực có của dãy kết quả $\{\widetilde{S}_{uy}(jn\Omega_\lambda)\}$ trong mảng **S[]**.

```
int spec(double *u,double *y,double Ta,int N,int Sexp,int M,
         int bias,int w,complex *S)
{  int i,j,p=pow(2,Sexp);
   double *r,T=Ta*2*M,s;
   r=new double[M+1];
   if(cor(u,y,N,M,bias,r)!=0) return(-1);
   for(i=0;i<=M;i++)  S[i]=complex(r[i]);
   for(i=M+1;i<p-M;i++) S[i]=complex(0.);
   if(cor(y,u,N,M,bias,r)!=0) return(-1);
   for(i=1;i<=M;i++) S[p-i]=complex(r[i]);
   for (i=0;i<=M;i++)
```

```
    {
        s=fw(w,Ta*i,T);
        S[i]=S[i]*s*Ta;
        if(i>0) {j=p-i; S[j]=S[j]*s*Ta;}
    }
    fft(Sexp,S);
    delete [] r;
    return(p);
}
```

Muốn sử dụng hàm **spec()** để tính $\{\widetilde{S}_u(n\Omega_\lambda)\}$ ta chỉ cần gọi hàm với hai dãy đầu vào giống nhau **u[k]**= **y[k]**=$\widetilde{u}_k$, k=0,1, … , N–1.

2.3 Nhận dạng mô hình không tham số

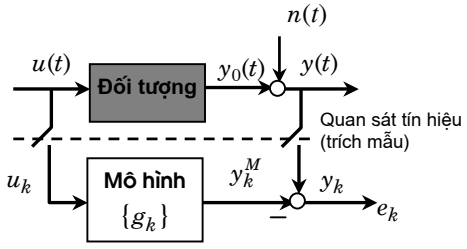
2.3.1 Xác định đường đặc tính tần biên pha

Cho một đối tượng cần nhận dạng. Giả sử rằng thông tin A–priori đã cho ta biết đối tượng có thể mô tả được bởi một mô hình tuyến tính. Nếu chọn dãy hàm trọng lượng $\{g_k\}$ làm lớp mô hình tuyến tính thì nhiệm vụ đặt ra cho bài toán nhận dạng sẽ là: Thông qua việc quan sát các tín hiệu vào ra $u(t)$, $y(t)$ với kết quả quan sát là dãy giá trị $\{u_k\}$, $\{y_k\}$, trong đó

$$u_k = u(kT_a),\ y_k = y(kT_a)$$

và T_a là chu kỳ trích mẫu, hãy xác định dãy hàm trọng lượng $\{g_k\}$ sao cho tổng bình phương các sai lệch e_k là nhỏ nhất (hình 2.12).

Hình 2.12: Phát biểu bài toán nhận dạng mô hình không tham số.



Ký hiệu y_k^M là giá trị tín hiệu đầu ra của mô hình tại thời điểm kT_a thì sai lệch e_k cũng tại thời điểm đó sẽ là

$$e_k = y_k - y_k^M = y_k - g_k * y_k = y_k - \sum_{n=-\infty}^{\infty} g_n u_{k-n}$$

do đó tổng bình phương các sai lệch được tính bằng

$$Q = \sum_{k=-\infty}^{\infty} |e_k|^2 = \sum_{k=-\infty}^{\infty} \left| y_k - \sum_{n=-\infty}^{\infty} g_n u_{k-n} \right|^2$$

Đây là hàm theo biến $\{g_k\}$. Hàm có dạng toàn phương với giá trị không âm và bằng 0 khi và chỉ khi $\{g_k\}$ mô tả chính xác đối tượng, tức là tín hiệu đầu ra của đối tượng đúng bằng tín hiệu đầu ra của mô hình. Bởi vậy để Q có giá trị nhỏ nhất thì cần và đủ là

$$\frac{\partial Q}{\partial g_l} = 0 \qquad \text{với mọi } l.$$

Điều này dẫn đến

$$0 = \sum_{k=-\infty}^{\infty} \left[\left(y_k - \sum_{n=-\infty}^{\infty} g_n u_{k-n} \right) u_{k-l} \right]$$
$$= \sum_{k=-\infty}^{\infty} u_{k-l} y_k - \sum_{n=-\infty}^{\infty} \left(g_n \sum_{k=-\infty}^{\infty} u_{k-n} u_{k-l} \right).$$

Thay $k–l$ bằng q trong tổng thứ nhất và $k–n$ bằng p trong tổng thứ hai

$$\sum_{q=-\infty}^{\infty} u_q y_{q+l} = \sum_{n=-\infty}^{\infty} \left(g_n \sum_{p=-\infty}^{\infty} u_p u_{p+n-l} \right)$$

sau đó lấy giá trị trung bình của hai vế

$$r_{uy}(lT_a) = \sum_{n=-\infty}^{\infty} g_n r_u((n-l)T_a) = g_l * r_u(lT_a)$$

rồi chuyển sang miền phức, chẳng hạn như với chương trình **spec()** đã được trình bày trong mục 2.2, sẽ được

$$\widetilde{S}_{uy}(jn\Omega_\lambda) = G(jn\Omega_\lambda)\, \widetilde{S}_u(n\Omega_\lambda)$$

$$\Rightarrow \qquad G(jn\Omega_\lambda) = \frac{\widetilde{S}_{uy}(jn\Omega_\lambda)}{\widetilde{S}_u(n\Omega_\lambda)}, \tag{2.52}$$

trong đó $\Omega_\lambda = \dfrac{2\pi}{\lambda T_a}$ với λ là một số nguyên lũy thừa của 2 nhỏ nhất nhưng không nhỏ hơn $2N–1$, $\widetilde{S}_u(n\Omega_\lambda)$, $\widetilde{S}_{uy}(jn\Omega_\lambda)$, $n = 0, 1, , \ldots , \lambda–1$ là các giá trị nhận dạng được của hàm mật độ phổ $S_u(\omega)$, $S_{uy}(j\omega)$ tại các điểm tần số $\omega = n\Omega_\lambda$ và N là số các giá trị tín hiệu u_k , y_k đã quan sát được.

Công thức (2.52) nói rằng hàm trọng lượng $g(t)$ của đối tượng tuyến tính, nhận dạng theo phương pháp cực tiểu hóa sai lệch đầu ra, có các giá trị ảnh Fourier là tỷ số giữa giá trị mật độ phổ chéo và giá trị mật độ phổ hợp tín hiệu vào/ra. Kết quả của phương pháp nhận dạng này sẽ không bị ảnh hưởng bởi nhiễu $n(t)$ nếu nhiễu đó tác động tại đầu ra của đối tượng, có giá trị trung bình m_n=0 và không tương quan với tín hiệu vào $u(t)$, vì

$$S_{uy}(j\omega) = S_{uy_0}(j\omega) + \underbrace{S_{un}(j\omega)}_{=0} = S_{uy_0}(j\omega) .$$

Cũng từ công thức trên ta đi đến thuật toán xác định ảnh Fourier $G(jn\Omega_\lambda)$ của dãy hàm trọng lượng $\{g_k\}$ từ N các giá trị tín hiệu u_k , y_k như sau:

1) Nhận dạng mật độ phổ $\widetilde{S}_u(n\Omega_\lambda)$, $\widetilde{S}_{uy}(jn\Omega_\lambda)$, n = 0,1, … ,λ–1 từ những giá trị tín hiệu vào/ra u_k , y_k , k=0,1, … , N–1 của đối tượng theo thuật toán đã trình bày trong mục 2.2, chẳng hạn như theo chương trình **`spec()`**. Chúng được xem là những giá trị gần đúng của $S_u(n\Omega_\lambda)$, $S_{uy}(jn\Omega_\lambda)$.

2) Xác định giá trị ảnh Fourier $G(jn\Omega_\lambda)$, n=0,1, … ,2M của hàm dãy trọng lượng $\{g_k\}$ theo công thức (2.52).

Thuật toán trên đã được cài đặt thành hàm **`nonpar()`** viết trên C cho dưới đây để tham khảo. Hàm này có các biến hình thức sau:

a) Con trỏ **`u`** chỉ đầu mảng số thực **`u[]`** chứa các giá trị u_k, k=0,1, … , N–1.

b) Con trỏ **`y`** chỉ đầu mảng số thực **`y[]`** chứa các giá trị y_k, k=0,1, … , N–1.

c) Số thực **`Ta`** chứa hằng số thời gian trích mẫu.

d) Số nguyên **`N`** chứa độ dài hai dãy $\{u_k\}$, $\{y_k\}$, tức là chứa chỉ số N.

e) Số nguyên **`M`** chứa chỉ số Lag, tức là độ dài cần phải có của dãy giá trị hàm tương quan $\{\ \widetilde{r}_{uy}(mT_a)\ \}$, đồng thời cũng xác định độ dài của dãy kết quả $\{G(jn\Omega_\lambda)\}$ là 2M+1, tức là $n = 0, 1, , \ldots , 2M$ với $\Omega_\lambda = \dfrac{2\pi}{\lambda T_a}$, trong đó λ là một số nguyên lũy thừa của 2 nhỏ nhất nhưng không nhỏ hơn 2N–1.

f) Số nguyên **`bias`** xác định giá trị hàm tương quan sẽ được tính theo công thức bias (**`bias`**=1) hay unbias (**`bias`**≠1).

g) Số nguyên **`w`** xác định chỉ số $0 \le i \le 7$ của hàm cửa sổ sẽ được sử dụng nhằm làm giảm sai số rò rỉ (mục 2.1.6) khi sử dụng kỹ thuật DFT. Nếu nội dung của **`w`** là một số ngoài khoảng [0,7] thì hàm sẽ sử dụng hàm cửa sổ $w_0(t)$.

h) Con trỏ **G** chỉ đầu mảng số phức **G[]** có độ dài $2M+1$ chứa kết quả $\{G(jn\Omega_\lambda)\}$ theo thứ tự **G[0]**=$G(0)$, **G[1]**=$G(j\Omega_\lambda)$, … , **G[2M]**=$G(j(2M)\Omega_\lambda)$.

Hàm **nonpar()** trả về giá trị báo lỗi –1 nếu $M \geq N$. Trường hợp không có lỗi ($M<N$) hàm sẽ trả về giá trị λ. Hàm không làm thay đổi nội dung của các mảng **u[]**, **y[]**.

```
int nonpar(double *u,double *y,double Ta,int N,int M,int bias,
            int w,complex *G)
{
   int p,Sexp=0,k=2*N-1,i;
   complex *Su,*Suy;
   while ((p=pow(2,Sexp))<k) Sexp++;
   Su=new complex[p];
   Suy=new complex[p];
   if((p=spec(u,u,Ta,N,Sexp,M,bias,w,Su))>0)
      if((p=spec(u,y,Ta,N,Sexp,M,bias,w,Suy))>0)
      {
         k=2*M;
         for(i=0;i<=k;i++) G[i]=Suy[i]/Su[i];
      }
   delete [] Su;
   delete [] Suy;
   return(p);
}
```

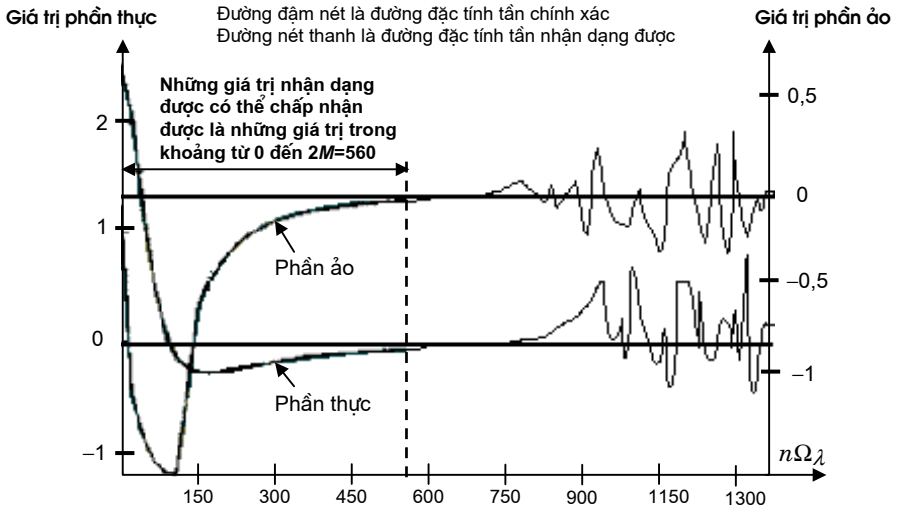
Chú ý rằng trong số các giá trị $\widetilde{S}_u(n\Omega_\lambda)$, $\widetilde{S}_{uy}(jn\Omega_\lambda)$ tính bằng chương trình **spec()** vừa trình bày trong mục 2.2 thì chỉ những giá trị có chỉ số n trong khoảng $0 \leq n \leq 2M$ là có sai số chấp nhận được ([12]). Bởi vậy khi sử dụng (2.52) để xác định $G(jn\Omega_\lambda)$ ta cũng chỉ nên cho n chạy từ 0 đến $2M$.

Hình 2.13 là một ví dụ minh họa cho điều khẳng định trên. Đường đậm nét trong hình là đồ thị biểu diễn phần thực và ảo của đường đặc tính tần $G(j\omega)$ đối tượng quán tính bậc hai:

$$G(s) = \frac{2{,}5}{(1+1{,}6s)(1+0{,}55s)}.$$

(đường đặc tính tần chính xác).

Sử dụng hàm **nonpar()** với 2048 giá trị tín hiệu vào ra $\{u_k\}$, $\{y_k\}$, trong đó thời gian trích mẫu T_a=0,1ms, chỉ số Lag M=280 ta có được dãy $\{G(jn\Omega_\lambda)\}$, Ω_λ =15,8s^{-1} tính theo thuật toán vừa trình bày. Đường nét thanh trong hình là đường biểu diễn dãy các giá trị đó (đường đặc tính tần nhận dạng được).



Hình 2.13: So sánh đường đặc tính tần $G(j\omega)$ nhận dạng được với đường thực phải có của một đối tượng quán tính bậc 2.

Ta nhận thấy ngay rằng trong khoảng

$0 \le n \le 560 = 2M$,

sai lệch giữa hai đường là không đáng kể, thậm chí chũng gần như trùng nhau. Ngược lại, khi $n > 2M = 560$ đường đặc tính tần nhận dạng được có một sai lệch khá lớn so với đường đặc tính tần chính xác.

2.3.2 Xác định hàm trọng lượng từ đường đặc tính tần

Để tính ngược giá trị g_k của hàm trọng lượng $g(t)$ từ dãy $G(jn\Omega_\lambda)$, n=0,1, … ,2M ta sử dụng chương trình **`invdft()`** đã trình bày trong mục 2.1.8. Kết quả nhận được sẽ là dãy các giá trị $g_k = g(kT_a)$ của $g(t)$.

1) Sử dụng hàm **`nonpar()`** để xác định $\{G(jn\Omega_\lambda)\}$, $n = 0, 1, , \dots, 2M$ với $\Omega_\lambda = \dfrac{2\pi}{\lambda T_a}$,

   trong đó λ là một số nguyên lũy thừa của 2 nhỏ nhất nhưng không nhỏ hơn $2N$–1 và N là độ dài dãy giá trị tín hiệu $\{u_k\}$, $\{y_k\}$ đo được với chu kỳ lấy mẫu T_a.

2) Gắn thêm những giá trị 0 vào sau dãy $G(jn\Omega_\lambda)$, n=0,1, … ,2M để dãy mới có đúng λ phần tử.

3) Sử dụng hàm **invdft()** để chuyển ngược $\{G(jn\Omega_\lambda)\}$, $n=0,1,,\ldots,\lambda-1$, trong đó $G(jn\Omega_\lambda)=0$ nếu $n > 2M$ sang miền thời gian. Kết quả thu được sẽ là $\{g_k\}$, $k=0,1,\ldots,\lambda-1$ với $g_k = g(kT_a)$.

Câu hỏi ôn tập và bài tập

1. Với những điều kiện gì thì việc nhận dạng ảnh Fourier của một tín hiệu theo công thức $X(jn\Omega)=\sum_{k=0}^{N-1} x(kT_a)e^{-jkn\frac{2\pi}{N}}$ với $\Omega=\frac{2\pi}{NT_a}$ sẽ không có hiệu ứng trùng phổ và với những điều kiện gì thì việc nhận dạng ảnh Fourier của một tín hiệu cũng theo công thức đó sẽ không có hiệu ứng rò rỉ.
2. Với những tín hiệu như thế nào thì việc nhận dạng ảnh Fourier của nó theo công thức $X(jn\Omega)=\sum_{k=0}^{N-1} x(kT_a)e^{-jkn\frac{2\pi}{N}}$ với $\Omega=\frac{2\pi}{NT_a}$ sẽ cho ra kết quả đúng, tức là không chứa cả hai loại sai số trùng phổ và sai số rò rỉ?.
3. Hãy chỉ rằng từ dãy gồm N giá trị u_k , y_k, k=0,1, … , N–1 đo được của tín hiệu $u(t)$, $y(t)$ trong khoảng thời gian [0, T) với chu kỳ lấy mẫu T_a , tức là $u_k = u(kT_a)$, $y_k = y(kT_a)$ và $T = NT_a$ thì các giá trị của hàm tương quan

$$r_{uy}(mT_a) \approx \frac{1}{p}\sum_{k=0}^{N-|m|-1} u_k y_{k+m} \text{ , } m = -N+1,\ldots,-1,0,1, \ldots, N-1$$

trong đó $p=\begin{cases} N & \text{nếu nhận dạng bias} \\ N-m & \text{nếu nhận dạng unbias} \end{cases}$

có thể được tính nhanh theo thuật toán sau:

a) Tìm một số nguyên λ lũy thừa của 2 nhỏ nhất nhưng không nhỏ hơn $2N$–1, sau đó xây dựng dãy có λ phần tử như sau

$$\widetilde{u}_k=\begin{cases} u_k & \text{nếu } 0 \le k \le N-1 \\ 0 & \text{nếu } N \le k \le \lambda-1 \end{cases},$$

$$\widetilde{y}_k=\begin{cases} y_{k-N+1} & \text{nếu } N-1 \le k \le 2N-2 \\ 0 & \text{nếu } \ 0 \le k \le N-2 \ \text{ hoặc } \ 2N-1 \le k \le \lambda-1 \end{cases}$$

b) Gọi hàm **fft()** để tính ảnh Fourier $\widetilde{U}_a(jn\Omega_\lambda)$ của dãy $\{\widetilde{u}_k\}$ và $\widetilde{Y}_a(jn\Omega_\lambda)$ của dãy $\{\widetilde{y}_k\}$, n = 0,1 … , λ–1.

c) Xác định dãy $\widetilde{R}_a(jn\Omega_\lambda)=\overline{\widetilde{U}_a(jn\Omega_\lambda)}\widetilde{Y}_a(jn\Omega_\lambda)$, n = 0,1 …, λ–1.

d) Chuyển ngược $\{\widetilde{R}_a(jn\Omega_\lambda)\}$ sang miền thời gian bằng cách sử dụng hàm **fft()** một lần nữa với dãy đầu vào $\overline{\widetilde{R}_a(jn\Omega_\lambda)} = \widetilde{U}_a(jn\Omega_\lambda)\overline{\widetilde{Y}_a(jn\Omega_\lambda)}$.

e) Chia từng phần tử của dãy kết quả thu được sau khi chuyển ngược sang miền thời gian cho hằng số p ta sẽ được dãy các giá trị $\{r_{uy}(mT_a)\}$ theo thứ tự

$\{r_{uy}((-N+1)T_a), \ldots, r_{uy}(-T_a), r_{uy}(T_a), \ldots, r_{uy}((-N+1)T_a)\}$

4. Hãy để xây dựng một chương trình tính nhanh các giá trị mật độ phổ tín hiệu $S_{uy}(jn\Omega\lambda)$, n = 0,1 …, λ–1 từ các giá trị tín hiệu u_k, y_k, k=0,1, … , N–1 đo được của tín hiệu $u(t)$, $y(t)$ trong khoảng thời gian [0, T) với chu kỳ lấy mẫu T_a bằng cách sử dụng thuật toán đã cho ở bài 3, trong đó $T = NT_a$, λ là số nguyên nhỏ nhất có dạng lũy thừa của 2 nhưng không nhỏ hơn 2N–1 và $\Omega\lambda = \dfrac{2\pi}{\lambda T_a}$.

5. Chứng minh rằng các giá trị mật độ phổ

$$S_{uy}(jn\Omega) = T_a \sum_{m=0}^{N-1} r_{uy}(mT_a) e^{-jmn\frac{2\pi}{N}}$$

với $\Omega = \dfrac{2\pi}{NT_a}$, tính từ N giá trị u_k, y_k, k=0,1, … , N–1 đo được của tín hiệu $u(t)$, $y(t)$ trong khoảng thời gian [0, T) với chu kỳ lấy mẫu T_a, có thể được xác định theo công thức

$$S_{uy}(jn\Omega) = \frac{T_a}{p}\overline{U_a(jn\Omega)}Y_a(jn\Omega)$$

trong đó $p = \begin{cases} N & \text{nếu nhận dạng bias} \\ N-m & \text{nếu nhận dạng unbias} \end{cases}$

$$U_a(jn\Omega) = \sum_{k=0}^{N-1} u_k e^{-jkn\frac{2\pi}{N}}, \qquad Y_a(jn\Omega) = \sum_{k=0}^{N-1} y_k e^{-jkn\frac{2\pi}{N}},$$

nói cách khác $U_a(jn\Omega)$ là ảnh Fourier thu được nhờ **fft()** của $\{u_k\}$, k=0,1, … , N–1 và $Y_a(jn\Omega)$ là ảnh Fourier của $\{y_k\}$, k=0,1, , … , N–1.

3 NHẬN DẠNG MÔ HÌNH LIÊN TỤC, TUYẾN TÍNH CÓ THAM SỐ TỪ MÔ HÌNH KHÔNG THAM SỐ

Mô hình liên tục có tham số được dùng để mô tả đối tượng tuyến tính ở chương này là hàm truyền đạt biểu diễn tỷ số giữa ảnh Laplace $Y(s)$ của tín hiệu đầu ra $y(t)$ và ảnh Laplace $U(s)$ của tín hiệu đầu vào $u(t)$:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_0 + b_1 s + \cdots + b_{n_b} s^{n_b}}{a_0 + a_1 s + \cdots + a_{n_a} s^{n_a}}, \qquad n_a \geq n_b \tag{3.1}$$

trong đó n_a , n_b có thể là cho trước (mô hình có cấu trúc) hoặc là những tham số cần phải được xác định (mô hình không có cấu trúc). Bài toán đặt ra là từ mô hình không tham số đã có, hãy xác định $b_0, b_1, \ldots, b_{n_b}, a_0, a_1, \ldots, a_{n_a}$ thuộc $\mathbb{R}$ cho (3.1).

Mô hình không tham số đã có là hàm quá độ $h(t)$ thu được tại đầu ra nhờ phương pháp nhận dạng chủ động với tín hiệu chọn trước là hàm Heaviside $1(t)$ ở đầu vào, hoặc dãy các giá trị ảnh Fourier $G(jn\Omega_\lambda)$ của hàm trọng lượng $g(t)$ thu được trên cơ sở quan sát các tín hiệu vào/ra (nhận dạng bị động). Phương pháp nhận dạng dãy giá trị $G(jn\Omega_\lambda)$, $n = 0,1, \ldots, 2M$ với M là một chỉ số Lag chọn trước trên cơ sở phân tích phổ tín hiệu, có để ý đến nhiệm vụ lọc nhiễu đã được trình bày trong chương 2.

3.1 Xác định tham số mô hình từ hàm quá độ

3.1.1 Những kết luận tổng quát

Một số kết luận tổng quát trình bày dưới đây có thể chưa đủ cho việc xác định toàn bộ các tham số $b_0, b_1, \ldots, b_{n_b}, a_0, a_1, \ldots, a_{n_a} \in \mathbb{R}$ của mô hình (3.1) từ hàm quá độ $h(t)$, song cũng là cần thiết để có được những kiến thức ban đầu hỗ trợ cho các công việc tiếp theo. Những kết luận đó sẽ là:

- Kết luận về bậc mô hình, tức là về n_a và n_b .
- Kết luận về các thành phần cơ bản như khâu khuếch đại P, tích phân I, vi phân D có trong mô hình (3.1).

– Kết luận về dạng các điểm cực cũng như các điểm không của (3.1) và nếu có thể thì còn về sự phân bố của chúng trong mặt phẳng phức.

Kết luận 1: 1) Nếu $h(+0) = 0$ thì $n_a > n_b$. Ngược lại nếu $h(+0) \neq 0$ thì $n_a = n_b$.

2) Nếu $\frac{d}{dt}h(+0) = 0$ thì $n_a - n_b > 1$. Ngược lại nếu $\frac{d}{dt}h(+0) \neq 0$ thì $n_a = n_b + 1$.

3) Nếu $h(+\infty) = \infty$ thì $a_0 = 0$, hay trong $G(s)$ có một khâu I nối tiếp:

$$G(s) = \frac{b_0 + b_1 s + \cdots + b_{n_b} s^{n_b}}{s(a_1 + a_2 s + \cdots + a_{n_a} s^{n_a - 1})} .$$

4) Nếu $h(+\infty) = 0$ thì $b_0 = 0$, hay trong $G(s)$ có một khâu D nối tiếp:

$$G(s) = \frac{s(b_1 + b_2 s + \cdots + b_{n_b} s^{n_b - 1})}{a_0 + a_1 s + \cdots + a_{n_a} s^{n_a}} .$$

5) Nếu $h(+\infty)$ là một hằng số khác 0 thì trong $G(s)$ có một khâu P nối tiếp với hệ số khuếch đại $k = \frac{b_0}{a_0}$:

$$G(s) = k \frac{1 + \tilde{b}_1 s + \cdots + \tilde{b}_{n_b} s^{n_b}}{1 + \tilde{a}_1 s + \cdots + \tilde{a}_{n_a} s^{n_a}} .$$

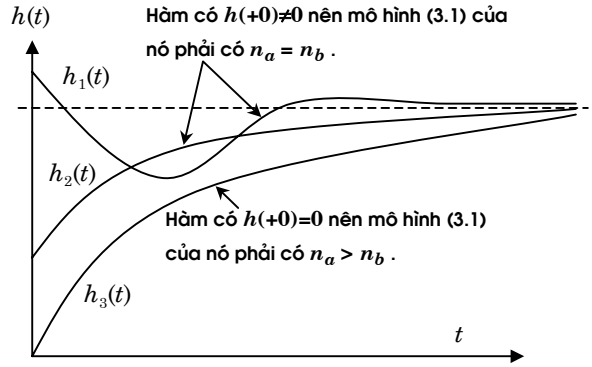
Để chứng minh kết luận 1 ta có thể sử dụng các công thức sau của toán tử Laplace:

1) $x(+0) = \lim_{s \to \infty} sX(s)$ (3.2a)

2) $x(+\infty) = \lim_{s \to 0} sX(s)$ (3.2b)

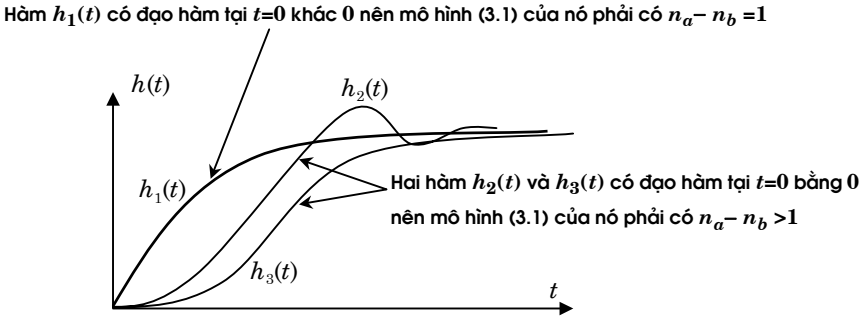
3) $\mathcal{L}\{ \frac{dx}{dt} \} = sX(s) - x(+0)$ (3.2c)

trong đó ký hiệu $\mathcal{L}\{\cdot\}$ chỉ phép tính lấy ảnh Laplace và việc đó dành cho bạn đọc như những bài tập ôn luyện.

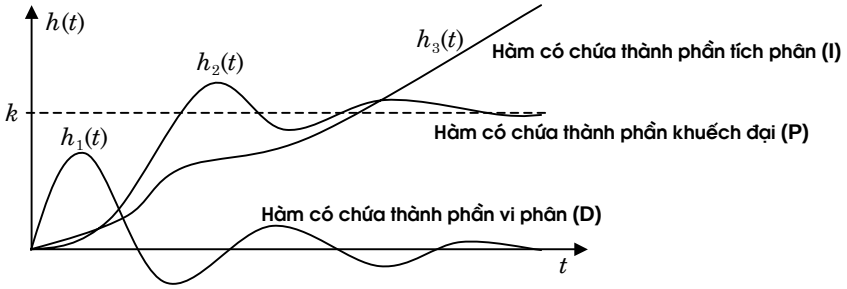


Hình 3.1: Xác định quan hệ ban đầu giữa bậc của đa thức tử số và đa thức mẫu số thông qua giá trị của hàm quá độ $h(t)$ tại điểm $t=0$.

Các hình 3.1, 3.2 và 3.3 minh họa trực quan nội dung kết luận 1.



Hình 3.2: Một số dạng hàm quá độ của đối tượng mà mô hình (3.1) của nó có bậc của tử số lớn hơn bậc của mẫu số.



Hình 3.3: Xác định các thành phần cơ bản (**P, I, D**) có trong mô hình (3.1) thông qua dạng hàm quá độ khi $t \to \infty$.

Ví dụ 1: Một đối tượng với hàm truyền đạt $G(s)$ dạng (3.1) có hàm quá độ (hình 3.4)

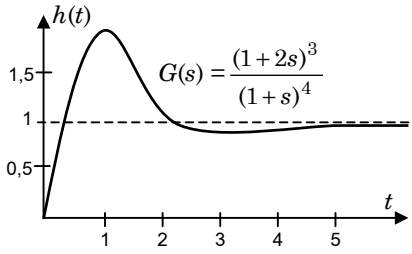
$$h(t) = 1 - \frac{-t^3 + 15t^2 - 42t + 6}{6e^t}.$$

Do $h(+0) = 0$ nên $n_a > n_b$. Thêm nữa đạo hàm

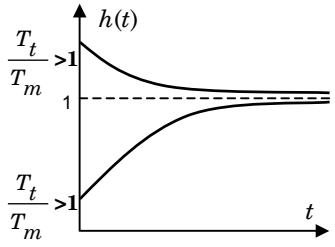
$$\frac{d}{dt}h(t) = -\frac{t^3 - 18t^2 + 72t - 48}{6e^t}$$

tại điểm $t = +0$ có $\frac{dh(+0)}{dt} \neq 0$ nên $n_a - n_b = 1$.

Ngoài ra, vì $\lim_{t\to\infty} h(t) = 1$ nên trong $G(s)$ có thành phần khâu khuếch đại, tức là hai tham số đầu sẽ thỏa mãn $\frac{b_0}{a_0} = 1$. ❑



Hình 3.4: Hình minh họa cho ví dụ 1.



Hình 3.5: Hàm quá độ cho ví dụ 2.

Ví dụ 2: Từ hàm quá độ $h(t)$ cho trong hình 3.5 của một đối tượng ta có thể suy ra được những kết luận cho mô hình $G(s) = \frac{1+T_t s}{1+T_m s}$ của nó như sau:

- $T_t > T_m$ nếu $h(t)$ khi $t < \infty$ luôn có giá trị lớn hơn $h(\infty) = 1$.
- $T_t < T_m$ nếu $h(t)$ khi $t < \infty$ luôn có giá trị nhỏ hơn $h(\infty) = 1$. ❑

Tổng quát hóa kết quả minh họa của ví dụ 2, tiếp theo ta sẽ xét đối tượng mô tả bởi

$$G(s) = k\frac{(1+T_1's)(1+T_2's)\cdots(1+T'_{n_b}s)}{(1+T_1s)(1+T_2s)\cdots(1+T'_{n_a}s)} \tag{3.2}$$

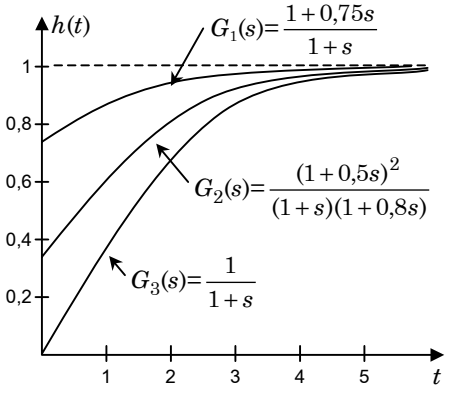
tức là từ $h(t)$ nhận biết được rằng đối tượng có thành phần khuếch đại. Tham số T_i được gọi là hằng số thời gian của đa thức mẫu số và T_i' là hằng số thời gian đa thức tử số. Không mất tính tổng quát nếu ta giả thiết

$$T_1 \le T_2 \le \ldots \le T_{n_a} \quad \text{và} \quad T_1' \le T_2' \le \ldots \le T'_{n_b}$$

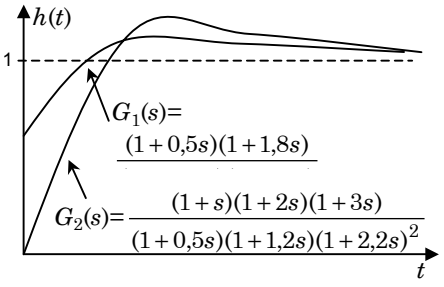
Kết luận 2: Nếu $h(t)$ không lượn sóng và không giảm, tức là $h(t)$ không chứa thành phần quá điều chỉnh, thì các tham số T_i , T_i' của mô hình (3.2) tương ứng phải là những số thực và phải thỏa mãn:

$$T_{n_a} > T'_{n_b},\ T_{n_a-1} > T'_{n_b-1},\ \ldots,\ T_{n_a-n_b-1} > T_1' \tag{3.3}$$

Hình 3.6 minh họa một số dạng hàm quá độ $h(t)$ không giảm, không lượn sóng (không có độ quá điều chỉnh) mà mô hình (3.2) tương ứng của nó có dạng như kết luận 2 vừa nêu.



Hình 3.6: Một số dạng hàm quá độ minh họa cho kết luận 2



Hình 3.7: Một số dạng hàm quá độ minh họa cho kết luận 3.

Kết luận 3: Nếu $h(t)$ không lượn sóng, có độ quá điều chỉnh nhưng sau đó giảm dần về $h(\infty)=k$ và không nhỏ hơn k thì tham số T_i , T_i' của mô hình (3.2) tương ứng phải là những số thực và phải tồn tại duy nhất một chỉ số $l \in \{1,2, \dots , n_b\}$ để một trong n_b bất đẳng thức (3.3) không được thỏa mãn.

Hình 3.7 là một số ví dụ về các hàm quá độ $h(t)$ có dạng thỏa mãn giả thiết của kết luận 3 cùng với mô hình (3.2) tương ứng của nó. Ở đây ta thấy trong $G_1(s)$ có

$$T_2=1,5 < T_2'=1,8$$

và trong $G_2(s)$ có

$$T_4=2,2 < T_3'=3.$$

Kết luận 4: Nếu $h(t)$ có m điểm cực trị, trong đó điểm cực đại nằm trên đường $h(\infty)=k$ và điểm cực tiểu nằm dưới đường $h(\infty)=k$ thì những tham số T_i , T_i' của mô hình (3.2) tương ứng phải là những số thực và phải tồn tại m chỉ số trong khoảng $\{1, 2, \dots , n_b\}$ để có m bất đẳng thức trong (3.3) không được thỏa mãn.

Để minh họa cho kết luận 4 ta xét lại ví dụ 1 với hàm $h(t)$ cho trong hình 3.8. Hàm này có dạng thỏa mãn giả thiết yêu cầu trong kết luận 4 với m=3 điểm cực trị. Kiểm tra lại mô hình (3.2) tương ứng của nó

$$G(s) = \frac{(1+2s)^3}{(1+s)^4}$$

ta thấy đúng là có 3 bất đẳng thức (3.3) không được thỏa mãn và đó là

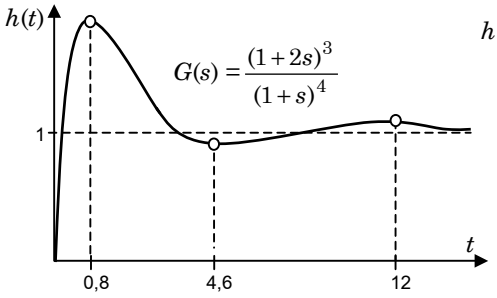
$$T_{i+1} = 1 < T_i' = 2, \qquad i = 1, 2, 3.$$

Ngược lại, từ

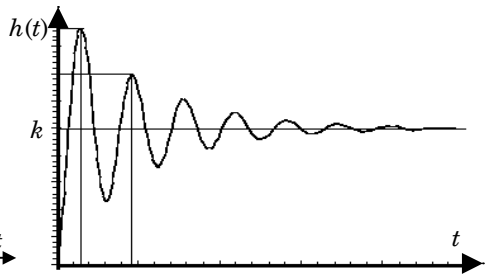
$$\frac{d}{dt}h(t) = -\frac{t^3 - 18t^2 + 72t - 48}{6e^t}$$

có $\frac{d}{dt}h(t) = 0$ tại $t_1 \approx 0{,}8$, $t_2 \approx 4{,}6$, $t_3 \approx 12{,}6$.

Suy ra: $h_{max1} \approx 2{,}4$, $h_{min} \approx 0{,}9$, $h_{max2} \approx 1{,}01$.



Hình 3.8: Hàm quá độ với 3 điểm cực trị.



Hình 3.9: Dạng hàm quá độ có vô số điểm cực trị phân bố cách đều nhau trên và dưới đường giới hạn $h(\infty)=k$.

Theo kết quả của kết luận 4, nếu nhận thấy từ đường đồ thị hàm quá độ là $h(t)$ có hữu hạn m điểm cực trị phân bố xen kẽ trên và dưới đường giới hạn $h(\infty)=k$ thì có thể khẳng định ngay được rằng mô hình (3.1) của nó phải có $n_a \geq n_b \geq m$.

Một dạng nữa của hàm quá độ $h(t)$ thường gặp mà ta cần phải để ý là $h(t)$ có vô số điểm cực trị phân bố xen kẽ trên/dưới đường giới hạn $h(\infty)=k$ và cách đều nhau như ở hình 3.9 mô tả. Chắc chắn rằng các hàm $h(t)$ dạng này phải có chứa những thành phần dao động (có thể biên độ tắt dần) dạng

$$e^{-\alpha t}(\sin\omega t + \cos\omega t)$$

bởi vậy mô hình (3.1) tương ứng của nó cũng phải có các điểm cực phức (không nằm trên trục thực) và ta đi đến kết luận thứ 5 như sau:

Kết luận 5: Nếu $h(t)$ có vô số điểm cực trị cách đều nhau, trong đó điểm cực đại nằm trên đường $h(\infty)=k$ và điểm cực tiểu nằm dưới đường $h(\infty)=k$ thì mô hình (3.1) của nó phải có các điểm cực là những giá trị phức (đa thức mẫu số có nghiệm phức).

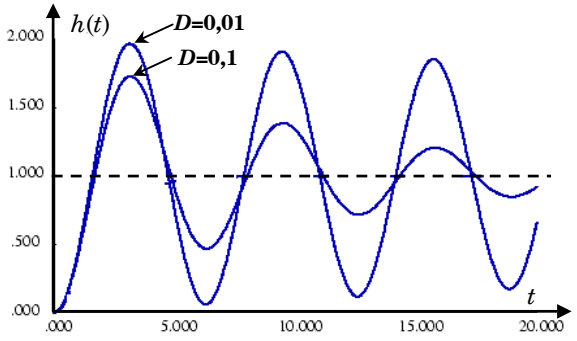
Ví dụ 3: Xét đối tượng với hàm truyền đạt

$$G(s) = \frac{1}{1+2Ds+s^2}, \qquad D < 1.$$

Đa thức mẫu số có hai nghiệm phức $s_{1,2}=-D\pm ja$. Hàm quá độ tương ứng của nó có dạng

$$h(t) = 1-\frac{1}{a}[D\sin(at)+a\cos(at)]e^{-Dt},$$

với $a=\sqrt{1-D^2}$. Hàm $h(t)$ có vô số các điểm cực trị phân bố nằm xen kẽ trên/dưới đường giới hạn $h(\infty)=1$ và cách đều nhau (hình 3.10). ❑



Hình 3.10: Hàm quá độ có vô số điểm cực phân bố xen kẽ và cách đều nhau minh họa cho ví dụ 3.

Năm kết luận trên đặt một cơ sở cho sự suy luận về dạng mô hình để ta có thể tiếp tục giải quyết được bài toán xác định tham số từ hàm quá độ $h(t)$. Tuy nhiên một điều phải chú ý là không bao giờ lại chọn mô hình có cấu trúc phức tạp quá mức cần thiết, ví dụ như từ $h(t)$ mà ta đã suy ra được $n_a-n_b>1$, $h(t)$ không có dạng lượn sóng, có thành phần khuếch đại $h(\infty)=k$ và không bao giờ vượt quá giá trị giới hạn $h(\infty)=k$, tức là mô hình có các giá trị T_i , T_i' thực thỏa mãn (3.3), thì đủ nếu ta giả thiết mô hình của nó là:

$$G(s) = \frac{k}{(1+T_1 s)(1+T_2 s)}.$$

Với phương châm xây dựng mô hình đủ chính xác như yêu cầu chứ không phải chính xác như có thể, sau đây ta sẽ tập trung chủ yếu vào việc xác định tham số cho:

1) Mô hình PT_1: $G(s) = \dfrac{k}{1+Ts}$.

2) Mô hình IT_1: $G(s) = \dfrac{k}{s(1+Ts)}$.

3) Mô hình IT_n: $G(s) = \dfrac{k}{s(1+Ts)^n}$.

4) Mô hình PT_2: $G(s) = \dfrac{k}{(1+T_1 s)(1+T_2 s)}$, $T_1 \neq T_2$.

5) Mô hình PT_n: $G(s) = \dfrac{k}{(1+Ts)^n}$.

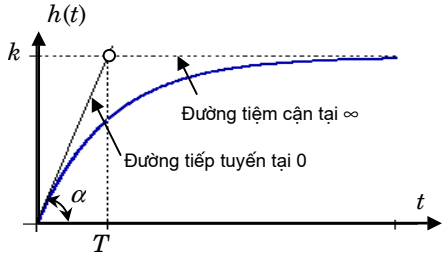
6) Mô hình Lead/Lag: $G(s) = \dfrac{1+T_t s}{1+T_m s}$.

7) Mô hình khâu dao động bậc hai tắt dần: $G(s) = \dfrac{k}{1+2DTs+T^2 s^2}$, $0<D<1$.

3.1.2 Xác định tham số mô hình quán tính bậc nhất

Giả sử rằng khi kích thích một đối tượng tuyến tính bằng hàm Heaviside $1(t)$ tại đầu vào ta đo được hàm $h(t)$ ở đầu ra có dạng như hình 3.11. Dựa theo 5 kết luận vừa nêu trong mục trước thì:

- bậc của đa thức mẫu số phải lớn hơn bậc của tử số một bậc, tức là $n_a - n_b = 1$, vì $h(t)$ xuất phát từ 0 và có đạo hàm tại đó khác 0,
- do có $h(\infty)=k$ nên hàm truyền đạt $G(s)$ của đối tượng phải thỏa mãn $G(0)=k$,
- hàm $h(t)$ không có dạng lượn sóng, luôn có xu hướng tăng dần đến giá trị giới hạn k nên các điểm cực và không của $G(s)$ phải là số thực thỏa mãn (3.3).



Hình 3.11: Hàm quá độ của khâu quán tính bậc nhất.

Căn cứ những sơ kiện như vậy ta thấy hoàn toàn đủ để có thể mô tả đối tượng nếu sử dụng hàm truyền đạt

$$G(s) = \frac{k}{1+Ts} \tag{3.4}$$

và vấn đề còn lại là phải xác định nốt tham số T, vì đã có $k = h(\infty)$.

Xuất phát từ (3.4) và công thức giới hạn (3.2a) ta có tại điểm 0:

$$\mathrm{tg}\alpha = \frac{dh(+0)}{dt} = g(+0) = \lim_{s\to\infty} sG(s) = \lim_{s\to\infty} \frac{ks}{1+Ts} = \frac{k}{T}$$

cho nên T có thể sẽ được xác định một cách đơn giản như sau:

1) Kẻ đường tiếp tuyến với $h(t)$ tại t=0.
2) Xác định giao điểm của đường tiếp tuyến đó với đường tiệm cận k=$h(\infty)$.
3) Hoành độ của giao điểm vừa xác định chính là tham số T cần tìm (hình 3.11).

Tuy nhiên phương thức xác định T vừa nêu cũng có nhược điểm là phụ thuộc khá nhiều vào độ chính xác việc kẻ đường tiếp tuyến, tức là phụ thuộc vào sự khéo tay của ta. Nếu rủi ro ta đặt đường tiếp tuyến tại 0 "sai một ly" thì có thể sẽ nhận được kết quả T có sai số "một dặm", nhất là ở những hàm $h(t)$ có hệ số khuếch đại k tương đối lớn. Để tránh điều "rủi ro" này người ta đã đi từ (3.4) để có $H(s) = \frac{G(s)}{s}$ rồi chuyển ngược sang miền thời gian sẽ được:

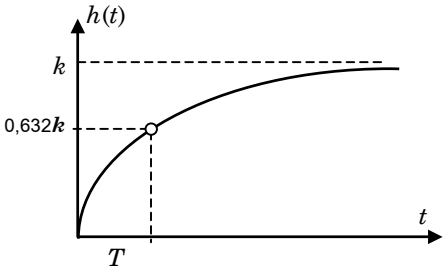
$$h(t) = k(1 - e^{-\frac{t}{T}})$$

và như vậy tại thời điểm T thì:

$$h(T) = k(1-e^{-1}) \approx 0{,}632 \cdot k \qquad (3.5)$$

Nói cách khác, tại đúng thời điểm T hàm $h(t)$ sẽ đạt được 63,2% giá trị cực đại. Giá trị (3.5) được xem như công thức xác định tham số T cho mô hình (3.4) từ đường thực nghiệm $h(t)$. Ta đi đến thuật toán (hình 3.12):

1) Kẻ đường tiệm cận với $h(t)$ tại t=∞ để có k=$h(\infty)$.
2) Xác định điểm có tung độ bằng $0{,}632 \cdot k$ của $h(t)$.
3) Hoành độ của điểm vừa xác định chính là tham số T cần tìm.



Hình 3.12: Xác định gần đúng tham số T.

Trong khá nhiều trường hợp, với một lý do khách quan nào đó người ta không tạo ra được hàm Heaviside ở đầu vào mà thay vào đó là hàm $u(t)$=$u_0 1(t)$ thì do tính chất tuyến tính của đối tượng, đáp ứng $y(t)$ tại thời điểm T cũng có giá trị đúng bằng 63,2% giá trị cực đại $y(\infty)$ của nó. Bởi vậy thuật toán trên vẫn sử dụng được để xác

định tham số mô hình (3.4) từ tín hiệu $y(t)= u_0\, h(t)$ đo được ở đầu ra với một thay đổi nhỏ như sau:

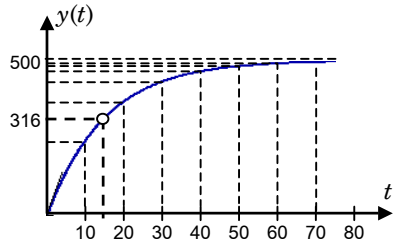
1) Kẻ đường tiệm cận với $y(t)$ tại $t\to\infty$ để có $y(\infty)$, sau đó suy ra $k = \dfrac{y(\infty)}{u_0}$.

2) Xác định điểm có tung độ bằng $0{,}632\cdot y(\infty)$ của $y(t)$.

3) Hoành độ của điểm vừa xác định chính là tham số T cần tìm.

Ví dụ: Xét đối tượng là lò nhiệt điện trở với tín hiệu đầu vào $u(t)$ đo theo áp trong dải từ 0V đến 10V. Tín hiệu đầu ra $y(t)$ của lò điện trở là nhiệt độ. Khi đặt điện áp 10V tại thời điểm t=0 người ta đã đo được các giá trị sau ở đầu ra:

***t* [giây *s*]**	0	10	20	30	40	50	60	70	80	90
***y*(*t*) [^0C]**	0	205	335	395	435	441	449	453	457	459



Hình 3.13: Minh họa cho ví dụ nhận dạng tham số mô hình $\mathbf{PT_1}$.

Dựng đường đồ thị từ các kết quả đo được đó ta có hình 3.13. Với đường đồ thị này ta được $h(t)=\dfrac{y(t)}{10}$. Theo các bước của thuật toán vừa trình bày, ta có

$$y(\infty) = 500 \quad \Rightarrow \quad k = h(\infty) = 50$$

Sau đúng khoảng thời gian T, tín hiệu đầu ra $y(t)$ sẽ đạt được giá trị xấp xỉ bằng 63,2% giá trị cực đại $y(\infty)$ của nó, tức là khi đó nó có giá trị

$$0{,}632\cdot y(\infty) = 316.$$

Suy ra

$$y(T) = 316 \quad \Rightarrow \quad T \approx 14$$

Vậy mô hình của lò điện trở là:

$$G(s) = \frac{50}{1+14s}$$

❑

3.1.3 Xác định tham số cho mô hình tích phân quán tính

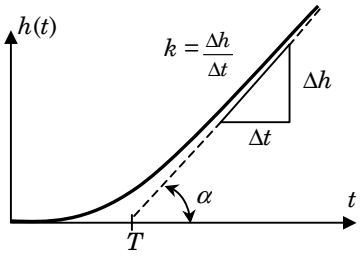
Cho một đối tượng tuyến tính cần nhận dạng. Bằng phương pháp thực nghiệm chủ động với tín hiệu đầu vào đặt trước là hàm 1(t), người ta đo được ở đầu ra hàm quá độ $h(t)$ có dạng cho trong hình 3.14. Hàm $h(t)$ cũng có thể nhận được thông qua phương pháp nhận dạng bị động có để ý đến sự ảnh hưởng của nhiễu bằng phân tích phổ tín hiệu đã được trình bày ở chương 2.

Căn cứ theo những kết luận đã trình bày ở mục 3.1.1 ta đi đến nhận định ban đầu về mô hình tham số của đối tượng như sau:

- bậc của đa thức mẫu số phải lớn hơn bậc của đa thức tử số ít nhất là 2 bậc, tức là $n_a - n_b \geq 2$, vì $h(t)$ xuất phát từ 0 và có đạo hàm tại đó cũng bằng 0,
- do có $\lim_{t\to\infty} h(t)=\infty$ nên hàm truyền đạt $G(s)$ của đối tượng phải có chứa thành phần tích phân, nói cách khác nó phải có dạng

$$G(s) = \frac{b_0 + b_1 s + \cdots + b_{n_b} s^{n_b}}{s(a_1 + a_2 s + \cdots + a_{n_a} s^{n_a - 1})},$$

- hàm $h(t)$ không có dạng lượn sóng nên các điểm cực và không của $G(s)$ phải là số thực.



Hình 3.14: Hàm quá độ của khâu $\mathbf{IT_1}$.

Trên cơ sở các sơ kiện vừa nêu và theo nguyên tắc không phức tạp mô hình quá mức cần thiết, ta có thể giả thiết rằng đối tượng có hàm truyền đạt thuộc lớp IT_1 :

$$G(s) = \frac{k}{s(1+Ts)}. \tag{3.6}$$

hoặc lớp các mô hình IT_n:

$$G(s) = \frac{k}{s(1+Ts)^n}. \tag{3.7}$$

trong đó n có thể là cho trước, song cũng có thể là một tham số cần được xác định.

Xác định tham số mô hình tích phân quán tính bậc nhất

Trước tiên ta tập trung vào mô hình IT_1 (3.6). Bài toán đặt ra ở đây là phải xác định các tham số k và T từ đường thực nghiệm $h(t)$ đã có.

Kẻ đường tiệm cận của $h(t)$ khi $t\to\infty$. Theo công thức tính giới hạn (3.2b) có

$$\mathrm{tg}\alpha = \lim_{t\to\infty}\frac{dh(t)}{dt} = \lim_{t\to\infty} g(t) = \lim_{s\to 0} sG(s) = k\ ,$$

do đó muốn xác định k ta chỉ cần lấy một đoạn bất kỳ của đường tiệm cận, chiếu nó lên hai trục tọa độ để có Δh và Δt (hình 3.14), rồi tính:

$$k = \mathrm{tg}\alpha = \frac{\Delta h}{\Delta t}$$

Vấn đề còn lại là tìm T và để làm được việc này ta chuyển ngược $H(s) = \dfrac{G(s)}{s}$ sang miền thời gian:

$$h(t) = k[t - T(1 - e^{-\frac{t}{T}})]$$

Đường tiệm cận với $h(t)$ khi $t\to\infty$ phải là đường thẳng có sai lệch so với $h(t)$ tiến tới 0, mặt khác khi $t\to\infty$ thì $e^{-\frac{t}{T}} \to 0$ nên mô hình đường tiệm cận sẽ chính là:

$$h_{tc}(t) = k\,(t - T)\,.$$

Suy ra T là giao điểm của đường tiệm cận $h_{tc}(t)$ với trục hoành. Ta đi đến thuật toán:

1) Kẻ đường tiệm cận $h_{tc}(t)$ với $h(t)$ tại $t=\infty$.
2) Giao điểm của đường tiệm cận $h_{tc}(t)$ với trục thời gian (trục hoành) chính là tham số T của mô hình (3.6).
3) Lấy một đoạn bất kỳ của đường tiệm cận, chiếu nó lên hai trục tọa độ để có Δh và Δt rồi tính $k = \dfrac{\Delta h}{\Delta t}$.

Ở những trường hợp có tín hiệu đầu vào không phải là hàm $1(t)$ mà là $u(t)=u_0 1(t)$ thì với tính chất tuyến tính của đối tượng, tín hiệu đầu ra sẽ là $y(t)= u_0 h(t)$. Thuật toán tương ứng cho các trường hợp như vậy có dạng như sau:

1) Kẻ đường tiệm cận $y_{tc}(t)$ với $y(t)$ tại $t=\infty$.
2) Giao điểm của đường tiệm cận $y_{tc}(t)$ với trục thời gian chính là tham số T.
3) Lấy một đoạn bất kỳ của đường tiệm cận $y_{tc}(t)$, chiếu nó lên hai trục tọa độ để có Δy và Δt rồi tính $k = \dfrac{\Delta y}{u_0 \Delta t}$.

Ví dụ 1: Để minh họa cho thuật toán trên ta xét bài toán nhận dạng mô hình động cơ chỉnh vị trí với tín hiệu vào là điện áp $u(t)$ tính theo *Volt* và tín hiệu ra là quãng đường bàn trượt đi được $y(t)$ tính theo *cm* (hình 3.15 bên trái).

Chủ động kích thích đối tượng bằng điện áp $u(t)$=50V tại đầu vào ta thu được quãng đường bàn trượt đi được $y(t)$ như sau:

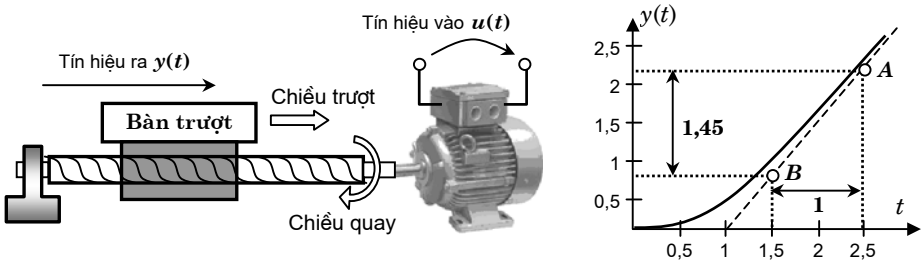
t [giây s]	0	0,5	1	1,5	2	2,5	3	3,5	4	4,5
$y(t)$ [cm]	0	0,15	0,35	0,8	1,65	2,2	2,7	3,35	3,9	4,45

Biểu diễn kết quả đo được dưới dạng đồ thị ta có hình 3.15 bên phải. Từ đường đồ thị đó của $y(t)$ và sau khi kẻ đường tiệm cận với $y(t)$ khi $t\to\infty$ ta được T=1s là hoành độ giao điểm của đường tiệm cận với trục thời gian. Lấy hai điểm A, B bất kỳ trên đường tiệm cận rồi chiếu đoạn $\overline{AB}$ lên hai trục tọa độ ta có Δy=1,45cm và Δt=1s. Suy ra:

$$k = \frac{1{,}45cm}{50V \cdot 1s} \approx 0{,}03\,\frac{cm}{V \cdot s}$$

Vậy mô hình tham số của động cơ chỉnh vị trí là:

$$G(s) = \frac{0{,}03}{s(1+s)}.$$ ❑



Hình 3.15: Đối tượng nhận dạng là động cơ chỉnh vị trí.

Xác định tham số mô hình tích phân quán tính bậc cao

Thuật toán xác định các tham số k và T của mô hình (3.6) có thể được mở rộng cho cả trường hợp mô hình tham số $\mathrm{IT_n}$ của đối tượng dạng (3.7), tức là

$$G(s) = \frac{k}{s(1+Ts)^n},$$

trong đó n có thể là cho trước hoặc có thể là một tham số mô hình cần phải xác định. Đặc biệt mô hình (3.7) này cũng thỏa mãn các sơ kiện ứng với dạng hàm quá độ $h(t)$ cho

trong hình 3.14. Bởi vậy thuật toán xác định tham số k, T, n mô hình (3.7) sẽ là sự quan tâm tiếp theo của ta.

Đặt tín hiệu $u(t)=u_0 1(t)$ tại đầu vào, ảnh Laplace $Y(s)$ của tín hiệu ra $y(t)$ sẽ được suy ra từ (3.7) như sau

$$Y(s) = \frac{u_0 k}{s^2 (1+Ts)^n}. \tag{3.8}$$

Phân tích (3.8) thành tổng các phân thức tối giản

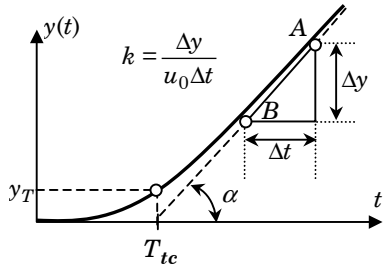
$$Y(s) = u_0 k \left(\frac{1}{s^2} - \frac{nT}{s} + \sum_{i=1}^{n} \frac{(n+1-i)T^2}{(1+Ts)^i} \right)$$

rồi áp dụng công thức biến đổi Laplace ngược

$$\mathcal{L}^{-1}\{ \frac{1}{(1+Ts)^i} \} = \frac{t^{i-1} e^{-\frac{t}{T}}}{T^i (i-1)!}$$

ta có đáp ứng đầu ra của IT_n:

$$y(t) = u_0 k \cdot \left(t - nT + \sum_{i=1}^{n} \frac{(n+1-i) t^{i-1} e^{-\frac{t}{T}}}{T^{i-2}(i-1)!} \right).$$



Hình 3.16: Hàm quá độ của khâu $\mathbf{IT_n}$.

Với t rất lớn thì tất cả các thành phần $e^{-\frac{t}{T}}$ trong tổng trên có thể được xem như bằng 0. Do đó đường thẳng $y_{tc}(t)$ tiệm cận với $y(t)$ tại $t \to \infty$ sẽ là

$$y_{tc}(t) = u_0 k (t - nT)$$

Đường tiệm cận có hệ số góc $u_0 k$ nên để tìm k ta lấy một đoạn bất kỳ của nó, chiếu lên hai trục tọa độ để có Δy và Δt. Hệ số khuếch đại k khi đó được tính bằng

$$k = \frac{tg\alpha}{u_0} = \frac{\Delta y}{u_0 \Delta t} \tag{3.9}$$

Giao điểm của đường tiệm cận $y_{tc}(t)$ với trục hoành là $T_{tc}=nT$ (hình 3.16). Nếu như n là cho trước, ta có thể tính ngay ra được tham số T bằng cách chia T_{tc} cho n. Ta đi đến thuật toán xác định hai tham số k và T cho mô hình (3.7) từ đường thực nghiệm $y(t)$ khi biết truốc n như sau:

a) Dựng đường tiệm cận $y_{tc}(t)$ của $y(t)$ khi $t\to\infty$. Lấy một đoạn $\overline{AB}$ bất kỳ trên $y_{tc}(t)$ sau đó chiếu lên hai trục tọa độ để có Δy và Δt. Hệ số khuếch đại k sẽ được tính từ Δy và Δt theo (3.9).

b) Xác định T_{tc} là giao điểm của đường tiệm cận $y_{tc}(t)$ với trục thời gian.

c) Tính $T=\dfrac{T_{tc}}{n}$.

Trong trường hợp n không biết trước thì trước tiên ta phải xác định n. Để làm được điều này ta tính giá trị đáp ứng $y(t)$ tại thời điểm T_{tc} :

$$y_T=y(T_{tc}) \ =u_0ke^{-n}T\sum_{i=1}^{n}\frac{(n+1-i)n^{i-1}}{(i-1)!}$$

Sau đó lập tỷ số

$$\varphi=\frac{y_T}{u_0kT_{tc}}=\frac{y_T}{u_0knT}=\frac{e^{-n}}{n^2}\sum_{i=1}^{n}\frac{(n+1-i)n^{i}}{(i-1)!}=f_0(n) \qquad (3.10)$$

rồi biểu diễn quan hệ giữa φ và n dưới dạng bảng tra (bảng 3.1) nhằm phục vụ công việc xác định ngược n từ φ sau này được thuận tiện.

Bảng 3.1: Bảng tra cứu ***n*** từ giá trị ***φ***.

n	1	2	3	4	5	6	7	8	9	10
φ	0,3679	0,2707	0,224	0,1954	0,1755	0,1606	0,149	0,1396	0,1318	0,1144

Bảng 3.1 trên được thiết lập bởi hàm **phi_n()** thực hiện công thức (3.10) viết trên ngôn ngữ lập trình C cho dưới đây. Hàm **phi_n()** có biến hình thức **n** chứa giá trị đầu vào n và trả về giá trị φ tính được:

```
double phi_n(int n)
{
   int i,j,mau_so;
   double phi,sum=0.,tu_so=1./n;
   for(i=1;i<=n;i++)
   {
      mau_so = 1;
      for(j=2;j<i;j++) mau_so=mau_so*j;
      sum=sum+(tu_so*(n+1-i))/(float)mau_so;
```

```
        tu_so=tu_so*n;
    }
    phi=sum*exp(-n);
    return(phi);
}
```

Với bảng 3.1, ta thấy chỉ có thể xác định được n=10 (ứng với giá trị φ=0,1144) là cao nhất. Tất nhiên là ta có thể mở rộng bảng 3.3 với sự trợ giúp của hàm **phi_n()**. Nhưng tổng quát hơn cả là xây dựng hàm giải ngược phương trình (3.10) xác định n từ φ. Để thực hiện điều này ta dựa vào tính chất nghịch biến của (3.10) và đi đến:

- Xuất phát với k=1 và tính $f_0(k)$ theo (3.10).
- Nếu $f_0(k)>\varphi$ thì tiếp tục tăng k cho tới khi có được $f_0(k)\le\varphi$.
- Trong trường hợp $2\varphi > [f_0(k-1)+ f_0(k)]$ thì $n= k-1$, ngược lại thì $n=k$.

Theo các bước tính như trên thì số nguyên n có được sẽ làm cho sai lệch $|f_0(n) - \varphi|$ là nhỏ nhất. Một hàm **n_phi()** viết trên C phục vụ việc xác định n từ φ theo các bước vừa nêu cho dưới đây để tham khảo. Biến hình thức **phi** của hàm chứa giá trị đầu vào φ và sẽ trả về số nguyên n tìm được.

```
int n_phi(double phi)
{
    int n=1;
    double save=0.3679, result;
    do
    {   n=n+1;
        result=save;
        save= phi_n(n);
    } while (save>phi);
    save=(save+result)/2.;
    if (save<phi) return(n-1); else return (n);
}
```

Chú ý: Giá trị đầu vào φ thích hợp cho hàm **n_phi()** phải thỏa mãn $\varphi \le 0{,}3679$.

Cùng với bảng 3.1 hoặc hàm **n_phi()** ta đi đến thuật toán xác định tham số k, n và T cho mô hình (3.7) từ đường thực nghiệm $y(t)$ như sau:

a) Dựng đường tiệm cận $y_{tc}(t)$ của $y(t)$ khi $t\to\infty$. Lấy một đoạn $\overline{AB}$ bất kỳ trên $y_{tc}(t)$ sau đó chiếu lên hai trục tọa độ để có Δy và Δt. Hệ số khuếch đại k sẽ được tính từ Δy và Δt theo $k = \dfrac{\Delta y}{u_0 \Delta t}$.

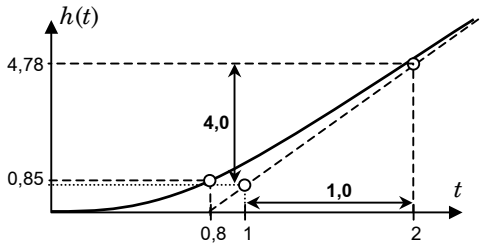
b) Xác định giao điểm T_{tc} của $y_{tc}(t)$ với trục thời gian và giá trị $y_T = y(T_{tc})$ từ đường thực nghiệm.

c) Tính $\varphi = \dfrac{y_T}{u_0 k T_{tc}}$

d) Xác định n theo φ từ bảng 3.1 hoặc sử dụng hàm **`n_phi()`**.

e) Tính $T = \dfrac{T_{tc}}{n}$.

Ví dụ 2: Giả sử bằng phương pháp nhận dạng bị động với việc đo cả tín hiệu vào $u(t)$ và ra $y(t)$ ta đã có dãy giá trị $G(jn\Omega_\lambda)$ của đường đặc tính tần của một đối tượng tuyến tính cần nhận dạng, trong đó $\Omega_\lambda = 0{,}061359375s^{-1}$ và n = 0, 1, 2 … , λ–1 với λ=1024. Lấy chỉ số Lag M=100 rồi chuyển ngược dãy $\{G(jn\Omega_\lambda)\}$, n=0, 1, 2 … , 200 sang miền thời gian ta có được dãy gồm 200 giá trị hàm trọng lượng $g_k = g(kT_a)$ với $T_a = \dfrac{2\pi}{\lambda\Omega_\lambda} = 0{,}1s$ và k=0, 1, 2 … , 200. Tiếp theo ta xác định hàm quá độ theo công thức $h(lT_a) = T_a \sum_{k=0}^{l} g_k$, l=0,1, … , 200 rồi biểu diễn các giá trị $h(lT_a)$ dưới dạng đồ thị ta có hình 3.17.



Hình 3.17: Hình minh họa cho ví dụ 2.

Đường đáp ứng $h(t)$ ứng với kích thích đầu vào $1(t)$ nên ở đây có u_0 =1. Từ đồ thị $h(t)$ và đường tiệm cận của nó khi $t \to \infty$ ta đọc ra được T_{tc} = 0,8 ; y_T = 0,85 ; Δy = 4 và Δt =1. Do đó

$$k = \frac{\Delta y}{u_0 \Delta t} = 4 \quad \text{và} \qquad \varphi = \frac{y_T}{u_0 k T_{tc}} = 0{,}266.$$

Tra bảng 3.1 ta được n =2 (được chọn ứng với giá trị φ gần nhất là φ =0,2707). Ta cũng có thể sử dụng hàm **`n_phi()`** với đầu vào φ=0,66, hàm sẽ trả về giá trị n =2. Vậy:

$$T = \frac{T_{tc}}{n} = \frac{0{,}8}{2} = 0{,}4.$$

và mô hình tham số của đối tượng là

$$G(s) = \frac{4}{s(1+0{,}4s)^2}.$$ ❑

3.1.4 Xác định tham số mô hình quán tính bậc cao

Ngay ở chương 1 ta đã mở đầu với một ví dụ về việc nhận dạng mô hình động cơ một chiều bằng phương pháp chủ động kích thích ở đầu vào tín hiệu 1(t) rồi thu được ở đầu ra đường thực nghiệm $h(t)$ có dạng mô tả trong hình 3.18.

Trong thực tế có khá nhiều đối tượng có đường thực nghiệm $h(t)$ giống như vậy. Chúng đều có chung những đặc điểm sau:

a) $h(t)$ xuất phát từ 0 và có đạo hàm tại đó cũng bằng 0. Theo kết luận 1 của mục 3.1.1 thì mô hình (3.1) tương ứng của đối tượng phải có $n_a - n_b \geq 2$.

b) Đường thực nghiệm $h(t)$ có giá trị hằng số k khi $t \to \infty$ nên cũng theo kết luận 1 hàm truyền đạt $G(s)$ phải thỏa mãn $G(0) = \dfrac{b_0}{a_0} = k$.

c) Đường $h(t)$ không có dạng lượn sóng, có tốc độ không âm (không có độ quá điều chỉnh) và giá trị luôn tăng dần đến $h(\infty)=k$ nên theo kết luận 2, các hằng số thời gian của tử số cũng như của mẫu số phải là những số thực thỏa mãn (3.3).

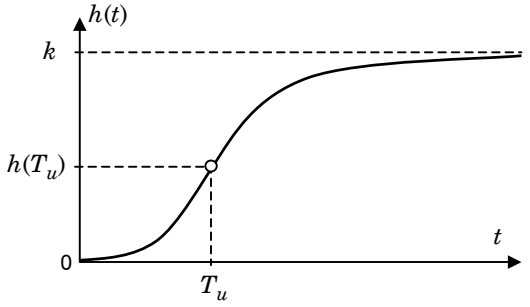
Như vậy đủ để có thể mô tả đối tượng nếu ta sử dụng mô hình (3.1) có cấu trúc:

$$G(s) = \frac{k}{(1+T_1 s)(1+T_2 s)} \qquad \text{với} \qquad T_1 \neq T_2 \qquad \text{(mô hình PT}_2\text{)} \tag{3.11}$$

hoặc

$$G(s) = \frac{k}{(1+Ts)^n} \qquad \text{(mô hình PT}_\text{n}\text{)} \tag{3.12}$$

Vấn đề tiếp theo đặt ra cho bài toán nhận dạng là từ đường đồ thị của $h(t)$ ta phải xác định loại mô hình nào trong số (3.11), (3.12) thích hợp với đối tượng, cũng như các tham số k, T_1, T_2 hoặc k, T, n cho mô hình đó.



Hình 3.18: Hàm quá độ của khâu **PT$_2$** và **PT$_\text{n}$**.

Trong khi tham số k của cả hai mô hình có thể đọc được ngay một cách trực tiếp từ $h(t)$ theo công thức:

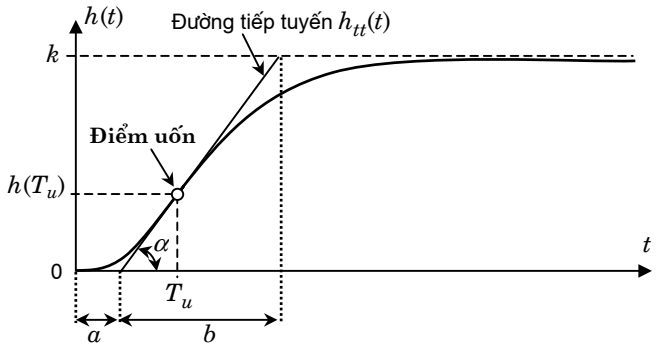
$$k = \lim_{t\to\infty} h(t) = h(\infty)$$

hay k chính là tung độ của giao điểm đường tiệm cận của $h(t)$ khi $t\to\infty$, thì việc xác định những tham số còn lại (T_1, T_2 hoặc T, n) có phần phức tạp hơn và chúng sẽ được xác định dựa vào hai giá trị a, b (hình 3.19) có từ đồ thị $h(t)$.

Đường thực nghiệm của $h(t)$ có một đặc điểm mấu chốt mà ta sẽ sử dụng để xác định T_1, T_2 hoặc T, n từ a, b là trong khoảng thời gian $0\le t<T_u$ (hình 3.19) $h(t)$ có tốc độ tăng dần (gia tốc dương), sau đó là giảm dần vận tốc về 0 khi $t\to\infty$ (gia tốc âm), nói cách khác $h(t)$ có vận tốc cực đại

$$v^* = \max_t \frac{dh(t)}{dt}$$

tại thời điểm T_u và như vậy tại T_u đường cong $h(t)$ có điểm uốn.



Hình 3.19: Xác định hai tham số ***a*** và ***b*** từ ***h*(*t*)**.

Kẻ đường tiếp tuyến $h_{tt}(t)$ của $h(t)$ tại T_u thì phương trình đường tiếp tuyến đó sẽ là

$$h_{tt}(t) = \mathrm{tg}\alpha(t-a), \tag{3.13}$$

trong đó a được định nghĩa là hoành độ giao điểm của đường tiếp tuyến của $h(t)$ tại điểm uốn. Gọi b là khoảng thời gian để đường tiếp tuyến đó đi được từ 0 tới k ta có:

$$v^* = \mathrm{tg}\alpha = \frac{k}{b} \quad\Rightarrow\quad b = \frac{k}{v^*} \tag{3.14}$$

$$a = T_u - \frac{h(T_u)}{\mathrm{tg}\alpha} = T_u - \frac{h(T_u)}{v^*} \tag{3.15}$$

Xác định tham số mô hình PT$_2$ ($T_1 \neq T_2$)

Không mất tính tổng quát nếu ta giả thiết $T_1 > T_2$, như vậy thì từ (3.11) ta có mô hình của đường thực nghiệm $h(t)$ bằng phép biến đổi ngược toán tử Laplace như sau:

$$h(t) = k \cdot \left[1 - \frac{T_1 e^{-\frac{t}{T_1}} - T_2 e^{-\frac{t}{T_2}}}{T_1 - T_2} \right] \tag{3.16}$$

Suy ra:

$$\frac{dh(t)}{dt} = k \frac{e^{-\frac{t}{T_1}} - e^{-\frac{t}{T_2}}}{T_1 - T_2} \tag{3.17}$$

và $$\frac{d^2h(t)}{dt^2} = k \cdot \left[\frac{e^{-\frac{t}{T_1}}}{T_1(T_2 - T_1)} + \frac{e^{-\frac{t}{T_2}}}{T_2(T_1 - T_2)} \right]. \tag{3.18}$$

Bởi vậy để có T_u ta xác định thời điểm mà $\frac{d^2h(t)}{dt^2}$ bị triệt tiêu và đi đến

$$T_u = \frac{T_1 T_2}{T_2 - T_1} \ln \frac{T_2}{T_1} \tag{3.19}$$

Thay (3.19) vào (3.17) có

$$v^* = \frac{dh(T_u)}{dt} = \frac{k}{T_1} \left(\frac{T_2}{T_1} \right)^{\frac{T_2}{T_1 - T_2}} \tag{3.20}$$

và thay (3.19), (3.16) vào (3.15) được

$$a = T_u - b + T_1 + T_2 \tag{3.21}$$

Do đó nếu ta ký hiệu

$$x = \frac{T_2}{T_1} \qquad (0 < x < 1 \text{ vì } T_1 > T_2). \tag{3.22}$$

thì cùng với (3.14), (3.19) ta sẽ được

$$\frac{b}{T_1} = \frac{k}{T_1 v^*} = \left(\frac{T_2}{T_1} \right)^{\frac{T_2}{T_2 - T_1}} = x^{\frac{x}{x-1}} = f_1(x). \tag{3.23}$$

và với (3.21), (3.21) có

$$\frac{a}{b}=\frac{v^*(kT_u-bv^*)}{k^2}=x^{\frac{x}{1-x}}\frac{x\ln x+x^2-1}{x-1}-1=f_2(x) \tag{3.24}$$

Ba công thức (3.22), (3.23) và (3.24) chính là công cụ để tìm T_1, T_2 từ a và b. Cụ thể như sau:

- Tìm x thỏa mãn $0<x<1$ từ $\frac{a}{b}$ bằng cách giải ngược (3.24), tức là $x=f_2^{-1}\left(\frac{a}{b}\right)$
- Tìm T_1 từ x theo (3.23), tức là $T_1=\frac{b}{f_1(x)}$
- Tìm T_2 theo (3.22), tức là $T_2=xT_1$

Nhưng phải chú ý rằng các bước xác định T_1, T_2 từ a và b trên đây không phải lúc nào cũng áp dụng được. Tại sao lại như vậy?. Câu trả lời nằm ở ngay điều kiện là giá trị x tìm được ở bước đầu tiên phải thỏa mãn $0<x<1$. Nếu như rằng giá trị x tìm được ở bước đầu tiên nằm ngoài khoảng $(0,1)$ thì ta có thể dừng ngay công việc tính toán và đưa ra kết luận rằng đối tượng không thể mô tả bởi mô hình PT_2.

Hơn nữa, hàm $f_2(x)$ định nghĩa theo (3.24) là hàm đồng biến (phần chứng minh dành cho bài tập) với giá trị giới hạn

$$\lim_{x\to 1}f_2(x)=\lim_{x\to 1}\left(x^{\frac{x}{1-x}}\frac{x\ln x+x^2-1}{x-1}-1\right)=0{,}103648$$

nên ta có thể kiểm tra điều kiện thực hiện thuật toán mà không cần giải ngược phương trình (3.24) bằng cách kiểm tra xem tỷ số $\frac{a}{b}$ có nằm trong khoảng

$$0<\frac{a}{b}<0{,}103648 \tag{3.25}$$

hay không.

Tổng kết lại tất cả các kết quả nêu trên, ta có được thuật toán xác định các tham số k, T_1, T_2 của mô hình (3.11) từ đường thực nghiệm $h(t)$ như sau:

1) Tìm hằng số k theo $k=h(\infty)$.
2) Kẻ đường tiếp tuyến $h_{tt}(t)$ với $h(t)$ tại điểm uốn. Sau đó xác định hai tham số a là hoành độ giao điểm của đường tiếp tuyến $h_{tt}(t)$ với trục thời gian và b là khoảng thời gian để đường tiếp tuyến đó đi được từ 0 tới k.
3) Lập tỷ số $\frac{a}{b}$. Nếu tỷ số này nằm ngoài khoảng (3.25), tức là $\frac{a}{b}\geq 0{,}103648$, thì dừng thuật toán với kết luận rằng đối tượng không mô tả được bằng mô hình (3.11).

4) Tìm x thỏa mãn $0<x<1$ từ $\frac{a}{b}$ bằng cách giải ngược (3.24), tức là $x = f_2^{-1}\left(\frac{a}{b}\right)$.

5) Tìm T_1 từ x theo (3.23), tức là $T_1 = \frac{b}{f_1(x)}$.

6) Tìm T_2 theo (3.22), tức là $T_2 = xT_1$.

Dựa vào tính chất là $f_2(x) = \frac{a}{b}$ đơn điệu tăng trong khoảng $0<x<1$, một hàm viết trên ngôn ngữ lập trình C có tên **x_adivb()** theo nguyên tắc chia đôi khoảng nghiệm sau mỗi lần tính cho dưới đây để phục vụ việc xác định ngược $x = f_2^{-1}\left(\frac{a}{b}\right)$ như bước 4) yêu cầu. Hàm này có biến hình thức **adivb** chứa giá trị đầu vào là tỷ số $\frac{a}{b}$ và trả về giá trị x tính được với độ chính xác 10^{-5}.

```
double x_adivb(double adivb)
{
   double x1=0., x2=1.,f2,x,mu,err=0.00001;
   do {
      x=(x1+x2)/2.;
      mu=pow(x,(x/(1.-x)));
      f2=(mu*(x*log(x)+x*x-1.)/(x-1.))-1.;
      if (adivb<f2) x2=x; else x1=x;
   } while(fabs(f2-adivb)>err);
   return(x);
}
```

Bảng 3.2 dưới đây được thiết lập nhờ trợ giúp của hàm **x_adivb()** biểu diễn một vài giá trị x được tính ngược từ những tỷ số $\frac{a}{b}$ đặc trưng.

Bảng 3.2: Một số giá trị ***x*** có từ những giá trị ***a/b*** tương ứng.

a/b	0,01	0,02	0,03	0,04	0,05	0,06	0,07	0,08	0,09	0,1
x	0,012	0,0275	0,0467	0,707	0,1008	0,1393	0,1904	0,2622	0,3740	0,6113

Nếu tín hiệu kích thích chủ động ở đầu vào của đối tượng không phải là hàm Heaviside $1(t)$ mà là $u(t)=u_0 1(t)$ và tín hiệu đo được ở đầu ra là $y(t)$ thì do tính tuyến tính của đối tượng, hàm quá độ $h(t)$ sẽ là

$$h(t) = \frac{y(t)}{u_0}. \tag{3.26}$$

Do đó, cùng với (3.26) thuật toán trên cũng có một sửa đổi nhỏ cho phù hợp với bài toán xác định các tham số k, T_1, T_2 của mô hình (3.11) từ đường thực nghiệm $y(t)$ như sau

1) Tìm hằng số k theo $k = \dfrac{y(\infty)}{u_0}$.

2) Kẻ đường tiếp tuyến $y_{tt}(t)$ với $y(t)$ tại điểm uốn. Sau đó xác định hai tham số a là hoành độ điểm mà tại đó có $y_{tt}(t)=0$ của đường tiếp tuyến và b là khoảng thời gian để đường tiếp tuyến đó đi được từ 0 tới $y(\infty)$.

3) Lập tỷ số $\dfrac{a}{b}$. Nếu $\dfrac{a}{b} \geq 0{,}103648$, thì dừng thuật toán với kết luận rằng đối tượng không mô tả được bằng mô hình (3.11).

4) Tìm $x = f_2^{-1}\left(\dfrac{a}{b}\right)$ bằng cách giải ngược (3.24).

5) Tìm T_1 từ x theo (3.23), tức là $T_1 = \dfrac{b}{f_1(x)}$.

6) Tìm T_2 theo (3.22), tức là $T_2 = xT_1$.

Ví dụ 1: Giả sử ta phải xác định mô hình có tham số dạng (3.1) cho bộ cảm biến chuyển đổi áp suất thành độ dịch chuyển như hình 3.20 bên trái mô tả. Kích thích với đầu vào $u(t)$=1,5bar người ta đo được ở đầu ra đường thực nghiệm $y(t)$ cho ở hình 3.20 bên phải.

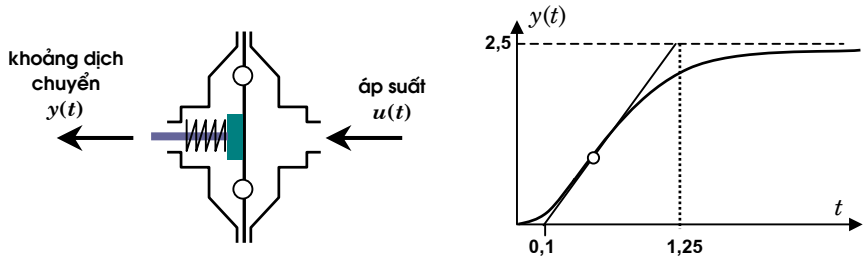
Từ đường thực nghiệm $y(t)$ và sau khi kẻ đường tiếp tuyến $y_{tt}(t)$ với nó tại điểm uốn ta có

$$k = \frac{2{,}5}{1{,}5} = 1{,}7, \qquad a = 0{,}1 \qquad \text{và} \qquad b = 1{,}15.$$

Suy ra

$$\frac{a}{b} = 0{,}087 < 0{,}103648$$

và do đó ta áp dụng được thuật toán trên để tìm T_1, T_2. Gọi hàm **`x_adivb()`** với giá trị **`adivb`** =0,087 ở đầu vào ta có x = 0,334.



Hình 3.20: Xác định mô hình bộ cảm biến chuyển đổi áp suất/độ dịch chuyển.

Vậy $T_1 = \dfrac{b}{f_1(x)} \approx 0{,}6638$, $T_2 = xT_1 \approx 0{,}2214$ và mô hình tham số của bộ cảm biến là:

$$G(s) = \frac{1{,}7}{(1+0{,}6638s)(1+0{,}2214s)} .$$ ❑

Các bước 3), 4), 5) và 6) của thuật toán xác định tham số mô hình PT_2 trên đã được cài đặt thành hàm **pt2()** với 4 biến hình thức **a**, **b**, **T1** và **T2** được viết trên ngôn ngữ lập trình C cho dưới đây để tham khảo, trong đó hai giá trị đầu vào a, b là nội dung của hai biến hình thức **a**, **b** và hai tham số đầu ra T_1, T_2 tính được sẽ được hàm cất giữ trong các biến hình thức **T1**, **T2** .

Hàm **pt2()** có sử dụng thêm hàm **x_adivb()** để tính ngược x từ $\dfrac{a}{b}$ theo công thức $x = f_2^{-1}\left(\dfrac{a}{b}\right)$. Giá trị trả về của **pt2()** là một số nguyên. Nếu **pt2()** trả về giá trị 0 thì đối tượng mô tả được bởi mô hình PT_2 , trường hợp ngược lại thì không.

```
int pt2(double a, double b, double &T1, double &T2)
{
   double adivb, x;
   adivb=a/b;
   if ((0<adivb)&&(adivb<0.103648))
   {
      x=x_adivb(adivb);
      T1=b/pow(x,(x/(x-1.)));
      T2=x*T1;
      return 0;
   } else return 1;
}
```

Ví dụ 2: Giả sử từ đường thực nghiệm $y(t)$ người ta đã xác định được a=2,5 và b=30. Gọi chương trình **pt2()** bằng lệnh

```
void main()
{
   double a=2.5,b=30,T1,T2;
   if (pt2(a,b,T1,T2)== 0) printf("T1=%1.4f\tT2=%1.4f\n",T1,T2);
   getch();
}
```

ta sẽ nhận được T_1 = 18,0312 và T_2 = 5,2829. ❑

Xác định tham số mô hình quán tính bậc cao

Ở thuật toán trên có một điều kiện ràng buộc là tỷ số $\frac{a}{b}$ phải thỏa mãn (3.25) và nếu điều đó không xảy ra người ta phải nghĩ tới một mô hình khác. Mô hình đầu tiên có nhiều khả năng thích hợp là mô hình PT_n (3.12) với n là một tham số cần phải xác định.

Xuất phát từ (3.12) với công thức biến đổi Laplace ngược sang miền thời gian sẽ có

$$h(t) = k \cdot \left[1 - e^{-\frac{t}{T}} \sum_{i=0}^{n-1} \frac{\left(\frac{t}{T}\right)^i}{i!} \right] \tag{3.27}$$

và như vậy ta được

$$\frac{dh(t)}{dt} = \frac{ke^{-\frac{t}{T}}}{tT} \cdot \left[t \sum_{i=0}^{n-1} \frac{\left(\frac{t}{T}\right)^i}{i!} - T \sum_{i=0}^{n-1} \frac{\left(\frac{t}{T}\right)^i}{(i-1)!} \right] \tag{3.28}$$

$$\frac{d^2h(t)}{dt^2} = \frac{ke^{-\frac{t}{T}}}{t^2T^2} \cdot \sum_{i=0}^{n-1} \frac{\left(\frac{t}{T}\right)^i iT(T+2t) - \left(\frac{t}{T}\right)^i i^2T^2 - \left(\frac{t}{T}\right)^i t^2}{i!} \tag{3.29}$$

Đặt $\frac{d^2h(t)}{dt^2} = 0$ và giải ra để tìm T_u ta đi đến

$$T_u = (n-1)T \tag{3.30}$$

Thay (3.30) vào (3.28) và (3.27) có

$$v^* = \frac{dh(T_u)}{dt} = k \cdot \frac{e^{1-n}(n-1)^{n-2}}{T(n-2)!} \tag{3.31}$$

$$h(T_u) = k \cdot \left[e^{n-1} - \sum_{i=0}^{n-1} \frac{(n-1)^i}{i!} \right] \tag{3.32}$$

Thay tiếp (3.31) vào (3.14)

$$b = \frac{(n-2)!}{(n-1)^{n-2}} e^{n-1} T \tag{3.33}$$

và (3.30), (3.31), (3.32) vào (3.15)

$$a = (n-1)T - \frac{T(n-2)!}{(n-1)^{n-2}} \cdot \left[e^{n-1} - \sum_{i=0}^{n-1} \frac{(n-1)^i}{i!} \right]$$

Suy ra

$$\frac{a}{b} = e^{1-n} \cdot \left[\frac{(n-1)^n}{(n-1)!} + \sum_{i=0}^{n-1} \frac{(n-1)^i}{i!} \right] - 1 = f_3(n). \tag{3.34}$$

Công thức (3.34) chính là công thức xác định tham số n cho mô hình (3.12) bằng cách giải ngược

$$n = \text{phần nguyên gần nhất của } f_3^{-1}\left(\frac{a}{b}\right). \tag{3.35}$$

và sau khi đã có n từ (3.35) ta cũng sẽ có nốt tham số T còn lại nhờ (3.33):

$$T = \frac{b(n-1)^{n-2}}{e^{n-1}(n-2)!}. \tag{3.36}$$

Do $n \in \mathbb{N}$ nên khi n tương đối nhỏ (3.35) có thể được thay thế bằng bảng tra cho đơn giản, tức là sau khi đã có tỷ số $\frac{a}{b}$ ta chỉ cần tra bảng để có n mà không cần giải ngược phương trình (3.35). Một bảng tra như vậy là bảng 3.3 cho dưới đây.

Bảng 3.3: Bảng tra giá trị n từ tỷ số a/b.

n	2	3	4	5	6	7	8	9	10	11
a/b	0,1036	0,218	0,3194	0,4103	0,4933	0,57	0,6417	0,7092	0,7732	0,8341

Bảng 3.3 được tạo lập bởi hàm **adivb_n()** với giá trị đầu vào n là nội dung của biến hình thức **n** và giá trị trả về là tỷ số $\frac{a}{b}$ viết trên C như sau:

```
double adivb_n(int n)
{
   int i;
   double tu_so,mau_so,sum;
   mau_so=1.;
   tu_so=n-1.;
   sum=1.;
   for (i=1;i<n;i++)
   {  mau_so=mau_so*i;
      sum=sum+(tu_so/mau_so);
      tu_so=tu_so*(n-1.);
   }
   return((exp(1.-n)*((tu_so/mau_so)+sum))-1.);
}
```

Tổng quát, một hàm khác viết trên C với tên **n_adivb()** cho sau đây có tác dụng thực hiện công thức (3.35) tính ngược n từ tỷ số $\frac{a}{b}$ mà không cần tra bảng 3.3. Giá trị

đầu vào của hàm $\frac{a}{b}$ là nội dung của biến hình thức **adivb** và hàm sẽ trả về giá trị n tính được có sai lệch $|f_3(n) - \frac{a}{b}|$ nhỏ nhất.

```
int n_adivb(double adivb)
{  int n=2;
   double save=0.1036, result;
   do
   {  n=n+1;
      result=save;
      save=adivb_n(n);
   } while (save<=adivb);
   save=(save+result)/2.;
   if (save<adivb) return(n); else return (n-1);
}
```

Chú ý: Gọi rằng tổng quát, song hàm **n_adivb()** cũng có nhược điểm là không thực hiện được khi $\frac{a}{b} > 4{,}75$ do tràn ô nhớ. Với $\frac{a}{b} = 4{,}75$ hàm sẽ trả về giá trị n=127. Nhưng một mô hình (3.12) với n>50 đã được xếp vào loại mô hình có cấu trúc phức tạp, ít có ý nghĩa thực tế nên nhược điểm này sẽ không hạn chế khả năng ứng dụng của hàm.

Ngoài ra, do đối tượng là tuyến tính nên các công thức (3.35), (3.36) cũng như hàm **n_adivb()** xác định n, T từ a, b không phụ thuộc vào tín hiệu chủ động kích thích đối tượng ở đầu vào $u(t)$ là $1(t)$ hay $u_0 1(t)$. Bởi vậy khi $u(t)=u_0 1(t)$, thuật toán xác định các tham số k, T và n từ đường thực nghiệm $y(t)$ đo được tại đầu ra sẽ có dạng như sau:

1) Tìm hằng số k theo $k = \frac{y(\infty)}{u_0}$.

2) Kẻ đường tiếp tuyến $y_{tt}(t)$ với $y(t)$ tại điểm uốn. Sau đó xác định hai tham số a là hoành độ giao điểm của đường tiếp tuyến $y_{tt}(t)$ với trục thời gian và b là khoảng thời gian để đường tiếp tuyến đó đi được từ 0 tới $y(\infty)$.

3) Lập tỷ số $\frac{a}{b}$. Nếu $\frac{a}{b} < 0{,}103648$, thì dừng thuật toán với kết luận rằng đối tượng phải được mô tả bằng mô hình (3.11). Ta cũng dừng thuật toán khi $\frac{a}{b} > 4{,}75$.

4) Tìm n bằng cách tra bảng 3.3 hoặc nhờ hàm **n_adivb()**.

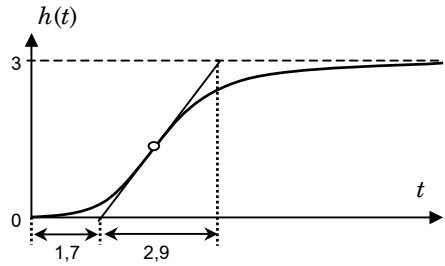
5) Tìm T từ n và b theo (3.36).

Một phần thuật toán trên, cụ thể là các bước 3), 4), 5) nhằm xác định tham số n, T cho mô hình PT_n đã được cài đặt thành hàm **ptn()** với 4 biến hình thức **a**, **b**, **n** và **T** viết trên C cho dưới đây để tham khảo. Hai giá trị đầu vào a, b là nội dung của **a**, **b**. Hai

tham số đầu ra n, T sẽ được hàm cất trong **n** và **T**. Giá trị trả về của **ptn()** là một số nguyên. Nếu **ptn()** trả về giá trị 0 thì đối tượng mô tả được bởi mô hình PT_n (3.12), 1 nếu đối tượng phải được mô tả bởi mô hình PT_2 (3.11) và 2 nếu đối tượng không thể mô tả bởi mô hình PT_n và PT_2 . Chỉ khi giá trị của hàm trả về bằng 0 thì kết quả n, T mà hàm tính được mới có ý nghĩa.

```
int ptn(double a, double b, int &n, double &T)
{
   double adivb=a/b;
   int k,p=0;
   if (adivb<0.103648) p=1;
   else if (adivb>4.75) p=2;
   else
   {  n=n_adivb(adivb);
      T=b*(double)(n-1)/exp(n-1);
      for (k=2;k<n-1;k++) T=T*(n-1)/k;
   }
   return (p);
}
```

Ví dụ 3: Giả sử khi kích thích bằng tín hiệu 1(t) tại đầu vào của một đối tượng cần nhận dạng người ta thu được ở đầu ra đường thực nghiệm $h(t)$ cho ở hình 3.21. Với đường thực nghiệm đó thì k =3 và sau khi kẻ đường tiếp tuyến tại điểm uốn ta có thêm a =1,7 ; b = 2,9.



Hình 3.21: Hàm quá độ của đối tượng trong ví dụ 3.

Gọi hàm **ptn(a,b,n,T)** với hai giá trị đầu vào trên sẽ nhận được n = 7, T = 0.4658. Vậy mô hình của đối tượng là

$$G(s) = \frac{3}{(1 + 0{,}4658s)^7}$$ ❑

3.1.5 Xác định tham số mô hình Lead/Lag

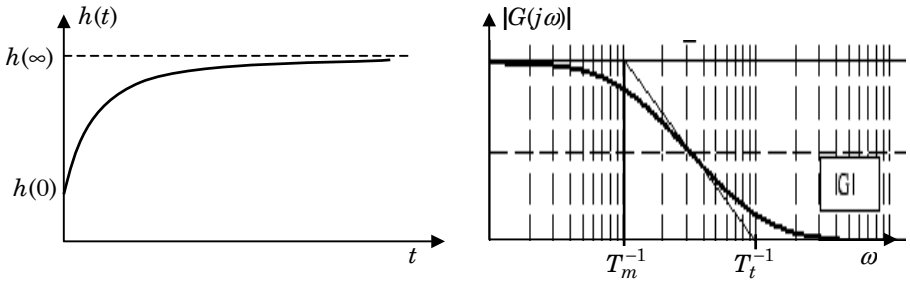
Nếu đường thực nghiệm $h(t)$ của một đối tượng có dạng giống hình 3.22 bên trái thì với các sơ kiện:

- $h(t)$ không xuất phát từ 0 và có đạo hàm tại 0 cũng khác không nên theo kết luận 1 trong mục 3.1.1, bậc của tử số và bậc của mẫu số của mô hình tham số (3.1) phải bằng nhau, tức là $n_a = n_b$,
- khi $t \to \infty$ hàm $h(t)$ tiến tới một hằng số $k \neq 0$ nên cũng theo kết luận 1, mô hình (3.1) phải thỏa mãn $k = \dfrac{b_0}{a_0}$,
- $h(t)$ không có dạng lượn sóng, luôn có xu hướng tăng dần tới k nên theo kết luận 2 các hằng số thời gian của đa thức tử số và mẫu số phải là những số thực và phải thỏa mãn các bất đẳng thức (3.3),

ta có thể mô tả đối tượng một cách tạm đủ bằng mô hình Lag (cắt bớt) như sau

$$G(s) = k\frac{1+T_t s}{1+T_m s}, \qquad T_t < T_m \tag{3.37}$$

Hình 3.22 bên phải là biểu đồ Bode của (3.37). Từ biểu đồ Bode đó ta thấy mô hình (3.37) không ưu tiên những thành phần tín hiệu có tần số cao khi đi qua nó, chính vì vậy mà mô hình (3.37) có tên là là mô hình Lag.



Hình 3.22: Hàm quá độ và biểu đồ Bode của mô hình Lag (cắt bớt).

Tương tự, trường hợp đường thực nghiệm $h(t)$ của một đối tượng có dạng như hình 3.23 bên trái, ta sẽ có các sơ kiện sau:

- vì $h(t)$ không xuất phát từ 0 và có đạo hàm tại 0 khác không nên $n_a = n_b$,
- do với $t \to \infty$ hàm $h(t)$ tiến tới một hằng số $k \neq 0$ nên $k = \dfrac{b_0}{a_0}$,

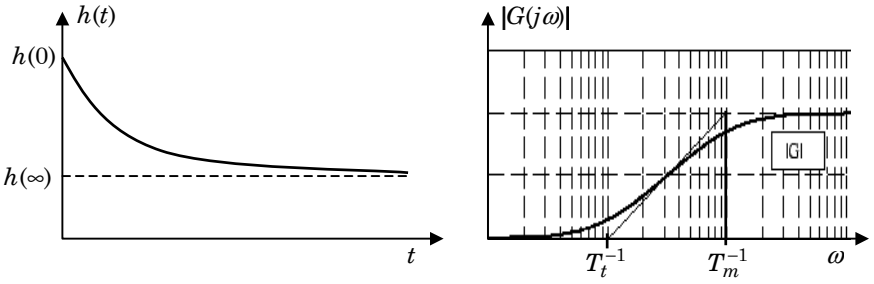
– $h(t)$ không có dạng lượn sóng, có một điểm cực đại tại t=0 sau đó giảm dần tới k nhưng không nhỏ hơn k nên theo kết luận 3 các hằng số thời gian của đa thức tử số và mẫu số phải là những số thực và phải có một bất đẳng thức trong (3.3) không được thỏa mãn.

Từ các sơ kiện nêu trên ta có thể đi đến nhận định rằng đối tượng sẽ mô tả được một cách tạm đủ bởi mô hình Lead (dẫn qua) như sau:

$$G(s) = k\frac{1+T_t s}{1+T_m s}, \qquad T_t > T_m \tag{3.38}$$

trong đó tên gọi Lead của (3.38) được suy ra từ tính chất là (3.38) sẽ ưu tiên các tín hiệu có tần số cao khi đi qua nó (xem biểu đồ Bode tương ứng trong hình 3.23 bên phải).

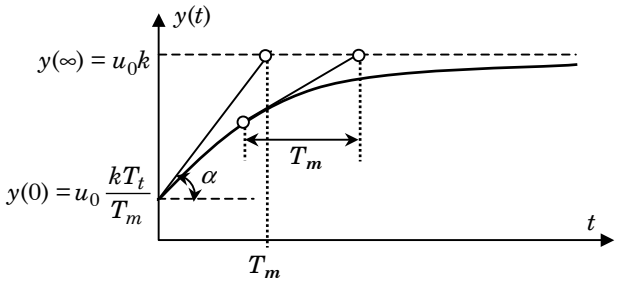
Vấn đề còn lại là xác định các tham số k, T_t và T_m cho mô hình.



Hình 3.23: Hàm quá độ và biểu đồ Bode của mô hình Lead (dẫn qua).

Xác định tham số mô hình Lag

Kích thích một cách chủ động tại đầu vào của đối tượng có thể mô tả bởi được mô hình Lag (3.37) bằng tín hiệu $u(t)=u_0 1(t)$ ta sẽ thu được ở đầu ra tín hiệu $y(t)$ có dạng cho trong hình 3.24.



Hình 3.24: Xác định tham số cho mô hình Lag (cắt bớt).

Nhiệm vụ bây giờ là phải xác định các tham số k, T_t và T_m cho mô hình từ đường thực nghiệm $y(t)$ thu được đó. Trong khi tham số k xác định ngay được bởi

$$k = \frac{1}{u_0} \lim_{t \to \infty} y(t) = \frac{y(\infty)}{u_0} \tag{3.39}$$

tức là bằng tỷ số của giá trị $y(\infty)$ và u_0 thì việc xác định hai tham số T_t , T_m còn lại có phần phức tạp hơn.

Trước hết ta có

$$Y(s) = \frac{u_0}{s} G(s) = u_0 \frac{k(1 + T_t s)}{s(1 + T_m s)} \tag{3.40}$$

cho nên

$$y(0) = \lim_{s \to \infty} sY(s) = u_0 \frac{kT_t}{T_m} \tag{3.41}$$

Kẻ tiếp đường tiếp tuyến $y_{tt}(t)$ với $y(t)$ tại điểm t=0 và gọi tgα là hệ số góc của đường tiếp tuyến đó thì (hình 3.24)

$$\text{tg}\alpha = \frac{dy(0)}{dt} = \lim_{s \to \infty} s\dot{Y}(s)$$

trong đó $\dot{Y}(s)$ là ký hiệu chỉ ảnh Laplace của $\frac{dy(t)}{dt}$. Với (3.40), (3.41) và công thức biến đổi (3.2c) ta được

$$\dot{Y}(s) = sY(s) - y(0) = u_0 \left(\frac{k(1 + T_t s)}{1 + T_m s} - \frac{kT_t}{T_m} \right)$$

Suy ra

$$\text{tg}\alpha = \frac{u_0 k - u_0 \frac{kT_t}{T_m}}{T_m} = \frac{y(\infty) - y(0)}{T_m}$$

Do đó T_m chính là hoành độ của giao điểm giữa đường tiếp tuyến $y_{tt}(t)$ với đường tiệm cận $y(\infty)$. Từ T_m ta cũng có luôn T_t nhờ (3.41) như sau

$$T_t = \frac{y(0) T_m}{u_0 k} \tag{3.42}$$

Ta đi đến thuật toán thứ nhất:

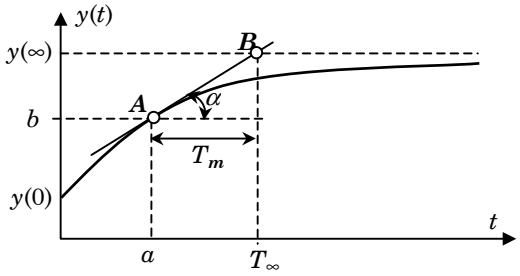
a) Kẻ đường tiệm cận $y(\infty)$ với $y(t)$ tại t=∞ rồi xác định k theo (3.39).

b) Kẻ đường tiếp tuyến $y_{tt}(t)$ với $y(t)$ tại t=0, sau đó xác định T_m là hoành độ của giao điểm giữa đường tiếp tuyến $y_{tt}(t)$ với đường tiệm cận $y(\infty)$.

c) Xác định T_t theo (3.42).

Thuật toán trên vướng lại vấn đề đã được đề cập khi nhận dạng tham số T cho mô hình PT_1 ở mục 3.1.2 là điểm $y(0)$ nơi bắt đầu đặt đường tiếp tuyến nằm khá xa đường tiệm cận $y(\infty)$. Điều này dẫn tới nguy cơ kết quả sẽ bị sai số khá lớn khi mà chỉ cần có một sai lệch nhỏ trong việc dựng đường tiếp tuyến.

Để tránh nguy cơ này người ta sẽ không dựng đường tiếp tuyến tại $y(0)$ nữa mà thay vào đó là tại một điểm khác nằm tương đối gần hơn so với đường tiệm cận $y(\infty)$, chẳng hạn điểm A có tọa độ $\begin{pmatrix} a \\ b \end{pmatrix}$ như hình 3.25 mô tả.



Hình 3.25: Xác định tham số T_m có sai số nhỏ cho mô hình Lag (cắt bớt).

Đường tiếp tuyến này sẽ có phương trình

$$y_{tt}(t) = \text{tg}\alpha\,(t-a)+b \tag{3.43}$$

Để xác định hệ số góc $\text{tg}\alpha$ ta đi từ ảnh Laplace $Y(s)$ của $y(t)$ theo công thức (3.40) và sau khi chuyển ngược sang miền thời gian sẽ có

$$y(t) = u_0 k(1 - \frac{T_m - T_t}{T_m} e^{-\frac{t}{T_m}})$$

$$\Rightarrow \qquad \text{tg}\alpha = \dot{y}(a) = u_0 k \frac{T_m - T_t}{T_m^2} e^{-\frac{a}{T_m}} \tag{3.44}$$

Gọi B là giao điểm của tiếp tuyến $y_{tt}(t)$ với đường tiệm cận $y(\infty)$ và T_∞ là hoành độ của B thì

$$y_{tt}(T_\infty) = \text{tg}\alpha(\,T_\infty - a)+b = y(\infty) = ku_0$$

do đó cùng với (3.43), (3.44) được

$$T_\infty - a = \frac{y(\infty)-y(a)}{\text{tg}\alpha} = \frac{T_m^2}{u_0 k(T_m - T_t)e^{-\frac{a}{T_m}}} \left[u_0 k - u_0 k(1 - \frac{T_m - T_t}{T_m} e^{-\frac{a}{T_m}}) \right] = T_m$$

Vậy T_m chính là khoảng thời gian cần thiết để $y_{tt}(t)$ đi được từ điểm A tới điểm B. Ta có được thuật toán thứ hai:

1) Kẻ đường tiệm cận $y(\infty)$ với $y(t)$ tại $t=\infty$ rồi xác định k theo (3.39).

2) Lấy một điểm A bất kỳ trên $y(t)$ và kẻ đường tiếp tuyến $y_{tt}(t)$ với $y(t)$ tại A, sau đó xác định B là giao điểm giữa đường tiếp tuyến $y_{tt}(t)$ với đường tiệm cận $y(\infty)$.

3) Chiếu đoạn $\overline{AB}$ lên trục thời gian (trục hoành) để có T_m.

4) Xác định T_t từ T_m và k theo (3.42).

Xác định tham số mô hình Lead

Về cơ bản, việc xác định các tham số k, T_t và T_m cho mô hình Lead cũng giống như cho mô hình Lag. Điểm khác biệt duy nhất là đường thực nghiệm $y(t)$ của mô hình Lead luôn nằm phía trên đường tiệm cận $y(\infty)$ nên phải có $T_t > T_m$ (hình 3.26).

Thuật toán tìm k, T_t và T_m từ đường thực nghiệm $y(t)$ có dạng như sau:

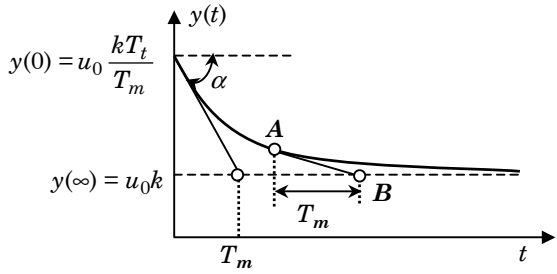
1) Xác định điểm $y(0)$.

2) Dựng đường tiệm cận $y(\infty)$ của $y(t)$ và tính $k = \dfrac{1}{u_0}\lim_{t\to\infty} y(t) = \dfrac{y(\infty)}{u_0}$

3) Kẻ đường tiếp tuyến $y_{tt}(t)$ với $y(t)$ tại một điểm A bất kỳ trên $y(t)$, sau đó xác định giao điểm B của $y_{tt}(t)$ với đường tiệm cận $y(\infty)$. Thông thường điểm A được chọn là điểm không nằm quá xa đường tiệm cận $y(\infty)$.

4) Tính T_m là hình chiếu của đoạn $\overline{AB}$ lên trục hoành.

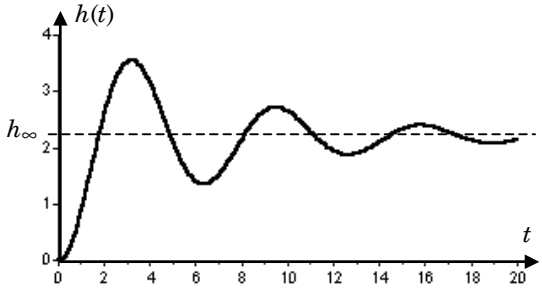
5) Tính $T_t = \dfrac{y(0)T_m}{u_0 k}$.



Hình 3.26: Xác định tham số cho mô hình Lead (dẫn qua).

3.1.6 Xác định tham số mô hình đối tượng dao động bậc hai tắt dần

Xét một đối tượng tuyến tính cần phải nhận dạng mà khi kích thích chủ động tại đầu vào bằng hàm $u(t)=1(t)$ ta thu được ở đầu ra đường thực nghiệm của hàm quá độ $h(t)$ có dạng giống như trong hình 3.27.



Hình 3.27: Hàm quá độ của đối tượng dao động bậc hai.

Đường thực nghiệm này có những đặc điểm sau:

- Hàm $h(t)$ xuất phát từ điểm 0 và có đạo hàm tại 0 bằng 0. Theo kết luận 1 đã nêu trong mục 3.1.1 thì mô hình (3.1) của nó phải có $n_a - n_b > 1$.
- Khi $t \to \infty$ thì $h(t)$ tiến tới một hằng số h_∞. Cũng theo kết luận 1 thì mô hình tham số (3.1) của nó có $a_0 \neq 0$.
- Hàm $h(t)$ có vô số điểm cực trị phân bố xen kẽ, cách đều nhau, trong đó các điểm cực đại lớn hơn h_∞ còn các giá trị cực tiểu thì nhỏ hơn h_∞. Theo kết luận 5, mô hình tham số (3.1) phải có những điểm cực là số phức liên hợp.

Với những kết luận như vậy, ta hoàn toàn có thể mô tả đối tượng bằng mô hình

$$G(s) = \frac{kq^2}{s^2 + 2qDs + q^2} \qquad \text{trong đó} \qquad 0 < D < 1 \tag{3.45}$$

Từ mô hình này ta có được ngay:

$$h_\infty = \lim_{t \to \infty} h(t) = \lim_{s \to 0} G(s) = k \tag{3.46}$$

bởi vậy vấn đề còn lại phải làm chỉ là xác định nốt D và q.

Gọi s_1 và s_2 là hai điểm cực của mô hình (3.45), ta có

$$s_{1,2} = -Dq \pm jq\sqrt{1 - D^2}$$

Suy ra

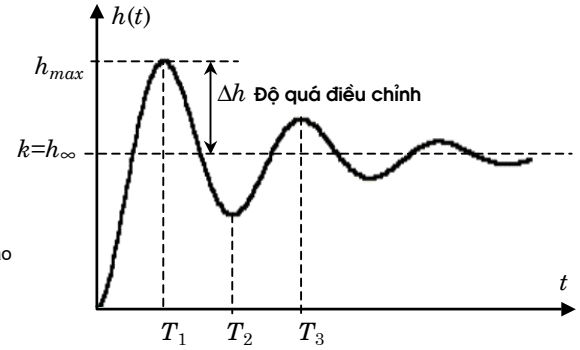
$$h(t) = k\left[1 - \frac{e^{-Dqt}}{\sqrt{1-D^2}}\sin\left(q\sqrt{1-D^2}\ t + \arccos D\right)\right]$$

và

$$\frac{dh(t)}{dt} = k\frac{qe^{-Dqt}}{\sqrt{1-D^2}}\sin\left(q\sqrt{1-D^2}\ t\right)$$

Giải phương trình $\frac{dh(t)}{dt} = 0$ để tìm các điểm cực trị (kể cả tại điểm t=0) được:

$$T_i = \frac{i\pi}{q\sqrt{1-D^2}}, \qquad i = 0, 1, \ldots . \tag{3.47}$$



Hình 3.28: Xác định tham số mô hình dao động bậc hai từ hàm quá độ.

Do đó

$$h_{max} = h(T_1) = k\left[1 - \frac{\sin(\pi + \arccos D)}{\sqrt{1-D^2}}\exp\left(\frac{-\pi D}{\sqrt{1-D^2}}\right)\right]$$

$$= k\left[1 + \exp\left(\frac{-\pi D}{\sqrt{1-D^2}}\right)\right] \tag{3.48}$$

Trừ (3.48) cho (3.46) ta sẽ có độ quá điều chỉnh (hình 3.28):

$$\Delta h = h_{max} - h_\infty = k\exp\left(\frac{-\pi D}{\sqrt{1-D^2}}\right) \tag{3.49}$$

Nếu chia (3.49) cho (3.46) theo từng vế được:

$$\left|\frac{\Delta h}{k}\right| = \exp\left(\frac{-\pi D}{\sqrt{1-D^2}}\right) \qquad \Leftrightarrow \qquad D = \frac{1}{\sqrt{1+\frac{\pi^2}{\ln^2\left|\frac{\Delta h}{k}\right|}}} \tag{3.50}$$

và đó chính là công thức cho phép xác định tham số D từ đường thực nghiệm $h(t)$.

Tham số q còn lại sẽ được xác định từ D với sự trợ giúp của T_1 theo (3.47) như sau:

$$T_1 = \frac{\pi}{q\sqrt{1-D^2}} \qquad \Leftrightarrow \qquad q = \frac{\pi}{T_1\sqrt{1-D^2}} \tag{3.51}$$

Ba công thức (3.36), (3.50) và (3.51) đặt cơ sở cho việc xác định các tham số k, D, q của mô hình (3.45) từ đường thực nghiệm $h(t)$. Nếu như rằng vì một lý do nào đó ta không có được hàm quá độ $h(t)$ mà chỉ có đáp ứng $y(t)$ khi đối tượng được kích thích bởi $u(t)= u_0 1(t)$ thì do đối tượng là tuyến tính, những công thức trên sẽ có một sửa đổi nhỏ lại thành (phần chứng minh dành cho bài tập):

$$y_\infty = \lim_{t\to\infty} y(t) = ku_0 \tag{3.52a}$$

$$D = \frac{1}{\sqrt{1+\frac{\pi^2}{\ln^2\left|\frac{\Delta y}{y_\infty}\right|}}}, \text{ trong đó } \Delta y = y_{max} - y_\infty \tag{3.52b}$$

$$q = \frac{\pi}{T_1\sqrt{1-D^2}} \tag{3.53c}$$

Tổng kết lại, ta đi đến thuật toán:

1) Kẻ đường tiệm cận với $y(t)$ tại $t=\infty$ rồi xác định k theo (3.52a), tức là $k = \frac{y_\infty}{u_0}$

2) Xác định y_{max} và T_1 là tọa độ của điểm cực đại đầu tiên của đường $y(t)$.

3) Tính D theo (3.52b), trong đó $\Delta y = y_{max} - y_\infty$

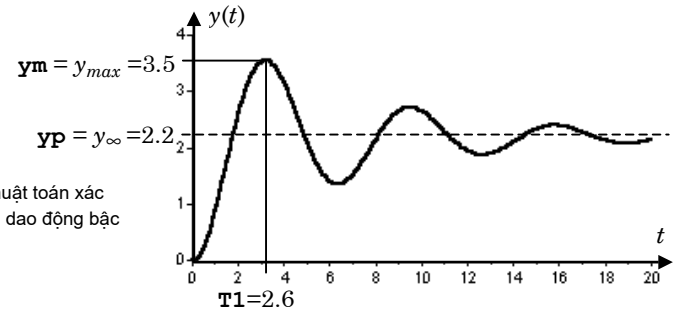
4) Tính q từ T_1 và D theo (3.53c).

Các bước tính toán bao gồm tính k, D và q từ y_∞ , y_{max} và T_1 của thuật toán đã được cài đặt thành chương trình con **`ptc()`** viết trên C cho dưới đây để tham khảo. Chương trình **`ptc()`** có 7 biến hình thức, trong đó **`u0`**, **`yp`**, **`ym`**, **`T1`** chứa các sơ kiện đầu vào u_0 , y_∞ , y_{max} , T_1 còn **`k`**, **`D`**, **`q`** chứa kết quả tính được là các tham số mô hình k, D, q.

```
void ptc( double u0,double yp,double ym,double T1,
          double &k,double &D,double &q)
{
   k=yp/u0;
   D=1./sqrt(1.+(pow(M_PI,2.)/pow(log(fabs(ym-yp)/yp),2.)));
   q=M_PI/(T1*sqrt(1.-D*D));
}
```

Ví dụ: Giả sử rằng khi kích thích một đối tượng bằng tín hiệu $u(t)=1(t)$ ở đầu vào ta thu được tại đầu ra đường thực nghiệm cho trong hình 3.29. Từ đường thực nghiệm đó ta đọc ra được:

$y_\infty = 2{,}2$ $\quad y_{max} = 3{,}5$ $\quad T_1 = 2{,}6$



Hình 3.29: Ví dụ minh họa thuật toán xác định tham số mô hình dao động bậc hai.

Gọi chương trình **ptc()** với các giá trị

```
double u0=1.,yp=2.2,ym=3.5,T1=2.6,k,D,q;
ptc(u0,yp,ym,T1,k,D,q);
```

sẽ thu được kết quả

k =2,2 $\quad$ **D** =0,165161 $\quad$ **q** =1,22513 ❑

3.2 Xác định tham số mô hình từ những giá trị $G(jn\Omega_\lambda)$ đã có

Những phương pháp xác định mô hình tham số từ đáp ứng đầu ra $y(t)$ của đối tượng tuyến tính cần nhận dạng khi đối tượng được kích thích bởi tín hiệu đầu vào $u(t)= u_0 1(t)$ đã được trình bày trong mục 3.1 có ưu điểm là chỉ cần từ dạng đường thực nghiệm $y(t)$ người ta đã có thể khoanh vùng lớp các mô hình thích hợp cho đối tượng, nhưng lại có một nhược điểm chính hạn chế ứng dụng tổng quát là việc xác định tham số cho mô hình thuộc lớp các mô hình đó phụ thuộc khá nhiều vào dạng lớp các mô hình cụ thể và không tổng quát hóa được cho một lớp mô hình có cấu trúc bất kỳ (bậc mô hình là cho trước).

Chính vì vậy, các phương pháp đó chỉ có thể được gói gọn trong phạm vi sử dụng với những mô hình đơn giản (bậc mô hình thấp) thuộc một trong 7 lớp:

1) Lớp các mô hình PT_1
2) Lớp các mô hình IT_1
3) Lớp các mô hình IT_n
4) Lớp các mô hình PT_2
5) Lớp các mô hình PT_n
6) Lớp các mô hình Lead/Lag
7) Lớp các mô hình khâu dao động bậc hai tắt dần.

Để bù đắp được hạn chế trên, mục này sẽ giới thiệu thêm các phương pháp xác định tham số $b_0, b_1, \dots, b_{n_b}, a_0, a_1, \dots, a_{n_a}$, cho mô hình thuộc lớp

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_0 + b_1 s + \cdots + b_{n_b} s^{n_b}}{a_0 + a_1 s + \cdots + a_{n_a} s^{n_a}}, \qquad n_a \geq n_b \tag{3.54}$$

từ các giá trị $\{G(jn\Omega_\lambda)\}$, n=0,1, … , $2M$ của $G(s)$ được giả thiết là đã có, chẳng hạn như nhờ thuật toán nhận dạng bị động bằng phân tích phổ và hàm **`nonpar()`** đã giới thiệu ở chương 2. Các phương pháp này cũng cần thêm giả thiết nữa là bậc của mô hình n_a, n_b đã biết trước (mô hình có cấu trúc).

Do các phương pháp nhận dạng $b_0, b_1, \dots, b_{n_b}, a_0, a_1, \dots, a_{n_a}$ có sử dụng phương pháp tính Cholesky giải hệ phương trình tuyến tính phức

$$\begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} k_1 \\ \vdots \\ k_n \end{pmatrix}, \text{ với } c_{ij}, k_i \in \mathbb{C} \tag{3.55}$$

nên để tiện cho việc theo dõi, thuật toán Cholesky sẽ được giới thiệu trước khi đi vào cụ thể các phương pháp nhận dạng tham số $b_0, b_1, \dots, b_{n_b}, a_0, a_1, \dots, a_{n_a}$.

3.2.1 Thuật toán Cholesky

Nếu sử dụng các ký hiệu

$$C = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{pmatrix}, \quad \underline{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \qquad \underline{k} = \begin{pmatrix} k_1 \\ \vdots \\ k_n \end{pmatrix}$$

phương trình (3.55) với các hệ số phức c_{ij} , k_i , $i, j = 1, \dots , n$ viết được thành

$$C\underline{x} = \underline{k} \tag{3.56}$$

trong đó nghiệm $\underline{x}$ cũng là vector gồm các phần tử phức.

Việc giải phương trình trên không thể được thực hiện đơn giản là tính ma trận nghịch đảo C^{-1} để đưa ra nghiệm $\underline{x}= C^{-1}\underline{k}$ vì như làm như vậy, số lượng các phép tính phải thực hiện là rất nhiều kéo theo sai số lớn, đặc biệt sai số tính toán sẽ càng lớn khi ma trận C tuy không suy biến nhưng có định thức tương đối nhỏ. Bởi vậy để giải phương trình (3.56) người ta cần phải có những thuật toán thích hợp và một trong những thuật toán đó là thuật toán của Cholesky.

Điều kiện để áp dụng được thuật toán Cholesky là cần phải có giả thiết rằng ma trận C xác định dương và đối xứng Hermitian, tức là

$$\underline{x}^H C\underline{x} \geq 0, \qquad \underline{x}^H C\underline{x} = 0 \quad \text{khi và chỉ khi} \quad \underline{x} = \underline{0},$$

$$C^H = C,$$

trong đó ký hiệu mũ H chỉ phép tính chuyển vị và lấy các phần tử liên hợp của một ma trận (hay vector). Giả thiết trên, như sau này chỉ rõ, hoàn toàn phù hợp với bài toán nhận dạng tham số mô hình của ta.

Tư tưởng chính của thuật toán Cholesky là phân tích ma trận C thành tích $C=DD^H$ với D là ma trận mà các phần tử phía trên đường chéo chính đồng nhất bằng 0

$$D = \begin{pmatrix} d_{11} & 0 & \cdots & 0 \\ d_{21} & d_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{pmatrix}, \quad \text{tức là} \quad d_{ij} = 0 \quad \text{khi } i<j$$

Việc phân tích C thành $C=DD^H$ luôn thực hiện được nếu C xác định dương và thỏa mãn $C^H = C$.

Nếu C đã được phân tích thành $C=DD^H$ thi công việc giải hệ phương trình (3.56) sẽ không cần đến ma trận nghịch đảo C^{-1} nữa mà trở nên đơn giản hơn nhiều bằng việc giải hai hệ phương trình tuyến tính:

$$D\,\underline{y} = \underline{k} \qquad \Leftrightarrow \qquad \begin{pmatrix} d_{11} & 0 & \cdots & 0 \\ d_{21} & d_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} k_1 \\ \vdots \\ k_n \end{pmatrix} \tag{3.57a}$$

và

$$D^H \underline{x} = \underline{y} \qquad \Leftrightarrow \qquad \begin{pmatrix} \overline{d}_{11} & \overline{d}_{21} & \cdots & \overline{d}_{n1} \\ 0 & \overline{d}_{22} & \cdots & \overline{d}_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \overline{d}_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \tag{3.57b}$$

trong đó $\overline{d}_{ij}$ là ký hiệu chỉ số phức liên hợp của d_{ij}.

Như đã nói, việc tìm nghiệm hai phương trình trên không cần phải thông qua ma trận nghịch đảo D^{-1}. Thật vậy để giải phương trình (3.57a) ta chỉ cần tiến hành các bước sau:

1) Xác định $y_1 = \dfrac{k_1}{d_{11}}$

2) Lần lượt với i=2, … , n tính $y_i = \dfrac{1}{d_{ii}} \left(k_i - \sum_{j=1}^{i-1} d_{ij} y_j \right)$

và để giải phương trình (3.57b) ta thực hiện (phần chứng minh dành cho bài tập):

1) Xác định $x_n = \dfrac{y_n}{\overline{d}_{nn}}$

2) Thực hiện lần lượt cho $j = n-1, \ldots, 1$ công thức $x_j = \dfrac{1}{\overline{d}_{jj}} \left(y_j - \sum_{i=j+1}^{n} \overline{d}_{ij} x_i \right)$

Điều băn khoan rằng hai thuật toán trên có thực hiện được không, tức là thực sự các phần tử d_{ii} trên đường chéo chính của D có khác 0 hay không, sẽ được trả lời với tính chất xác định dương của C như sau:

Định lý 3.1: Nếu ma trận C với các phần tử c_{ij} là những số phức có tính xác định dương thì

a) C không suy biến.

b) Các phần tử trên đường chéo chính c_{ii} là những số thực dương.

Chứng minh: a) Tính không suy biến của C được suy ra từ định nghĩa về tính xác định dương, vì nếu C suy biến thì tồn tại $\underline{x} \neq \underline{0}$ để có $C\underline{x} = \underline{0}$ và do đó $\underline{x}^H C \underline{x} = 0$.

Về b) ta chỉ cần thay $\underline{x} = \underline{e}_i$ vào công thức $\underline{x}^H C \underline{x} > 0$ khi $\underline{x} \neq \underline{0}$, trong đó $\underline{e}_i$ là vector đơn vị có phần tử thứ i bằng 1 còn các phần tử khác bằng 0, sẽ được

$$\underline{e}_i^H C \underline{e}_i > 0 \quad \Rightarrow \quad c_{ii} > 0 \qquad \text{(đ.p.c.m).}$$ ❑

Với định lý 3.1 và quan hệ $C = DD^H$ thì D cũng phải là ma trận không suy biến, tức là det $D \neq 0$. Nhưng do det $D = \prod_{i=1}^{n} d_{ii}$ nên phải có $d_{ii} \neq 0$ với mọi i và đó chính là điều kiện để thuật toán giải phương trình (3.57) thực hiện được.

Bây giờ ta sẽ xác định công thức thực hiện phân tích $C=DD^H$. Từ

$$\begin{pmatrix} d_{11} & 0 & \cdots & 0 \\ d_{21} & d_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{pmatrix} \begin{pmatrix} \overline{d}_{11} & \overline{d}_{21} & \cdots & \overline{d}_{n1} \\ 0 & \overline{d}_{22} & \cdots & \overline{d}_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \overline{d}_{nn} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{pmatrix}$$

có

$$c_{ij} = \sum_{q=1}^{n} d_{iq}\overline{d}_{jq} = \sum_{q=1}^{\min(i,j)} d_{iq}\overline{d}_{jq} \tag{3.58}$$

Suy ra

$$c_{jj} = \sum_{q=1}^{j} d_{jq}\overline{d}_{jq} = \sum_{q=1}^{j} \left|d_{jq}\right|^2$$

Công thức này khẳng định lại một lần nữa rằng các phần tử trên đường chéo chính c_{jj} của C là những số thực dương. Để đơn giản ta sẽ tìm ma trận D có các phần tử trên đường chéo chính cũng là các số thực dương, khi đó thì

$$c_{jj} = d_{jj}^2 + \sum_{q=1}^{j-1} \left|d_{jq}\right|^2 . \Rightarrow \qquad d_{jj} = \sqrt{c_{jj} - \sum_{q=1}^{j-1} \left|d_{jq}\right|^2} \tag{3.59}$$

Muốn tìm những phần tử d_{ij} với $i>j$ còn lại ta đi từ (3.58). Do có giả thiết $i>j$ nên (3.58) trở thành

$$c_{ij} = \sum_{q=1}^{j} d_{iq}\overline{d}_{jq} = d_{ij}d_{jj} + \sum_{q=1}^{j-1} d_{iq}\overline{d}_{jq}$$

$$\Rightarrow \qquad d_{ij} = \frac{1}{d_{jj}}\left(c_{ij} - \sum_{q=1}^{j-1} d_{iq}\overline{d}_{jq}\right) \tag{3.60}$$

và đó chính là công thức xác định d_{ij} khi $i>j$.

Dựa vào (3.59), (3.60) ta có thuật toán tìm D: Thực hiện với j =1,2, … , n:

a) Tính d_{jj} theo (3.59).

b) Tính lần lượt các giá trị d_{ij} có $i = j$+1, … , n theo công thức (3.60).

Nếu ghép chung thuật toán tìm D vừa phát biểu với hai thuật toán giải hệ phương trình (3.57a) và (3.57b) ta sẽ đi đến thuật toán Cholesky gồm các bước tính sau:

1) Phân tích C thành DD^H, tức là tìm d_{ij} với $i \geq j$, bằng cách thực hiện lần lượt các bước sau với j =1,2, … , n:

   a) Tính $d_{jj} = \sqrt{c_{jj} - \sum_{q=1}^{j-1} \left| d_{jq} \right|^2}$

   b) Lần lượt với $i = j$+1, … , n tính $d_{ij} = \frac{1}{d_{jj}} \left(c_{ij} - \sum_{q=1}^{j-1} d_{iq} \overline{d}_{jq} \right)$

2) Lần lượt với i=1, … , n tính $y_i = \frac{1}{d_{ii}} \left(k_i - \sum_{j=1}^{i-1} d_{ij} y_j \right)$

3) Lần lượt với j =n, … , 1 tính $x_j = \frac{1}{d_{jj}} \left(y_j - \sum_{i=j+1}^{n} \overline{d}_{ij} x_i \right)$

Thuật toán vừa trình bày trên đã được cài đặt thành hàm **`cholesky()`** viết trên C cho dưới đây để tham khảo. Hàm này có 3 biến hình thức:

a) Số nguyên **`n`** chứa bậc của ma trận C, tức là chứa số các phương trình là n.

b) Con trỏ **`c`** chỉ đầu mảng số phức **`c[]`** chứa giá trị các phần tử c_{ij} của ma trận C. Do ma trận C đã được giả thiết là đối xứng Hermitian (C=C^H) nên hàm cũng chỉ cần các giá trị c_{ij} có $i \geq j$ là đủ. Những phần tử này là các giá trị đầu vào của hàm **`cholesky()`** và phải được đưa vào mảng **`c[]`** theo thứ tự như sau:

$$\begin{pmatrix} c_{11} & & & \\ c_{21} & c_{22} & & \\ \vdots & \vdots & \ddots & \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{pmatrix} \quad \text{đưa vào} \quad \begin{pmatrix} \texttt{c[0]} & & & \\ \texttt{c[1]} & \texttt{c[2]} & & \\ \vdots & \vdots & \ddots & \\ \texttt{c}\left[\frac{\texttt{n(n-1)}}{\texttt{2}}\right] & \cdots & \cdots & \texttt{c}\left[\frac{\texttt{n(n+1)}}{\texttt{2}}\texttt{-1}\right] \end{pmatrix}$$

   nói cách khác, phần tử c_{ij} phải được đưa vào **`c[p]`** với **`p`**$= \frac{i(i-1)}{2} - 1 + j$.

   Sau khi hàm thực hiện xong, mảng sẽ chứa các phần tử d_{ij} của D cũng theo đúng thứ tự như trên.

c) Con trỏ **`k`** chỉ đầu mảng số phức **`k[]`** chứa các giá trị đầu vào k_i, i=1, … , n theo thứ tự k_i nằm trong **`k[i-1]`**. Khi hàm **`cholesky()`** thực hiện xong, mảng này sẽ chứa kết quả x_i, i=1, … , n cũng theo đúng thứ tự đó.

Hàm trả về giá trị 0 nếu quá trình thực hiện không có lỗi (ma trận C xác định dương) hoặc là một giá trị i khác 0 thông báo C có phần tử $c_{ii} \leq 0$.

```
int cholesky(int n,complex *c,complex *k)
{
   int i,j,q,p,l;
   double s;
   for(j=1;j<=n;j++)
   {  p=(j*(j-1)-2)/2;
      s= real(c[p+j]);
      for(q=1;q<j;q++)
         s -= pow(real(c[p+q]),2)+pow(imag(c[p+q]),2);
      if(s<=0.) return(j); else s=sqrt(s);
      c[p+j]=complex(s);
      for(i=j+1;i<=n;i++)
      {  l=(i*(i-1)-2)/2;
         for(q=1;q<j;q++) c[l+j]-= c[l+q]*conj(c[p+q]);
         c[l+j]=c[l+j]/s;
      }
   }
   for(i=1;i<=n;i++)
   {  p=(i*(i-1)-2)/2;
      for(j=1;j<i;j++) k[i-1]-= c[p+j]*k[j-1];
      k[i-1]=k[i-1]/c[p+i];
   }
   for(j=n;j>0;j--)
   {  for(i=j+1;i<=n;i++)
      {  p=(i*(i-1)-2)/2;
         k[j-1]-= conj(c[p+j])*k[i-1];
      }
      k[j-1]=k[j-1]/c[(j*(j+1)-2)/2];
   }
   return(0);
}
```

Ví dụ: Gọi hàm trên với các giá trị đầu vào **n**=3,

$$\begin{pmatrix} \texttt{c[0]} & & \\ \texttt{c[1]} & \texttt{c[2]} & \\ \texttt{c[3]} & \texttt{c[4]} & \texttt{c[5]} \end{pmatrix} = \begin{pmatrix} 36 & & \\ 24+6j & 42 & \\ 6+6j & 15+8j & 16 \end{pmatrix} \quad \text{và} \quad \begin{pmatrix} \texttt{k[0]} \\ \texttt{k[1]} \\ \texttt{k[2]} \end{pmatrix} = \begin{pmatrix} 102-30j \\ 153-18j \\ 84+22j \end{pmatrix}$$

sẽ thu được kết quả

$$\begin{pmatrix} \texttt{c[0]} & & \\ \texttt{c[1]} & \texttt{c[2]} & \\ \texttt{c[3]} & \texttt{c[4]} & \texttt{c[5]} \end{pmatrix} = \begin{pmatrix} 6 & & \\ 4+j & 5 & \\ 1+j & 2+j & 3 \end{pmatrix} \quad \text{và} \quad \begin{pmatrix} \texttt{k[0]} \\ \texttt{k[1]} \\ \texttt{k[2]} \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}. \qquad \square$$

3.2.2 Nhận dạng tham số mô hình

Sau khi đã có hàm **cholesky()** giải phương trình tuyến tính phức, ta quay lại bài toán nhận dạng tham số mô hình (3.54).

Giả sử rằng bằng cách nào đó, chẳng hạn như thông qua nhận dạng mô hình không tham số nhờ phân tích phổ tín hiệu với hàm **nonpar()** đã trình bày ở chương 2, ta đã có dãy các giá trị $\{G(jn\Omega_\lambda)\}$, $\Omega_\lambda=\dfrac{2\pi}{\lambda T_a}$, n=0,1, … , 2M của hàm truyền đạt $G(s)$ từ dãy các giá trị tín hiệu đo được $\{u_k\}$, $\{y_k\}$ với chu kỳ lấy mẫu T_a của tín hiệu vào ra $u(t)$, $y(t)$, trong đó λ là số nguyên lũy thừa 2 nhỏ nhất nhưng không nhỏ hơn 2N–1, $N=\dfrac{T}{T_a}$, T là khoảng thời gian quan sát (đo) tín hiệu [0, T) và M là chỉ số Lag. Nhiệm vụ đặt ra bây giờ là xác định các tham số b_0, b_1, … , b_{n_b}, a_0, a_1, … , a_{n_a} cho mô hình (3.54) từ các giá trị $G(jn\Omega_\lambda)$ đã có này.

Nếu như từ thông tin A–priori người ta không những biết được đối tượng là tuyến tính cũng như bậc của mô hình n_a, n_b mà còn biết thêm rằng đối tượng không có thành phần tích phân nối tiếp, ví dụ như hàm quá độ $h(t)$ không tiến đến ∞ khi $t\to\infty$, thì do $a_0\neq 0$ mô hình (3.54) hoàn toàn có thể được thay thế bởi dạng tương đương

$$G_M(s)=\frac{b_0+b_1s+\cdots+b_{n_b}s^{n_b}}{1+a_1s+\cdots+a_{n_a}s^{n_a}},\qquad (n_a\geq n_b). \tag{3.61}$$

Với (3.61) và $G(jn\Omega_\lambda)$, n=0,1, … , 2M thì khi thay s bởi $jn\Omega_\lambda$ vào về phải của (3.61) còn vế trái là những giá trị $G(jn\Omega_\lambda)$ đã có, sẽ được:

$$G(jn\Omega_\lambda)\approx\frac{b_0+b_1(jn\Omega_\lambda)+\cdots+b_{n_b}(jn\Omega_\lambda)^{n_b}}{1+a_1(jn\Omega_\lambda)+\cdots+a_{n_a}(jn\Omega_\lambda)^{n_a}} \tag{3.62}$$

$$\Rightarrow\qquad e_n=G(jn\Omega_\lambda)-\left[\sum_{i=0}^{n_b}b_i(jn\Omega_\lambda)^i-G(jn\Omega_\lambda)\sum_{i=1}^{n_a}a_i(jn\Omega_\lambda)^i\right]\neq 0 \tag{3.63}$$

Lý do cho việc dấu = ở (3.61) được thay bởi dấu ≈ trong (3.62) và do đó $e_n\neq 0$ là vế trái của (3.62) chỉ là những giá trị $G(jn\Omega_\lambda)$ nhận dạng được mà có nên không phải là giá trị đúng thực sự của hàm truyền đạt.

Từ (3.63) ta thấy bộ tham số b_0, b_1, … , b_{n_b}, a_1, … , a_{n_a} tốt nhất sẽ là bộ mà với nó tổng bình phương các sai lệch e_n có giá trị nhỏ nhất:

$$Q = \sum_{n=0}^{2M} |e_n|^2 \rightarrow \min!. \tag{3.64}$$

Nếu sử dụng ký hiệu

$$\underline{e} = \begin{pmatrix} e_0 \\ e_1 \\ \vdots \\ e_{\tilde{M}} \end{pmatrix}, \qquad \underline{g} = \begin{pmatrix} G(0) \\ G(j\Omega_\lambda) \\ \vdots \\ G(j\tilde{M}\Omega_\lambda) \end{pmatrix}, \qquad \underline{x} = \begin{pmatrix} b_0 \\ \vdots \\ b_{n_b} \\ -a_1 \\ \vdots \\ -a_{n_a} \end{pmatrix}$$

và
$$U = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 1 & (j\Omega_\lambda) & \cdots & (j\Omega_\lambda)^{n_b} & (j\Omega_\lambda)G(j\Omega_\lambda) & \cdots & (j\Omega_\lambda)^{n_a}G(j\Omega_\lambda) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 1 & (j\tilde{M}\Omega_\lambda) & \cdots & (j\tilde{M}\Omega_\lambda)^{n_b} & (j\tilde{M}\Omega_\lambda)G(j\tilde{M}\Omega_\lambda) & \cdots & (j\tilde{M}\Omega_\lambda)^{n_a}G(j\tilde{M}\Omega_\lambda) \end{pmatrix} \tag{3.65}$$

thì công thức (3.63), (3.64) cho tất cả n=0,1, … , $\tilde{M}$ =2M sẽ viết được thành

$$\underline{e} = \underline{g} - U\underline{x}$$

$$Q = \underline{e}^H \underline{e} \rightarrow \min!.$$

Suy ra

$$\begin{aligned} Q &= \underline{g}^H \underline{g} - \underline{g}^H U\underline{x} - \underline{x}^H U^H \underline{g} + \underline{x}^H U^H U\underline{x} \\ &= \underline{g}^H \underline{g} - \underline{g}^H U(U^H U)^{-1} U^H \underline{g} + \underbrace{(U^H \underline{g} - U^H U\underline{x})^H (U^H U)^{-1} (U^H \underline{g} - U^H U\underline{x})}_{\tilde{Q}} \end{aligned}$$

Vì chỉ có thành phần thứ 3 là $\tilde{Q}$ chứa vector các tham số $\underline{x}$ phải tìm nên Q nhỏ nhất khi và chỉ khi

$$\tilde{Q} = (U^H \underline{g} - U^H U\underline{x})^H (U^H U)^{-1} (U^H \underline{g} - U^H U\underline{x}) \rightarrow \min!.$$

Hơn nữa theo (3.65) ma trận U có $\tilde{M}$ +1 hàng n_a+n_b+1 cột. Nhưng do ở bài toán nhận dạng thường có $\tilde{M} > n_a$+n_b nên n_a+n_b+1 vector cột là độc lập tuyến tính, tức là U có hạng n_a+n_b+1, dẫn đến $U^H U$ không suy biến. Khi $U^H U$ không suy biến, nó sẽ xác định dương, kéo theo $(U^H U)^{-1}$ cũng là một ma trận xác định dương (phần chứng minh dành cho bài tập). Bởi vậy giá trị nhỏ nhất của $\tilde{Q}$ chỉ có thể là 0 với

$$U^H \underline{g} - U^H U\underline{x} = 0 \quad \Leftrightarrow \quad U^H U\underline{x} = U^H \underline{g} \tag{3.66}$$

và đó chính là công thức xác định bộ tham số $\underline{x}$ cho mô hình (3.61).

So sánh (3.66) với (3.56) ta thấy chúng hoàn toàn giống nhau, trong đó

$$C = U^H U \qquad \text{và} \qquad \underline{k} = U^H \underline{g} \tag{3.67}$$

nên (3.66) có thể được giải trực tiếp nhờ hàm **`cholesky()`** viết trên ngôn ngữ lập trình C đã giới thiệu tại mục trước.

Gọi các phần tử của C là c_{ik}, i, k=1, … , n_a+n_b+1, của U là u_{qk}, q= 1, … , $\tilde{M}$ +1, k=1, … , n_a+n_b+1 và của $\underline{k}$ là k_i, i = 1, … , n_a+n_b+1 thì từ (3.67) có

$$u_{qk} = \begin{cases} [j(q-1)\Omega_\lambda]^{k-1} & \text{khi} \quad 1 \le k \le n_b + 1 \\ [j(q-1)\Omega_\lambda]^{k-n_b-1} G(j(q-1)\Omega_\lambda) & \text{khi} \quad n_b + 1 < k \le n_a + n_b + 1 \end{cases} \tag{3.68}$$

$$c_{ik} = \sum_{q=1}^{\tilde{M}+1} \overline{u}_{qi} u_{qk} \tag{3.69}$$

$$k_i = \sum_{q=1}^{\tilde{M}+1} \overline{u}_{qi} G(j(q-1)\Omega_\lambda) . \tag{3.70}$$

Thay (3.68) vào (3.69) được

$$c_{11} = \tilde{M} + 1$$

$$c_{ik} = \begin{cases} (-1)^{k-1} \sum_{q=1}^{\tilde{M}} (-jq\Omega_\lambda)^{i+k-2} & \text{khi} \quad 1 < k \le i \le n_b + 1 \\ (-1)^{k-1} \sum_{q=1}^{\tilde{M}} (-jq\Omega_\lambda)^{i+k-2-n_b} \overline{G(jq\Omega_\lambda)} & \text{khi} \quad 1 < k \le n_b + 1 < i \\ (-1)^{k-n_b-1} \sum_{q=1}^{\tilde{M}} (-jq\Omega_\lambda)^{i+k-2-2n_b} \left|G(jq\Omega_\lambda)\right|^2 & \text{khi} \quad n_b + 1 < k \le i \end{cases} \tag{3.71}$$

và (3.68) vào (3.70) có

$$k_i = \begin{cases} \sum_{q=0}^{\tilde{M}} (\overline{jq\Omega_\lambda})^{i-1} G(jq\Omega_\lambda) & \text{khi} \quad 1 \le i \le n_b + 1 \\ \sum_{q=1}^{\tilde{M}} (\overline{jq\Omega_\lambda})^{i-1-n_b} \left|G(jq\Omega_\lambda)\right|^2 & \text{khi} \quad n_b + 1 < i \end{cases} \tag{3.72}$$

Hai công thức (3.71), (3.72) cũng như hàm **`nonpar()`** đã cho ở mục 2.3.1 (chương2) đặt cơ sở cho việc xây dựng thuật toán tìm bộ tham số tối ưu b_0 , b_1 , … , b_{n_b} , a_1 , … , a_{n_a} cho mô hình (3.61) trực tiếp từ dãy các giá trị tín hiệu $\{u_k\}$, $\{y_k\}$, k=1, … , N–1 đo được trong khoảng thời gian quan sát $[0, NT_a)$, trong đó T_a là thời gian trích mẫu.

Thuật toán này có dạng như sau:

1) Gọi hàm **nonpar()** với đầu vào $\{u_k\}$, $\{y_k\}$, k=1, … , N–1 để có $G(jn\Omega_\lambda)$, n=0,1, … , $\widetilde{M}$, trong đó $\widetilde{M}$=2M, M là chỉ số Lag (mục 2.2.1), $\Omega_\lambda=\dfrac{2\pi}{\lambda T_a}$ và λ là số nguyên lũy thừa 2 nhỏ nhất nhưng không nhỏ hơn 2N–1.

2) Xác định ma trận $C = U^H U$ và vector $\underline{k} = U^H \underline{g}$ bằng cách thực hiện lần lượt với i=1, … , n_a+n_b+1 các bước sau:

   a) Khi k=i, … , n_a+n_b+1, tính c_{ik} theo (3.71).

   b) Tính k_i theo (3.72).

3) Gọi hàm **cholesky()** với các tham số đầu vào đã có ở bước 2) để xác định $\underline{x}$ chứa bộ tham số b_0, b_1, … , b_{n_b}, a_1, … , a_{n_a} tối ưu.

Thuật toán trên đã được cài đặt thành hàm **par()** cho dưới đây để tham khảo. Hàm **par()** có các biến hình thức:

a) Con trỏ **u** chỉ đầu mảng số thực **u[]** chứa dãy $\{u_k\}$, k=1, … , N–1.

b) Con trỏ **y** chỉ đầu mảng số thực **y[]** chứa dãy $\{y_k\}$, k=1, … , N–1.

c) Số thức **Ta** chứa thời gian trích mẫu T_a.

d) Số nguyên **N** chứa độ dài hai dãy $\{u_k\}$, $\{y_k\}$, tức là chứa N.

e) Số nguyên **M** chứa chỉ số Lag, tức là độ dài cần phải có của dãy giá trị hàm tương quan $\{\widetilde{r}_{uy}(mT_a)\}$, đồng thời cũng xác định độ dài $\widetilde{M}$=2M của dãy kết quả $\{G(jn\Omega_\lambda)\}$ của hàm **nonpar()** là $\widetilde{M}$+1, tức là n=0, 1, , … , $\widetilde{M}$ với $\Omega_\lambda=\dfrac{2\pi}{\lambda T_a}$, trong đó λ là một số nguyên lũy thừa của 2 nhỏ nhất nhưng không nhỏ hơn 2N–1.

f) Số nguyên **bias** xác định giá trị hàm tương quan sẽ được tính theo công thức bias (**bias**=1) hay unbias (**bias**≠1).

g) Số nguyên **w** xác định chỉ số 0≤i≤7 của hàm cửa sổ sẽ được sử dụng nhằm làm giảm sai số rò rỉ khi sử dụng kỹ thuật DFT. Nếu nội dung của **w** là một số ngoài khoảng [0,7] thì hàm sẽ sử dụng hàm cửa sổ $w_0(t)$.

h) Số nguyên **na** xác định bậc đa thức mẫu số của mô hình (3.61).

i) Số nguyên **nb** xác định bậc đa thức tử số của mô hình (3.61).

j) Con trỏ **x** chỉ đầu mảng số phức **x[]** có độ dài n_a+n_b+1 chứa kết quả theo thứ tự **x[0]**= b_0, … , **x[nb]**= b_{n_b} , **x[nb+1]**= $-a_1$, … , **x[nb+na]**= $-a_{n_a}$.

Hàm sẽ trả về giá trị báo lỗi –2 nếu $n_a<n_b$, –1 khi $M \geq N$ hoặc $i > 0$ thông báo $U^H U$ suy biến với phần tử thứ ii không phải là số dương. Trong trường hợp không có lỗi, hàm trả về giá trị 0. Hàm không làm thay đổi nội dung hai mảng **u[]** và **y[]**.

```
int par(double *u,double *y,double Ta,int N,int M,int bias,
        int w,int na,int nb,complex *x)
{  int i,k,m=na+nb+1,q,L,ik,Q=2*M;
   double om,*s;
   if(nb>na) return(-2);
   complex ci,*G,*c,im=complex(1.);
   G=new complex[Q+1];
   c=new complex[m*(m+1)/2+1];
   s=new double[Q];
   if((L=nonpar(u,y,Ta,N,M,bias,w,G))>0)
   {  c[0]=complex(Q+1);
      x[0]=G[0];
      om=(2.*M_PI)/(Ta*L);
      for(i=0;i<Q;i++)
      {  s[i]=1.;
         x[0]=x[0]+G[i+1];
      }
      for(ik=1;ik<=2*na;ik++)
      {  for(q=0;q<Q;q++) s[q]=s[q]*om*(q+1);
         im=im*complex(0.,-1);
         if(ik<=nb)
         {  x[ik]=complex(0.);
            for(q=0;q<Q;q++) x[ik]+=im*G[q+1]*s[q];
         }
         if(ik<=na)
         {  x[nb+ik]=complex(0.);
            for(q=0;q<Q;q++) x[nb+ik]+=im*s[q]*pow(abs(G[q+1]),2.);
         }
         for(i=1;i<=na+nb+1;i++)
            for(k=1;k<=i;k++)
            {  if(i<=nb+1) L=i+k-2;
               else L=(k<=nb+1)? i-nb+k-2:i-2*nb+k-2;
               if(L!=ik) continue;
               L=k%2;
               ci=complex(0.);
               if(i<=nb+1) for(q=0;q<Q;q++) ci += im*s[q];
               else if(k<=nb+1)
                  for(q=0;q<Q;q++) ci += im*conj(G[q+1])*s[q];
               else
```

```
                {  for(q=0;q<Q;q++) ci+=im*s[q]*pow(abs(G[q+1]),2.);
                   L=(k-nb)%2
                }
                 c[i*(i-1)/2-1+k]=(L==0)? -ci : ci;
            }
      }
      L=cholesky(na+nb+1,c,x);
   }
   delete [] G; delete [] c;      delete [] s;
   return (L);
}
```

Chú ý: Thuật toán trên cũng như hàm **par()** chỉ sử dụng được khi đối tượng không có thành phần tích phân I, tức là $\lim_{t\to\infty} h(t) < \infty$.

Ví dụ: Cho một đối tượng tuyến tính có mô hình

$$G_M(s) = \frac{b_0}{1 + a_1 s + a_2 s^2},$$

trong đó

b_0 =2,5 , a_1 =1,6.10^{-3} và a_2 =5,5.10^{-7}.

Đối tượng làm việc tín hiệu vào $u(t)$ là ngẫu nhiên egodic có dải tần số lớn. Đo 2048 giá trị tín hiệu vào $u(t)$ và ra $y(t)$ của đối tượng với tần số trích mẫu **Ta**=T_a=$10^{-4}s$ được dãy $\{u_k\}$, $\{y_k\}$, k=1, 2, … , 2047.

Sử dụng hàm **par()** với hàm cửa sổ Hanning (**w**=w=3), **na**=n_a=2, **nb**=n_b=0, **N**=N=2048, **bias**=1 cho các giá trị **M**=M khác nhau ta sẽ có những kết quả sau:

Lag M	b_0 =2,5	a_1 =1,6.10^{-3}	a_2 =5,5.10^{-7}
195	2.39445e+00	1.46897e-03	5.43231e-07
225	2.41471e+00	1.51376e-03	5.54454e-07
300	2.42715e+00	1.53705e-03	5.50963e-07
350	2.42666e+00	1.50772e-03	5.43697e-07
365	2.44589e+00	1.51191e-03	5.50411e-07
390	2.34944e+00	1.45696e-03	5.22151e-07

Khi đối tượng có chứa thành phần tích phân I thì không mất tính tổng quát ta có thể giả thiết $b_0 \neq 0$, tức là không chứa thành phần vi phân D nối tiếp, vì trong trường hợp a_0=b_0=0 thì sau khi giản ước thừa số chung của tử và mẫu của (3.54) là s để hạ bậc ta lại có mô hình mới không chứa đồng thời cả hai thành phần I và D.

Khi $b_0 \neq 0$, ta làm tương tự như trên với mô hình rút gọn

$$G_M(s) = \frac{1 + b_1 s + \cdots + b_{n_b} s^{n_b}}{a_0 + a_1 s + \cdots + a_{n_a} s^{n_a}} \tag{3.73}$$

sẽ đi đến

$$G_M(jn\Omega_\lambda) \approx \frac{\widetilde{S}_{uy}(jn\Omega_\lambda)}{\widetilde{S}_u(n\Omega_\lambda)}$$

$$\Rightarrow \quad \widetilde{S}_u(n\Omega_\lambda)\left(1 + \sum_{i=1}^{n_b} b_i (jn\Omega_\lambda)^i\right) \approx \widetilde{S}_{uy}(n\Omega_\lambda) \sum_{i=0}^{n_a} a_i (jn\Omega_\lambda)^i$$

$$\Rightarrow \quad e_n = \widetilde{S}_u(n\Omega_\lambda) - \left(\widetilde{S}_{uy}(n\Omega_\lambda) \sum_{i=0}^{n_a} a_i (jn\Omega_\lambda)^i - \widetilde{S}_u(n\Omega_\lambda) \sum_{i=1}^{n_b} b_i (jn\Omega_\lambda)^i\right) \neq 0$$

$$\Rightarrow \quad \underline{e} = \underline{s} - L\underline{x}$$

trong đó

$$\underline{e} = \begin{pmatrix} e_0 \\ e_1 \\ \vdots \\ e_{2M} \end{pmatrix}, \quad \underline{x} = \begin{pmatrix} a_0 \\ \vdots \\ a_{n_a} \\ -b_1 \\ \vdots \\ -b_{n_b} \end{pmatrix}, \quad \underline{s} = \begin{pmatrix} \widetilde{S}_u(0) \\ \widetilde{S}_u(\Omega_\lambda) \\ \vdots \\ \widetilde{S}_u(2M\Omega_\lambda) \end{pmatrix}, \quad L = (l_{nk})$$

với

$$l_{nk} = \begin{cases} (jn\Omega_\lambda)^k \widetilde{S}_{uy}(jn\Omega_\lambda) & \text{nếu} \quad k \leq n_a \\ (jn\Omega_\lambda)^{k-n_a} \widetilde{S}_u(jn\Omega_\lambda) & \text{nếu} \quad n_a < k \end{cases}$$

$n = 0, 1, , \ldots, 2M$ và $k = 0, 1, , \ldots, n_a + n_b$.

Như vậy bộ tham số $b_1, \ldots, b_{n_b}, a_0, a_1, \ldots, a_{n_a}$ của mô hình (3.73) mà với nó phiếm hàm sai lệch (3.64) đạt giá trị nhỏ nhất sẽ là nghiệm của

$$L^H L\underline{x} = L^H \underline{s}$$

và cùng với kết luận đó ta được thuật toán xác định bộ tham số tối ưu $\underline{x}$ cho mô hình (3.73) từ dãy giá trị tín hiệu $\{u_k\}$, $\{y_k\}$, $k = 1, \ldots, N-1$ đo được trong khoảng thời gian quan sát $[0, NT_a)$ với thời gian trích mẫu T_a như sau:

1) Sử dụng hàm **spec()** tính $\widetilde{S}_u(n\Omega_\lambda)$, $\widetilde{S}_{uy}(n\Omega_\lambda)$, $n = 0,1, \ldots, 2M$ với M là chỉ số Lag, $\Omega_\lambda = \frac{2\pi}{\lambda T_a}$ và λ là số nguyên lũy thừa 2 nhỏ nhất nhưng không nhỏ hơn $2N-1$.

2) Tính $C=L^H L$ và $\underline{k}=L^H \underline{s}$.

3) Gọi hàm **cholesky()** để tìm $\underline{x}$ chứa bộ tham số $b_1, \ldots, b_{n_b}, a_0, a_1, \ldots, a_{n_a}$ tối ưu.

Việc cài đặt thuật toán trên thành chương trình trên ngôn ngữ lập trình C được tiến hành tương tự như đã làm với hàm **par()** và sẽ được dành cho bạn đọc như bài tập ôn luyện.

3.2.3 Nhận dạng lặp tham số mô hình

Ví dụ minh họa cho hàm **par()** trình bày trong mục 3.2.2 để nhận dạng tham số b_0, $b_1, \ldots, b_{n_b}, a_1, \ldots, a_{n_a}$ cho mô hình đối tượng không chứa thành phần tích phân

$$G_M(s)=\frac{b_0+b_1 s+\cdots+b_{n_b}s^{n_b}}{1+a_1 s+\cdots+a_{n_a}s^{n_a}}, \qquad (n_a \geq n_b) \tag{3.74}$$

xác nhận khả năng áp dụng tốt của thuật toán. Tuy nhiên không phải trong mọi trường hợp ta đều có thể sử dụng thuật toán đó. Chẳng hạn như ở bài toán nhận dạng bị động có nhiễu tác động lớn làm cho kết quả đo $\{u_k\}$, $\{y_k\}$ phản ánh không được sát thực tín hiệu hiện có của đối tượng. Điều này ảnh hưởng trực tiếp tới giá trị $G(jn\Omega_\lambda)$, n=0,1, … , $2M$ thu được của hàm truyền đạt và giữa giá trị thu được đó với giá trị đúng $G_M(jn\Omega_\lambda)$ tồn tại sai lệch không thể bỏ qua.

Ví dụ 1: Giả sử một đối tượng tuyến tính mô tả chính xác được bởi

$$G_M(s)=\frac{b_0+b_1 s}{1+a_1 s+a_2 s^2}$$

với b_0=2 , b_1=0,5 , a_1 = 0,8 và a_2 = 2.

Đo 1000 giá trị tín hiệu vào $u(t)$ và ra $y(t)$ của đối tượng với tần số trích mẫu **Ta**=T_a=$10^{-2}s$ được dãy $\{u_k\}$, $\{y_k\}$, k=1, 2, … , 999, trong đó ở cả đầu vào/ra của đối tượng có lẫn 12% nhiễu phân bố chuẩn với giá trị trung bình bằng 0. Sử dụng hàm **par()** với cửa sổ Hamming (**w**=4), **na**=2, **nb**=1, **N**=1000, **bias**=1 và **M**=200 ta thu được kết quả có sai lệch đáng kể ([12]):

b_0 = 0,38 , b_1 = 0,12, a_1 = 0,35 và a_2 = 0,17. ❑

Nhằm hiệu chỉnh kết quả thu được nhờ hàm **par()** một cách tốt hơn, người ta cần phải loại bỏ sự ảnh hưởng của nhiễu trong $G(jn\Omega_\lambda)$ và một trong những phương pháp loại bỏ sự ảnh hưởng đó của nhiễu là nhận dạng theo nguyên lý lặp qua nhiều bước được trình bày sau đây.

Xuất phát từ mô hình (3.74) và các giá trị $G(jn\Omega_\lambda)$, n=0,1, … , 2M đã có ta lập phương trình mô tả sai lệch đầu ra:

$$e_n = G(jn\Omega_\lambda) - G_M(jn\Omega_\lambda)$$
$$= \frac{G(jn\Omega_\lambda) - \left[\sum_{i=0}^{n_b} b_i (jn\Omega_\lambda)^i - G(jn\Omega_\lambda)\sum_{i=1}^{n_a} a_i (jn\Omega_\lambda)^i\right]}{1 + \sum_{i=1}^{n_a} a_i (jn\Omega_\lambda)^i}$$

Nếu ký hiệu $a_i^{(k)}$, $b_i^{(k)}$ là các tham số của mô hình (3.74) thu được tại bước lặp thứ k, ta sẽ cải biên sai lệch trên một ít cho phù hợp với thuật toán lặp như sau:

$$\tilde{e}_n = \frac{G(jn\Omega_\lambda) - \left[\sum_{i=0}^{n_b} b_i^{(k)} (jn\Omega_\lambda)^i - G(jn\Omega_\lambda)\sum_{i=1}^{n_a} a_i^{(k)} (jn\Omega_\lambda)^i\right]}{1 + \sum_{i=1}^{n_a} a_i^{(k-1)} (jn\Omega_\lambda)^i}$$
$$= \frac{G(jn\Omega_\lambda) - \left[\sum_{i=0}^{n_b} b_i^{(k)} (jn\Omega_\lambda)^i - G(jn\Omega_\lambda)\sum_{i=1}^{n_a} a_i^{(k)} (jn\Omega_\lambda)^i\right]}{w_n^{(k-1)}} \qquad (3.75)$$

trong đó

$$w_n^{(k-1)} = 1 + \sum_{i=1}^{n_a} a_i^{(k-1)} (jn\Omega_\lambda)^i \qquad \text{và} \qquad w_n^{(0)} = 1, \forall\, n.$$

Với ký hiệu ma trận U vector $\underline{g}$ cho trong (3.65) và

$$\underline{\tilde{e}} = \begin{pmatrix} \tilde{e}_0 \\ \tilde{e}_1 \\ \vdots \\ \tilde{e}_{2M} \end{pmatrix}, \quad \underline{x}_k = \begin{pmatrix} b_0^{(k)} \\ \vdots \\ b_{n_b}^{(k)} \\ -a_1^{(k)} \\ \vdots \\ -a_{n_a}^{(k)} \end{pmatrix}, \quad W^{(k-1)} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & \frac{1}{w_1^{(k-1)}} & 0 & \cdots & 0 \\ 0 & 0 & \frac{1}{w_2^{(k-1)}} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{1}{w_{2M}^{(k-1)}} \end{pmatrix}$$

thì (3.75) cho tất cả n=0,1, … , 2M sẽ viết chung lại được thành

$$\underline{\tilde{e}} = W^{(k-1)}\, \underline{g} - W^{(k-1)} U\, \underline{x}_k \qquad (3.76)$$

Nếu xem bộ tham số $\underline{x}_k$ tốt nhất của bước lặp thứ k là bộ mà với nó tổng bình phương các sai lệch

$$Q = \sum_{n=0}^{2M} \left|\tilde{e}_n\right|^2 = \underline{\tilde{e}}^H \underline{\tilde{e}}$$

đạt giá trị nhỏ nhất, thì tương tự như đã làm với (3.64) tại mục 3.2.2 để đến (3.66), ở đây ta cũng có phương trình xác định nghiệm $\underline{x}_k$ như sau:

$$\underbrace{\left(W^{(k-1)}U\right)^H \left(W^{(k-1)}U\right)}_{C} \cdot \underline{x}_k = \underbrace{\left(W^{(k-1)}U\right)^H W^{(k-1)} \underline{g}}_{\underline{k}}$$

và phương trình này hoàn toàn giải được trực tiếp nhờ hàm **cholesky()** với những giá trị $G(jn\Omega_\lambda)$, n=0,1, … , 2M cũng như bộ tham số $a_i^{(k-1)}$, i=1, … , n_a đã có từ bước lặp thứ k–1 trước đó.

Ta đi đến thuật toán lặp xác định bộ tham số tối ưu $b_0, \ldots, b_{n_b}, a_1, \ldots, a_{n_a}$ (sai lệch đầu ra nhỏ nhất) cho mô hình đối tượng không chứa thành phần tích phân từ những giá trị $G(jn\Omega_\lambda)$, n=0,1, … , 2M đã có (chẳng hạn như nhờ phương pháp nhận dạng bị động mô hình không tham số đã trình bày ở chương 2) như sau:

1) Xác định bộ tham số khởi phát $b_0^{(0)}, b_1^{(0)}, \ldots, b_{n_b}^{(0)}, a_1^{(0)}, a_2^{(0)}, \ldots, a_{n_a}^{(0)}$ từ $G(jn\Omega_\lambda)$, n=0,1, … , 2M (có thể sử dụng một phần hàm **par()** không có công đoạn gọi **nonpar()** để tính $G(jn\Omega_\lambda)$, n=0,1, … , 2M từ $\{u_k\}$, $\{y_k\}$, k=1, … , N–1).

2) Thực hiện lần lượt các bước sau với k=1, 2, …

   a) Tính $C = (W^{(k-1)}U)^H (W^{(k-1)}U)$ và $\underline{k} = (W^{(k-1)}U)^H W^{(k-1)} \underline{g}$

   b) Gọi hàm **cholesky()** để tính $b_0^{(k)}, b_1^{(k)}, \ldots, b_{n_b}^{(k)}, a_1^{(k)}, a_2^{(k)}, \ldots, a_{n_a}^{(k)}$.

   c) Nếu sai số $\sum_i \left(\left| b_i^{(k-1)} - b_i^{(k)} \right| + \left| a_i^{(k-1)} - a_i^{(k)} \right| \right) < \varepsilon$ với ε là một số dương đủ nhỏ cho trước thì dừng thuật toán và cho ra kết quả $b_0 = b_0^{(k)}, \ldots, b_{n_b} = b_{n_b}^{(k)}, a_1 = a_1^{(k)}, \ldots, a_{n_a} = a_{n_a}^{(k)}$.

Nếu như biết chắc rằng bộ tham số mô hình phải xác định với ít nhất hai vòng lặp thì ta có thể bắt đầu tại ngay bước 2, trong đó giá trị khởi phát $b_0^{(0)}, b_1^{(0)}, \ldots, b_{n_b}^{(0)}, a_1^{(0)}, a_2^{(0)}, \ldots, a_{n_a}^{(0)}$ chỉ cần chọn sao cho có được $w_n^{(0)}$=1, $\forall\, n$. Ví dụ như với

$$a_1^{(0)} = \ldots = a_{n_a}^{(0)} = 0$$

và $b_0^{(0)}, b_1^{(0)}, \ldots, b_{n_b}^{(0)}$ là tùy ý.

Để tiện cho việc cài đặt thuật toán trên, nhất là tại bước 2a) khi phải xác định C cũng như $\underline{k}$ ta có thể gọi $\widetilde{U} = W^{(k-1)}U$ và $\widetilde{u}_{qn}$ là các phần tử của $\widetilde{U}$, q=1, … , 2M+1, n=1, … , n_a+n_b+1. Vậy thì

$$\widetilde{u}_{qn} = \begin{cases} \dfrac{[j(q-1)\Omega_\lambda]^{n-1}}{w_{q-1}^{(k-1)}} & \text{khi} \quad 1 \le n \le n_b+1 \\ \dfrac{[j(q-1)\Omega_\lambda]^{n-n_b-1}}{w_{q-1}^{(k-1)}} G(j(q-1)\Omega_\lambda) & \text{khi} \quad n_b+1 < n \le n_a+n_b+1 \end{cases}$$

trong đó $w_0^{(k-1)}$ =1. Nếu gọi tiếp C=$(W^{(k-1)}U)^H(W^{(k-1)}U)$ và c_{mn}, m, n=1, … , n_a+n_b+1 là các phần tử của nó sẽ được

$$c_{11} = 1 + \sum_{q=1}^{2M} \frac{1}{\left|w_q^{(k-1)}\right|^2}$$

và

$$c_{mn} = \sum_{q=1}^{2M} \left(\overline{\widetilde{u}}_{qm}\widetilde{u}_{qn}\right)$$

$$= \begin{cases} (-1)^{n-1} \sum_{q=1}^{2M} \dfrac{(-jq\Omega_\lambda)^{m+n-2}}{\left|w_q^{(k-1)}\right|^2} & \text{khi} \quad 1 < n \le m \le n_b+1 \\ (-1)^{n-1} \sum_{q=1}^{2M} \dfrac{(-jq\Omega_\lambda)^{m+n-2-n_b}}{\left|w_q^{(k-1)}\right|^2} \overline{G(jq\Omega_\lambda)} & \text{khi} \quad 1 < n \le n_b+1 < m \\ (-1)^{n-n_b-1} \sum_{q=1}^{2M} \dfrac{(-jq\Omega_\lambda)^{m+n-2-2n_b}}{\left|w_q^{(k-1)}\right|^2} \left|G(jq\Omega_\lambda)\right|^2 & \text{khi} \quad n_b+1 < n \le m \end{cases}$$

Cũng như vậy, nếu ta tiếp tục gọi k_n, n = 1, … , n_a+n_b+1 là các phần tử của $\underline{k}$ thì

$$k_n = \sum_{q=1}^{2M+1} \left[\overline{\widetilde{u}}_{qn} \frac{G(j(q-1)\Omega_\lambda)}{w_{q-1}^{(k-1)}} \right]$$

$$k_n = \begin{cases} \sum_{q=0}^{2M} (-jq\Omega_\lambda)^{n-1} \dfrac{G(jq\Omega_\lambda)}{\left|w_q^{(k-1)}\right|^2} & \text{khi} \quad n \le n_b+1 \\ \sum_{q=0}^{2M} (-jq\Omega_\lambda)^{n-n_b-1} \left| \dfrac{G(jq\Omega_\lambda)}{w_q^{(k-1)}} \right|^2 & \text{khi} \quad n_b+1 < n \end{cases}$$

Sử dụng các công thức xác định C và $\underline{k}$ vừa nêu trên, ta tiến hành việc cài đặt thuật toán nhận dạng b_0 , … , b_{n_b} , a_1 , … , a_{n_a} cho đối tượng không chứa thành phần tích phân từ những giá trị $G(jn\Omega_\lambda)$, n=0,1, … , $2M$ của nó bằng ngôn ngữ lập trình C sẽ có được hàm **itpar()** cho dưới đây. Hàm **itpar()** này có các biến hình thức sau:

a) Con trỏ **G** chỉ đầu mảng phức **G[]** chứa các giá trị $G(jn\Omega_\lambda)$, n=0,1, … , $2M$.

b) Con trỏ **x** chỉ đầu mảng phức **x[]** chứa các tham số có từ vòng lặp thứ k–1 trước đó theo thứ tự:

x[0]=$b_0^{(k-1)}$, … , **x[nb]**=$b_{n_b}^{(k-1)}$, **x[nb+1]**=$-a_1^{(k-1)}$, … , **x[na+nb]**=$-a_{n_a}^{(k-1)}$

Sau khi hàm thực hiện xong, các giá trị tham số mới sẽ được ghi lại vào mảng này cũng theo đúng thứ tự đó, tức là:

x[0]=b_0^k , … , **x[nb]**=$b_{n_b}^k$, **x[nb+1]**=$-a_1^k$, … , **x[na+nb]**=$-a_{n_a}^k$.

c) Số thực **Om** chứa giá trị Ω_λ.

d) Số nguyên **M2** chứa độ dài dãy $\{G(jn\Omega_\lambda)\}$, tức là **M2**=$2M$.

e) Số nguyên **na** chứa bậc đa thức mẫu số n_a của mô hình.

f) Số nguyên **nb** chứa bậc đa thức tử số n_b của mô hình.

Hàm trả về giá trị báo lỗi –1 nếu $n_a<n_b$ hoặc $i > 0$ thông báo C suy biến với phần tử thứ ii không phải là số dương. Trong trường hợp không có lỗi, hàm trả về giá trị 0. Hàm không làm thay đổi nội dung mảng **G[]**.

```
int itpar(complex *G,complex *x,double Om,int M2,int na,int nb)
{  int n,m=na+nb+1,q,L,mn;
   double *s,*w;
   if(nb>na) return(-1);
   complex cik,*c,im=complex(1.);
   c=new complex[m*(m+1)/2+1];
   s=new double[M2];
   w=new double[M2];
   c[0]=complex(0.,Om);
   for(n=1;n<na;n++) c[n]=c[n-1]*complex(0.,Om);
   for(n=0;n<M2;n++)
   {  cik=complex(1.);
      for(m=0;m<na;m++) cik+=c[m]*pow(n+1,m+1)*real(x[nb+1+m]);
      w[n]=1./(pow(real(cik),2)+pow(imag(cik),2));
   }
   c[0]=complex(1.);
   for(n=0;n<M2;n++) c[0]+=complex(w[n]);
```

```
    x[0]=G[0];
    for(q=0;q<M2;q++)
    {   s[q]=1.;
        x[0]=x[0]+G[q+1]*w[q];
    }
    for(mn=1;mn<=2*na;mn++)
    {
        for(q=0;q<M2;q++) s[q]=s[q]*Om*(q+1);
        im=im*complex(0.,-1);
        for(m=1;m<=na+nb+1;m++)
            for(n=1;n<=m;n++)
            {
                if(m<=nb+1) L=n+m-2;
                else L=(n<=nb+1)? m-nb+n-2:m-2*nb+n-2;
                if(L!=mn) continue;
                cik=complex(0.);
                L=(n-1)%2;
                if(m<=nb+1) for(q=0;q<M2;q++) cik+=im*s[q]*w[q];
            else if(n<=nb+1)
                    for(q=0;q<M2;q++) cik+=im*conj(G[q+1])*s[q]*w[q];
                else
                {   for(q=0;q<M2;q++)
                        cik+=im*s[q]*pow(abs(G[q+1]),2.)*w[q];
                    L=(n-nb-1)%2;
                }
                c[m*(m-1)/2-1+n]=(L==0)? cik:-cik;
        }
        if(mn>na) continue;
        if(mn<=nb)
        {   x[mn]=complex(0.);
            for(q=0;q<M2;q++) x[mn]+=im*G[q+1]*s[q]*w[q];
        }
        x[nb+mn]=complex(0.);
        for(q=0;q<M2;q++)
            x[nb+mn]+=im*s[q]*pow(abs(G[q+1]),2.)*w[q];
    }
    L=cholesky(na+nb+1,c,x);
    delete [] c; delete [] w; delete [] s;
    return (L);
}
```

Ví dụ 2: Hàm **itpar()** trên đây có thể được dùng thay cho hàm **par()** đã giới thiệu ở mục 3.2.2 để xác định tham số mô hình b_0 , … , b_{n_b} , a_1 , … , a_{n_a} cho những đối tượng không chứa thành phần tích phân một cách trực tiếp từ dãy các giá trị $\{G(jn\Omega_\lambda)\}$ chứ

không phải từ dãy giá trị tín hiệu vào/ra $\{u_k\}$, $\{y_k\}$ bằng cách gọi hàm **itpar()** với đầu vào **x[n]**= 0, n=0,1, … , n_a+n_b . Ví dụ với các lệnh

```
void main()
{  complex *G,*x,b0,b1,a1,a2;
   int i;
   b0=complex(2.); b1=complex(0.5);
   a1=complex(0.8); a2=complex(2.);
   G=new complex [1000]; x=new complex [7];
   G[0]=complex(2.);
   for(i=1;i<1000;i++) G[i]=(b0+b1*i))/(complex(1.)+b1*i-b2*i*i);
   for(i=0;i<7;i++) x[i]=complex(0.);
   if(itpar(G,x,1.,999,2,1)==0)
     for(i=0;i<4;i++)
      printf("x[%d]=(%f,%f)\n",i,real(x[i]),imag(x[i]));
   delete [] G; delete [] x;
   getch();
}
```

sẽ có

```
   x[0]=(2.0,0.0)     x[1]=(0.5,0.0)
   x[2]=(-0.8,0.0)    x[3]=(-2.0,0.0).
```

❑

Ví dụ 3: Quay lại xét ví dụ 1 ban đầu là đối tượng tuyến tính cần nhận dạng có mô hình

$$G_M(s) = \frac{b_0 + b_1 s}{1 + a_1 s + a_2 s^2},$$

với

b_0 =2 , b_1 =0,5 , a_1 = 0,8 và a_2 = 2.

Từ dãy $\{G(jn\Omega_\lambda)\}$, n=0, 1, … , 999, Ω_λ = 1,5s^{-1} bị lẫn nhiễu phân bố chuẩn ta sử dụng thuật toán lặp với các bước lặp k=1,2,3,4 chẳng hạn như nhờ lệnh

```
for(i=0;i<7;i++) x[i]=complex(0.);
for(k=0;k<3;k++)
{
   i=itpar(G,x,1.5,999,2,1);
   printf("k=%d\titpar()=%d\n",k,i);
   if(i==0)
      for(i=0;i<4;i++)
         printf("x[%d]=(%f,%f)\n",i,real(x[i]),imag(x[i]));
}
```

sẽ thu được kết quả:

k	b_0 =2	b_1 =0,5	a_1 = 0,8	a_2 = 2
1	0.3876989	0.121715708	0.35801470	0.174858337
2	1.9936758	0.47759602	0.80031561	1.79909552
3	1.9999577	0.49980766	0.80189254	1.99115383
4	1.9999984	0.49802213	0.80144393	1.99111318

Thuật toán lặp trên đây cũng có thể được cải biên cho đối tượng chứa thành phần tích phân I nhưng không có thành phần vi phân D với mô hình rút gọn

$$G_M(s) = \frac{1 + b_1 s + \cdots + b_{n_b} s^{n_b}}{a_0 + a_1 s + \cdots + a_{n_a} s^{n_a}}.$$

Trước hết ta xem quan hệ xấp xỉ

$$G_M(jn\Omega_\lambda) \approx G(jn\Omega_\lambda) \qquad \Leftrightarrow \qquad \frac{1 + \sum_{i=1}^{n_b} b_i (jn\Omega_\lambda)^i}{\sum_{i=0}^{n_a} a_i (jn\Omega_\lambda)^i} \approx \frac{\widetilde{S}_{uy}(jn\Omega_\lambda)}{\widetilde{S}_u(n\Omega_\lambda)}$$

như là một sai lệch tại bước lặp thứ k

$$e_n = \frac{\widetilde{S}_u(n\Omega_\lambda) - \left(\widetilde{S}_{uy}(n\Omega_\lambda) \sum_{i=0}^{n_a} a_i^{(k)} (jn\Omega_\lambda)^i - \widetilde{S}_u(n\Omega_\lambda) \sum_{i=1}^{n_b} b_i^{(k)} (jn\Omega_\lambda)^i \right)}{\widetilde{S}_u(n\Omega_\lambda) \sum_{i=0}^{n_a} a_i^{(k-1)} (jn\Omega_\lambda)^i} \neq 0$$

rồi tìm

$$b_1^{(k)}, \ldots, b_{n_b}^{(k)}, a_0^{(k)}, a_1^{(k)}, \ldots, a_{n_a}^{(k)}$$

từ dãy

$$\{ \widetilde{S}_u(n\Omega_\lambda) \}, \{ \widetilde{S}_{uy}(jn\Omega_\lambda) \}$$

cũng như từ $a_i^{(k-1)}$, i=0, … , n_a, trong đó $a_0^{(0)}$ =1, $a_1^{(0)} = \ldots = a_{n_a}^{(0)} = 0$ sao cho

$$Q = \sum_{n=0}^{2M} |e_n|^2$$

đạt giá trị nhỏ nhất. Khi đó ta sẽ được

$$\underbrace{\left(\widetilde{W}^{(k-1)} L\right)^H \left(\widetilde{W}^{(k-1)} L\right)}_{C} \cdot \underline{x}_k = \underbrace{\left(\widetilde{W}^{(k-1)} L\right)^H \widetilde{W}^{(k-1)} \underline{s}}_{\underline{k}} \tag{3.77}$$

với

$$\underline{x}_k = \begin{pmatrix} a_0^{(k)} \\ \vdots \\ a_{n_a}^{(k)} \\ -b_1^{(k)} \\ \vdots \\ -b_{n_b}^{(k)} \end{pmatrix}, \ \underline{s} = \begin{pmatrix} \widetilde{S}_u(0) \\ \widetilde{S}_u(\Omega_\lambda) \\ \vdots \\ \widetilde{S}_u(2M\Omega_\lambda) \end{pmatrix}, \quad L = (l_{nk}), \qquad \widetilde{W}^{(k-1)} = (\widetilde{w}_{mn}^{(k-1)})$$

$$l_{nk} = \begin{cases} (jn\Omega_\lambda)^k \tilde{S}_{uy}(jn\Omega_\lambda) & \text{nếu} \quad k \le n_a \\ (jn\Omega_\lambda)^{k-n_a} \tilde{S}_u(jn\Omega_\lambda) & \text{nếu} \quad n_a < k \end{cases},$$

$$\tilde{w}_{mn}^{(k-1)} = \begin{cases} 0 & \text{nếu } m \neq n \\ \dfrac{1}{\tilde{S}_u(n\Omega_\lambda) \sum\limits_{i=0}^{n_a} a_i^{(k-1)} (jn\Omega_\lambda)^i} & \text{nếu} \quad m = n \end{cases}$$

$m, n = 0, 1, , \ldots, 2M$ và $k = 0, 1, , \ldots, n_a + n_b$.

Như vậy ta sẽ có được thuật toán xác định bộ tham số tối ưu $\underline{x}_k$ từ dãy giá trị tín hiệu $\{u_k\}$, $\{y_k\}$, $k=1, \ldots, N-1$ đo được trong khoảng thời gian quan sát $[0, NT_a)$ với thời gian trích mẫu T_a cũng như từ $a_0^{(k-1)}, a_1^{(k-1)}, \ldots, a_{n_a}^{(k-1)}$, trong đó

$$a_0^{(0)} = 1, \ a_1^{(0)} = \ldots = a_{n_a}^{(0)} = 0,$$

như sau:

1) Sử dụng hàm **spec()** tính $\widetilde{S}_u(n\Omega_\lambda)$, $\widetilde{S}_{uy}(n\Omega_\lambda)$.

2) Tính $C = \left(\widetilde{W}^{(k-1)} L\right)^H \left(\widetilde{W}^{(k-1)} L\right)$ và $\underline{k} = \left(\widetilde{W}^{(k-1)} L\right)^H \widetilde{W}^{(k-1)} \underline{s}$.

3) Gọi hàm **cholesky()** để giải phương trình (3.77) theo $\underline{x}_k$.

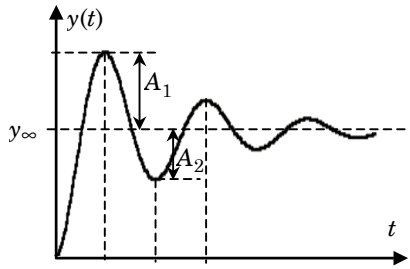
Câu hỏi ôn tập và bài tập

1. Giả sử rằng khi kích thích tại đầu vào của đối tượng cần nhận dạng một tín hiệu $u(t) = u_0 1(t)$ người ta thu được ở đầu ra đường thực nghiệm $y(t)$ có dạng một khâu dao động tắt dần (hình 3.30) với mô hình tham số thuộc lớp

$$G(s) = \frac{kq^2}{s^2 + 2qDs + q^2}, \ 0 < D < 1$$

Gọi y_∞ là giá trị giới hạn của $y(t)$ khi $t\to\infty$ và A_i , i = 0, 1, … là khoảng cách của các điểm cực trị tới đường $y=y_\infty$. Chứng minh rằng

$$D^{-1} = \sqrt{1+\frac{\pi^2}{\ln^2\left|\frac{A_{i+1}}{A_i}\right|}}$$



Hình 3.30: Cho bài tập 1.

2. Với kết quả của bài tập 1, hãy xây dựng thuật toán nhận dạng các tham số k, D, q của đối tượng có mô hình (3.54) từ tọa độ các điểm cực đại của đường thực nghiệm $y(t)$ thu được khi kích thích đối tượng bằng tín hiệu $u(t)= u_0 1(t)$ ở đầu vào. Cài đặt thuật toán vừa xây dựng thành chương trình viết trên C.
3. Chứng minh rằng ma trận U cho trong công thức (3.65) khi $\Omega_\lambda\neq 0$ và $G(jn\Omega_\lambda)$ không phải là hằng số với mọi n=0,1, … , M và $M\geq n_a+n_b$ sẽ có hạng là n_a+n_b+1.
4. Chứng minh rằng nếu ma trận U có m hàng, n cột với $m > n$ và các vector cột là độc lập tuyến tính thì $U^H U$ sẽ không suy biến với hạng là n. Hãy chỉ rằng $U^H U$ và $(U^H U)^{-1}$ là các ma trận xác định dương.
5. Điều gì sẽ xảy ra khi sử dụng thuật toán nhận dạng tham số mô hình theo phương pháp bị động đã trình bày trong các mục 3.2.2 và 3.2.3 (chẳng hạn như với hàm **`par()`** hay **`itpar()`**) nếu tín hiệu đầu vào có dải tần số rất hẹp, ví dụ tín hiệu hình sin hoặc cos.

4 NHẬN DẠNG THAM SỐ MÔ HÌNH ARMA

4.1 Đặt vấn đề

4.1.1 Phát biểu bài toán nhận dạng mô hình ARMA

Cùng với sự bùng nổ ứng dụng lớp mô hình tham số không liên tục biểu diễn mối quan hệ vào–ra cho một hệ tuyến tính, tham số hằng trong điều khiển tự động, ngành nhận dạng cũng chuyển bước từ nhận dạng mô hình liên tục trước đây mà trọng tâm chủ yếu là xây dựng hàm trọng lượng hoặc hàm đặc tính tần biên–pha dưới dạng một dãy số (phức) sang lĩnh vực *nhận dạng mô hình rời rạc*.

Nội dung chính của chương này là trình bày những phương pháp nhận dạng tham số K, $\underline{a} = \begin{pmatrix} a_1 \\ \vdots \\ a_{n_a} \end{pmatrix}$, $\underline{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_{n_b} \end{pmatrix}$ cho mô hình rời rạc ARMA (*Autoregressive moving average*)

$$G(z) = \frac{Y(z)}{U(z)} = K \frac{1 + b_1 z^{-1} + \cdots + b_{n_b} z^{-n_b}}{1 + a_1 z^{-1} + \cdots + a_{n_a} z^{-n_a}}, \tag{4.1}$$

trên cơ sở quan sát, đo tín hiệu vào $u(t)$ và ra $y(t)$ sao cho sai lệch giữa mô hình và đối tượng là nhỏ nhất. Với những kiểu mô tả sai lệch khác nhau sẽ có các phương pháp nhận dạng khác nhau.

Các phương pháp này được chia ra làm hai loại chính:

– loại nhận dạng active (chủ động). Tín hiệu đầu vào $u(t)$ được chọn là tín hiệu ồn trắng có giá trị mật độ phổ bằng 1, tức là

$$m_u = 0 \text{ và } S_u(\omega) = 1. \tag{4.2}$$

– và loại nhận dạng passive (bị động).

Đặc biệt, khi $n_b = 0$ mô hình (4.1) trở thành

$$G(z) = \frac{K}{1 + a_1 z^{-1} + \cdots + a_{n_a} z^{-n_a}} \tag{4.3}$$

và được gọi là mô hình AR (*Autoregressive*) của đối tượng.

Ngược lại khi n_a= 0 thì mô hình (4.1) trở thành

$$G(z)= K(1 + b_1 z^{-1} + \ldots + b_{n_b} z^{-n_b}) \tag{4.4}$$

có tên gọi là mô hình MA (*Moving average*).

Chuyển (4.1) sang miền thời gian, sẽ có được mô hình tương đương dạng phương trình sai phân sau:

$$y_n + \sum_{k=1}^{n_a} a_k y_{n-k} = K\left(u_n + \sum_{k=1}^{n_b} b_k u_{n-k} \right) \tag{4.5}$$

Hoàn toàn tương tự, dạng thời gian của mô hình AR là

$$y_n + \sum_{k=1}^{n_a} a_k y_{n-k} = K u_n \tag{4.6}$$

và của mô hình MA là

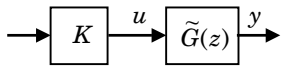
$$y_n = K\left(u_n + \sum_{k=1}^{n_b} b_k u_{n-k} \right) \tag{4.7}$$

4.1.2 Chuyển thành bài toán tương đương có hệ số khuếch đại của mô hình bằng 1

Để đơn giản, ta chuyển bài toán vừa nêu về dạng mô hình có hệ số khuếch đại bằng 1 (hình 4.1), bằng cách tách $G(z)$ thành hai khâu K và $\widetilde{G}(z)$ mắc nối tiếp. Khâu $\widetilde{G}(z)$ khi đó sẽ có hàm truyền đạt

$$\widetilde{G}(z) = \frac{Y(z)}{\widetilde{U}(z)} = \frac{1 + b_1 z^{-1} + \cdots + b_{n_b} z^{-n_b}}{1 + a_1 z^{-1} + \cdots + a_{n_a} z^{-n_a}}, \tag{4.8}$$

Tín hiệu đầu vào qua khâu khuếch đại K cũng là tín hiệu ngẫu nhiên egodic có giá trị trung bình bằng 0 và giá trị mật độ phổ bằng K. Với sự biến đổi mà ở đó hệ số khuếch đại của mô hình được chuyển thành giá trị mật độ phổ của tín hiệu đầu vào, ta có dạng tương đương của bài toán chuẩn cho việc nhận dạng chủ động tham số mô hình ARMA. Bài toán tương đương này phát biểu như sau: Trên cơ sở quan sát, đo tín hiệu ra $y(t)$, được dãy $\{y_k\}$, hãy xác định các vector tham số $\underline{a} = \begin{pmatrix} a_1 \\ \vdots \\ a_{n_a} \end{pmatrix}$, $\underline{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_{n_b} \end{pmatrix}$ của mô hình $\widetilde{G}(z)$ và giá trị mật độ phổ K của tín hiệu đầu vào $u(t)$ sao cho sai lệch giữa mô hình và đối tượng là nhỏ nhất, trong đó



Hình 4.1: Chuyển thành bài toán có mô hình với hệ số khuếch đại 1.

$$m_u = 0 \quad \text{và} \quad S_u(\omega) = K. \tag{4.9}$$

Tương ứng, mô hình $\widetilde{G}(z)$ trong miền thời gian có dạng

$$y_n + \sum_{k=1}^{n_a} a_k y_{n-k} = u_n + \sum_{k=1}^{n_b} b_k u_{n-k} \tag{4.10}$$

Hoàn toàn tương tự, dạng thời gian của mô hình AR tương đương là

$$y_n + \sum_{k=1}^{n_a} a_k y_{n-k} = u_n \tag{4.11}$$

và của mô hình MA tương đương là

$$y_n = u_n + \sum_{k=1}^{n_b} b_k u_{n-k} \tag{4.12}$$

4.2 Nhận dạng chủ động tham số mô hình AR

Nhiệm vụ của bài toán nhận dạng chủ động là khi đối tượng được kích thích chủ động bằng tín hiệu ồn trắng $u(t)$ thỏa mãn (4.9), hãy xác định các vector tham số $\underline{a}$, K của mô hình (4.11) từ dãy giá trị $\{y_k\}$, $k = 0,1, \ldots, N-1$ đo được của tín hiệu ra sao cho sai lệch giữa mô hình và đối tượng là nhỏ nhất.

4.2.1 Phương pháp Yule–Walker

Trước hết, ta nhân cả hai vế của mô hình (4.11) với y_{n-m} về phía trái

$$y_{n-m} y_n + \sum_{k=1}^{n_a} a_k y_{n-m} y_{n-k} = y_{n-m} u_n$$

sau đó lập giá trị trung bình theo công thức (1.30) và (1.33) cho cả hai vế, sẽ có

$$r_y(mT_a) + \sum_{k=1}^{n_a} a_k r_y((m-k)T_a) = r_{yu}(mT_a)$$

Nếu gọi $g_m = g(mT_a)$ là giá trị hàm hàm trọng lượng của mô hình AR (4.11) tại các điểm trích mẫu mT_a, $m = -\infty, \ldots, \infty$ thì do đối tượng có hệ số khuếch đại bằng 1 nên

$$g_m = \begin{cases} 1 & \text{khi} \quad m = 0 \\ 0 & \text{khi} \quad m < 0 \end{cases}$$

Ngoài ra, theo giả thiết (4.9) còn có

$$r_u(mT_a)=\begin{cases}K & \text{khi } m=0\\ 0 & \text{khi } m\neq 0\end{cases}$$

bởi vậy từ điều hiển nhiên

$$r_{yu}(mT_a)=r_{uy}(-mT_a)=r_u(-mT_a)*g_{-m}=\sum_{k=-\infty}^{\infty}r_u(-kT_a)g_{-m-k}=Kg_{-m}$$

ta được

$$r_{yu}(mT_a)=\begin{cases}K & \text{khi } m=0\\ 0 & \text{khi } m>0\\ Kg_{-m} & \text{khi } m<0\end{cases}$$

Suy ra

$$r_y(mT_a)+\sum_{k=1}^{n_a}a_k r_y((m-k)T_a)=\begin{cases}K & \text{khi } m=0\\ 0 & \text{khi } m>0\\ Kg_{-m} & \text{khi } m<0\end{cases}. \tag{4.13}$$

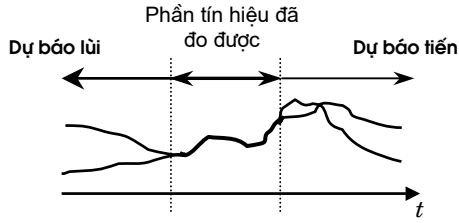
Viết lại (4.13) lần lượt cho m=0,1, … , n_a dưới dạng ma trận ta đi đến

$$\underbrace{\begin{pmatrix} r_y(0) & r_y(-T_a) & \cdots & r_y(-n_aT_a)\\ r_y(T_a) & r_y(0) & \cdots & r_y((-n_a+1)T_a)\\ \vdots & & & \\ r_y(n_aT_a) & r_y((n_a-1)T_a) & \cdots & r_y(0)\end{pmatrix}}_{H_{n_a}}\cdot\begin{pmatrix}1\\ \underline{a}\end{pmatrix}=\begin{pmatrix}K\\ 0\\ \vdots\\ 0\end{pmatrix} \tag{4.14}$$

Đây là phương trình xác định vector tham số $\underline{a}$, K từ việc đo tín hiệu ra $y(t)$ và có tên gọi là *phương trình Yule–Walker*.

4.2.2 Sai số dự báo tuyến tính của phương pháp Yule–Walker

Ưu điểm cơ bản của phương pháp Yule–Walker là với tham số mô hình $\underline{a}$, K đã xác định được, giá trị trung bình của bình phương sai lệch dự báo tuyến tính nhỏ nhất. Ưu điểm này đã được Burg sử dụng để xây dựng pháp truy hồi rất có ý nghĩa trong ứng dụng thực tế và sẽ được trình bày sau trong mục 4.2.4.



Hình 4.2: Giải thích khái niệm phương pháp dự báo tiến và dự báo lùi.

Hình 4.2 minh họa khái niệm dự báo tuyến tính. Giả sử qua việc quan sát tín hiệu ta đã có $y(t)$ trong khoảng quan sát $[T_1, T_2]$. Những phương pháp dự báo cho ra kết quả xấp xỉ của $y(t)$ khi $t>T_2$ được gọi là

phương pháp dự báo vượt trước hay dự báo tiến. Ngược lại, phương pháp cho ra kết quả gần đúng $y(t)$ khi $t<T_1$ được gọi là dự báo lùi.

Những phương pháp dự báo có dạng tổ hợp tuyến tính từ các giá trị tín hiệu quan sát được thì được gọi là dự báo tuyến tính. Các công thức dự báo tuyến tính tương ứng với mô hình AR gồm:

1) dự báo tuyến tính tiến:

$$y_n^f = -\sum_{k=1}^{n_a} a_k y_{n-k} \tag{4.15}$$

cho phép xác định y_n^f được xem như là giá trị gần đúng của y_n từ n_a các giá trị đã có trước đó $y_{n-1},\ldots,y_{n-n_a}$. Số mũ f trong y_n^f không có ý nghĩa lũy thừa mà đơn giản chỉ là ký hiệu nói rằng giá trị xấp xỉ đó được xác định theo phương pháp dự báo tuyến tính tiến mà đôi khi trong một vài tài liệu khác nhau còn gọi là phương pháp dự báo tuyến tính vượt trước (*forward*).

2) dự báo tuyến tính lùi:

$$y_n^b = -\sum_{k=1}^{n_a} a_k y_{n+k} \tag{4.16}$$

cho phép xác định y_n^b được xem như là giá trị xấp xỉ của y_n từ n_a các giá trị đã có sau đó $y_{n+1},\ldots,y_{n+n_a}$. Số mũ b trong y_n^b là ký hiệu chỉ rằng giá trị xấp xỉ được tính theo phương pháp dự báo tuyến tính lùi (*backward*).

Định lý 4.1: Với các tham số $\underline{a}$, K của mô hình AR xác định theo thuật toán Yule–Walker, phương pháp dự báo tuyến tính tiến (4.15) là phương pháp có giá trị trung bình bình phương các sai lệch nhỏ nhất.

Chứng minh:

Trước hết lập sai lệch $e_n^f = y_n - y_n^f$. Vậy giá trị trung bình của bình phương sai lệch này sẽ là

$$\begin{aligned} M\left[|e_n^f|^2\right] &= M\left[\left|y_n + \sum_{k=1}^{n_a} a_k y_{n-k}\right|^2\right] \\ &= M\left[y_n^2\right] + 2M\left[y_n \sum_{k=1}^{n_a} a_k y_{n-k}\right] + M\left[\left(\sum_{k=1}^{n_a} a_k y_{n-k}\right)\left(\sum_{l=1}^{n_a} a_l y_{n-l}\right)\right] \\ &= r_y(0) + 2\sum_{k=1}^{n_a} a_k r_y(kT_a) + \sum_{k=1}^{n_a}\sum_{l=1}^{n_a} a_k a_l r_y((k-l)T_a) \end{aligned}$$

Sử dụng ký hiệu

$$\underline{r} = \begin{pmatrix} r_y(T_a) \\ \vdots \\ r_y(n_a T_a) \end{pmatrix} \quad \text{và} \quad H_{n_a-1} = \begin{pmatrix} r_y(0) & r_y(-T_a) & \cdots & r_y((n_a-1)T_a) \\ r_y(T_a) & r_y(0) & \cdots & r_y((n_a-2)T_a) \\ \vdots & & & \\ r_y((n_a-1)T_a) & {}_y((n_a-2)T_a) & \cdots & r_y(0) \end{pmatrix}$$

đẳng thức trên viết được thành

$$\begin{aligned} M\left[|e_n^f|^2\right] &= r_y(0) + \underline{r}^T \underline{a} + \underline{a}^T \underline{r} + \underline{a}^T H_{n_a-1} \underline{a} \\ &= r_y(0) - \underline{r}^T H_{n_a-1}^{-1} \underline{r} + \left(\underline{r} + H_{n_a-1}\underline{a}\right)^T H_{n_a-1}^{-1} \left(\underline{r} + H_{n_a-1}\underline{a}\right) \end{aligned}$$

Riêng biểu thức thứ 3 chứa vector tham số $\underline{a}$ phải tìm. Bởi vậy khi $\underline{a}$ làm cho biểu thức này có giá trị nhỏ nhất, nó cũng sẽ làm cho $M[|e_f|^2]$ có giá trị nhỏ nhất. Nhưng ma trận H_{n_a-1} xác định không âm, nên biểu thức thứ 3 sẽ có giá trị nhỏ nhất khi

$$\underline{r} + H_{n_a-1}\underline{a} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} \underline{r} & H_{n_a-1} \end{pmatrix} \cdot \begin{pmatrix} 1 \\ \underline{a} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \tag{4.17}$$

So sánh với (4.14) ta thấy ngay được là nghiệm của (4.14) thỏa mãn (4.17) vì

$$H_{n_a} = \begin{pmatrix} r_y(0) & \underline{r}^T \\ \underline{r} & H_{n_a-1} \end{pmatrix} \tag{4.18}$$

và đó là điều phải chứng minh. ❑

Một cách hoàn toàn tượng tự, ta cũng sẽ chỉ ra được:

Định lý 4.2: Với các tham số $\underline{a}$, K của mô hình AR xác định theo thuật toán Yule–Walker, phương pháp dự báo tuyến tính lùi (4.16) là phương pháp có giá trị trung bình bình phương các sai lệch nhỏ nhất.

Trên đây là hai định lý khẳng định hai ưu điểm trong kỹ thuật dự báo tuyến tính của các hệ số a_1, a_2, … , a_{n_a} và K của mô hình AR (4.3) được xác định theo phương pháp Yule–Walker. Hai ưu điểm này đã được Burg sử dụng để xây dựng thuật toán truy hồi giải phương trình (4.14) sẽ giới thiệu sau ở mục 4.2.4. Bên cạnh hai ưu điểm vừa nêu, nghiệm của phương trình Yule–Walker còn có một ưu điểm thứ ba rất lý thú cho việc đánh giá lượng thông tin nguồn phát. Vì ưu điểm này không liên quan đến công việc chính của chúng ta là nhận dạng đối tượng điều khiển nên ở đây nó chỉ được giới thiệu dưới dạng một hệ quả không có chứng minh để tham khảo thêm.

Hệ quả 4.1: Trong lớp tất cả các mật độ phổ tín hiệu có dạng

$$S_y(\omega) = \frac{K}{1 + \sum_{k=1}^{n_a} a_k e^{-j\omega k T_a}}$$

thu được từ dãy các giá trị tín hiệu $\{y_k\}$ thì mật độ phổ có các hệ số $a_1, a_2, \ldots, a_{n_a}$, K xác định theo Yule–Walker sẽ có lượng thông tin (*entropie*) nhiều nhất về nguồn phát tín hiệu ngẫu nhiên $y(t)$. Nói cách khác với các hệ số đó sẽ có

$$H_1 = \int_{-\infty}^{\infty} \ln S_y(\omega) d\omega \rightarrow \max!$$

4.2.3 Giải phương trình Yule–Walker nhờ thuật toán Levinson

Ma trận H_{n_a} của (4.14) có những tính chất rất đặc biệt, đó là:

a) H_{n_a} là ma trận Toeplitz, tức là các phần tử tại hàng thứ i, cột j được xác định từ hiệu i–j và do đó người ta có thể xây dựng lại được toàn bộ ma trận chỉ từ các phần tử của hàng đầu tiên và cột đầu tiên.

b) H_{n_a} đối xứng qua đường chéo chính, tức là $H_{n_a}^T = H_{n_a}$. Kết hợp với H_{n_a} là ma trận *Toeplitz*, người ta chỉ cần các phần tử của hàng đầu tiên là đủ để có được toàn bộ ma trận.

c) H_{n_a} là ma trận xác định không âm. Điều này có thể suy ra từ tính chất (1.31) của hàm tự tương quan theo phương pháp quy nạp.

Tính chất b) và c) là điều kiện để áp dụng được thuật toán Cholesky và do đó ta có thể sử dụng được ngay hàm **`cholesky()`** đã giới thiệu ở chương 2 (mục 3.2.1) để giải phương trình 4.14 dưới dạng

$$H_{n_a} \begin{pmatrix} K^{-1} \\ K^{-1}\underline{a} \end{pmatrix} = \begin{pmatrix} \mathbf{1} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix}$$

Tuy nhiên cùng với tính chất a) việc cài đặt thuật toán giải phương trình (4.14) sẽ đơn giản hơn nhiều. Một thuật toán chuẩn để giải (4.14) theo phương pháp truy hồi thuộc về Levinson.

Thuật toán Levinson không giải trực tiếp phương trình Yule–Walker (4.14) bậc n_a để có các tham số $a_1, a_2, \ldots, a_{n_a}$ và K mà lần lượt giải (4.14) với những bậc i thấp hơn, bắt

đầu với i=1, sau đó cho i tăng dần đến n_a. Để tiện trong trình bày ta sẽ ký hiệu phương trình (4.14) cho bậc i có để ý đến tính đối xứng $H_i^T = H_i$ như sau:

$$\underbrace{\begin{pmatrix} r_y(0) & r_y(T_a) & \cdots & r_y(iT_a) \\ r_y(T_a) & r_y(0) & \cdots & r_y((i-1)T_a) \\ \vdots & \vdots & \ddots & \vdots \\ r_y(iT_a) & r_y((i-1)T_a) & \cdots & r_y(0) \end{pmatrix}}_{H_i} \cdot \begin{pmatrix} 1 \\ a_1[i] \\ \vdots \\ a_i[i] \end{pmatrix} = \begin{pmatrix} K_i \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{4.19}$$

nói cách khác, các tham số nay sẽ được ký hiệu bằng $a_1[i]$, $a_2[i]$, ... , $a_i[i]$, K_i để nhấn mạnh rằng chúng thuộc về phương trình Yule–Walker bậc i. Vấn đề đặt ra cho bài toán xây dựng thuật toán truy hồi là xác định i+1 tham số $a_1[i]$, $a_2[i]$, ... , $a_i[i]$, K_i của (4.19) từ i các tham số $a_1[i-1]$, $a_2[i-1]$, ... , $a_{i-1}[i-1]$, K_{i-1} của phương trình Yule–Walker bậc i–1 được giả thiết là đã biết:

$$\underbrace{\begin{pmatrix} r_y(0) & r_y(T_a) & \cdots & r_y((i-1)T_a) \\ r_y(T_a) & r_y(0) & \cdots & r_y((i-2)T_a) \\ \vdots & \vdots & \ddots & \vdots \\ r_y((i-1)T_a) & r_y((i-2)T_a) & \cdots & r_y(0) \end{pmatrix}}_{H_{i-1}} \cdot \begin{pmatrix} 1 \\ a_1[i-1] \\ \vdots \\ a_{i-1}[i-1] \end{pmatrix} = \begin{pmatrix} K_{i-1} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{4.20}$$

Trước hết ta chứng minh định lý sau:

Định lý 4.3: Nếu $\underline{a}[i] = \begin{pmatrix} a_1[i] \\ \vdots \\ a_i[i] \end{pmatrix}$ và K_i là nghiệm của (4.19) thì chúng cũng sẽ thỏa mãn

$$H_i \begin{pmatrix} a_i[i] \\ \vdots \\ a_1[i] \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_i \end{pmatrix} \tag{4.21}$$

Chứng minh:

Nhân cả hai vế của (4.19) với ma trận $J = \begin{pmatrix} 0 & \cdots & 1 \\ \vdots & \iddots & \vdots \\ 1 & \cdots & 0 \end{pmatrix}$ về phía trái ta có

$$JH_i \begin{pmatrix} 1 \\ \underline{a}[i] \end{pmatrix} = J \begin{pmatrix} K_i \\ \underline{0}_i \end{pmatrix}$$

Nếu để ý tiếp rằng

$$J\begin{pmatrix} K_i \\ \underline{0}_i \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_i \end{pmatrix}$$

và J, H_i là hai ma trận đối xứng, tức là tích của chúng cũng là một ma trận đối xứng

$$JH_i = (JH_i)^T = H_i J$$

ta sẽ được

$$H_i J\begin{pmatrix} 1 \\ \underline{a}[i] \end{pmatrix} = J\begin{pmatrix} K_i \\ \underline{0}_i \end{pmatrix} \quad \Leftrightarrow \quad H_i\begin{pmatrix} a_i[i] \\ \vdots \\ a_1[i] \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_i \end{pmatrix} \text{ (đ.p.c.m)} \qquad \square$$

Bây giờ ta sẽ xác định quan hệ truy hồi giữa các tham số của (4.19) và (4.20).

Định lý 4.4: Giữa vector các tham số $\underline{a}[i]=\begin{pmatrix} a_1[i] \\ \vdots \\ a_i[i] \end{pmatrix}$, K_i của (4.19) và $\underline{a}[i-1]=\begin{pmatrix} a_1[i-1] \\ \vdots \\ a_{i-1}[i-1] \end{pmatrix}$,

K_{i-1} của (4.20) có quan hệ truy hồi

a) $a_k[i] = a_k[i-1] + a_i[i]\, a_{i-k}[i-1]$ với $k = 1, 2, \ldots, i-1$ (4.22a)

b) $K_i = K_{i-1}\left(1 - a_i[i]^2\right)$ (4.22b)

Chứng minh:

Với ký hiệu

$$\underline{r}[i] = \begin{pmatrix} r_y(T_a) \\ \vdots \\ r_y(iT_a) \end{pmatrix}, \qquad \underline{r}_r[i] = \begin{pmatrix} r_y(iT_a) \\ \vdots \\ r_y(T_a) \end{pmatrix} = \begin{pmatrix} 0 & \cdots & 1 \\ \vdots & \iddots & \vdots \\ 1 & \cdots & 0 \end{pmatrix}\underline{r}[i]$$

$$\underline{a}[i] = \begin{pmatrix} a_1[i] \\ \vdots \\ a_i[i] \end{pmatrix}, \qquad \underline{a}_r[i] = \begin{pmatrix} a_i[i] \\ \vdots \\ a_1[i] \end{pmatrix} = \begin{pmatrix} 0 & \cdots & 1 \\ \vdots & \iddots & \vdots \\ 1 & \cdots & 0 \end{pmatrix}\underline{a}[i]$$

trong đó chỉ số r (reflection) của vector $r_r[i]$ nói rằng thứ tự các phần tử của nó được đảo lại so với $r[i]$, thì từ phương trình (4.17) cho trường hợp $n_a=i+1$ ta có:

$$\underline{r}[i] + H_i\, a[i] = \underline{0}_i \quad \leftarrow \text{Vector gồm có } i \text{ phần tử 0.}$$

Ký hiệu tiếp

$$\underline{\tilde{a}}[i] = \begin{pmatrix} a_1[i] \\ \vdots \\ a_{i-1}[i] \end{pmatrix} \qquad \Rightarrow \qquad \underline{a}[i] = \begin{pmatrix} \underline{\tilde{a}}[i] \\ a_i[i] \end{pmatrix}$$

và

$$\underline{r}[i-1] = \begin{pmatrix} r_y(T_a) \\ \vdots \\ r_y((i-1)T_a) \end{pmatrix} \qquad \Rightarrow \qquad \underline{r}[i] = \begin{pmatrix} \underline{r}[i-1] \\ r_y(iT_a) \end{pmatrix}$$

đẳng thức trên sẽ trở thành

$$\begin{pmatrix} \underline{r}[i-1] \\ r_y(iT_a) \end{pmatrix} + \begin{pmatrix} H_{i-1} & \underline{r}_r[i] \\ \underline{r}_r^T[i] & r_y(0) \end{pmatrix} \begin{pmatrix} \underline{\tilde{a}}[i] \\ a_i[i] \end{pmatrix} = \underline{0}_i$$

$$\Rightarrow \qquad \underline{r}[i-1] + H_{i-1}\underline{\tilde{a}}[i] + \underline{r}_r[i]a_i[i] = \underline{0}_{i-1} \tag{4.23}$$

Mặt khác, khi $n_a = i$, công thức (4.17) có dạng

$$\underline{r}[i-1] + H_{i-1}\underline{a}[i-1] = \underline{0}_{i-1}\,. \tag{4.24}$$

Bởi vậy khi trừ (4.23) và (4.24) theo từng vế ta sẽ được

$$H_{i-1}(\underline{\tilde{a}}[i] - \underline{a}[i-1]) = -\underline{r}_r[i]a_i[i]$$

$$\Rightarrow \qquad \underline{\tilde{a}}[i] - \underline{a}[i-1] = \left(-H_{i-1}^{-1}\underline{r}_r[i]\right)\cdot a_i[i] \tag{4.25}$$

Ngoài ra, do $H_{i\text{-}1}$ là ma trận đối xứng nên có $JH_{i\text{-}1} = H_{i-1}J$, trong đó J là ma trận reflection

$$J = \begin{pmatrix} 0 & \cdots & 1 \\ \vdots & \iddots & \vdots \\ 1 & \cdots & 0 \end{pmatrix}$$

bởi vậy sau khi nhân cả hai vế của (4.24) với J sẽ được

$$J\underline{r}[i-1] + JH_{i-1}\underline{a}[i-1] = \underline{0}_{i-1}$$

$$\Leftrightarrow \qquad \underline{r}_r[i-1] + H_{i-1}J\underline{a}[i-1] = \underline{0}_{i-1}$$

$$\Leftrightarrow \qquad \underline{r}_r[i-1] + H_{i-1}\underline{a}_r[i-1] = \underline{0}_{i-1}$$

nói cách khác đẳng thức (4.24) sẽ không đổi khi thay $\underline{a}[i-1]$ bởi $\underline{a}_r[i-1]$ và thay $\underline{r}[i-1]$ bằng $\underline{r}_r[i-1]$. Suy ra

$$-H_{i-1}^{-1}\underline{r}_r[i] = \underline{a}_r[i-1] \tag{4.26}$$

Thế (4.26) vào (4.25) ta đi đến

$$\underline{\tilde{a}}[i]-\underline{a}[i-1]=\underline{a}_r[i-1]\cdot a_i[i] \tag{4.27}$$

và đó chính là công thức (4.22a) phải chứng minh.

Ta chuyển sang chứng minh (4.22b). Từ (4.27) và (4.19) được

$$\begin{pmatrix} K_i \\ \underline{0}_i \end{pmatrix}=H_i\begin{pmatrix} 1 \\ \underline{a}[i] \end{pmatrix}=H_i\begin{pmatrix} 1 \\ \underline{\tilde{a}}[i] \\ a_i[i] \end{pmatrix}=H_i\begin{pmatrix} 1 \\ \underline{a}[i-1] \\ 0 \end{pmatrix}+H_i\begin{pmatrix} 0 \\ \underline{a}_r[i-1] \\ 1 \end{pmatrix}a_i[i] \tag{4.28}$$

Do số hạng thứ nhất ở vế phải có thể biến đổi thành

$$H_i\begin{pmatrix} 1 \\ \underline{a}[i-1] \\ 0 \end{pmatrix}=\begin{pmatrix} H_{i-1} & \underline{r}_r[i] \\ \underline{r}_r^T[i] & r_y(0) \end{pmatrix}\begin{pmatrix} 1 \\ \underline{a}[i-1] \\ 0 \end{pmatrix}=\begin{pmatrix} H_{i-1}\begin{pmatrix} 1 \\ \underline{a}[i-1] \end{pmatrix} \\ \underline{r}_r^T[i]\begin{pmatrix} 1 \\ \underline{a}[i-1] \end{pmatrix} \end{pmatrix}=\begin{pmatrix} K_{i-1} \\ \underline{0}_{i-1} \\ \underline{r}_r^T[i]\begin{pmatrix} 1 \\ \underline{a}[i-1] \end{pmatrix} \end{pmatrix}$$

nên đẳng thức (4.28) có dạng

$$\begin{pmatrix} K_i \\ \underline{0}_i \end{pmatrix}=\begin{pmatrix} K_{i-1} \\ \underline{0}_{i-1} \\ \underline{r}_r^T[i]\cdot\begin{pmatrix} 1 \\ \underline{a}[i-1] \end{pmatrix} \end{pmatrix}+H_i\begin{pmatrix} 0 \\ \underline{a}_r[i-1] \\ 1 \end{pmatrix}a_i[i] \tag{4.29}$$

Tương tự, ta làm với số hạng thứ hai trong tổng vế phải của (4.29) sẽ có

$$H_i\begin{pmatrix} 0 \\ \underline{a}_r[i-1] \\ 1 \end{pmatrix}a_i[i]=\begin{pmatrix} r_y(0) & \underline{r}^T[i] \\ \underline{r}[i] & H_{i-1} \end{pmatrix}\begin{pmatrix} 0 \\ \underline{a}_r[i-1] \\ 1 \end{pmatrix}a_i[i]=\begin{pmatrix} \underline{r}^T[i]\cdot\begin{pmatrix} \underline{a}_r[i-1] \\ 1 \end{pmatrix} \\ H_{i-1}\cdot\begin{pmatrix} \underline{a}_r[i-1] \\ 1 \end{pmatrix} \end{pmatrix}a_i[i] \tag{4.30}$$

Mặt khác, theo định lý 4.3 thì từ (4.20) ta cũng có điều tương đương

$$H_{i-1}\begin{pmatrix} \underline{a}_r[i-1] \\ 1 \end{pmatrix}=\begin{pmatrix} \underline{0}_{i-1} \\ K_{i-1} \end{pmatrix}$$

Do đó (4.30) trở thành

$$H_i\begin{pmatrix} 0 \\ \underline{a}_r[i-1] \\ 1 \end{pmatrix}a_i[i]=\begin{pmatrix} \underline{r}^T[i]\cdot\begin{pmatrix} \underline{a}_r[i-1] \\ 1 \end{pmatrix} \\ \underline{0}_{i-1} \\ K_{i-1} \end{pmatrix}a_i[i] \tag{4.31}$$

Thay (4.31) vào (4.29) ta đi đến

$$\begin{pmatrix} K_i \\ \underline{0}_{i-1} \\ 0 \end{pmatrix} = \begin{pmatrix} K_{i-1} \\ \underline{0}_{i-1} \\ \underline{r}_r^T[i] \cdot \begin{pmatrix} 1 \\ \underline{a}[i-1] \end{pmatrix} \end{pmatrix} + \begin{pmatrix} \underline{r}^T[i] \cdot \begin{pmatrix} \underline{a}_r[i-1] \\ 1 \end{pmatrix} \\ \underline{0}_{i-1} \\ K_{i-1} \end{pmatrix} a_i[i]$$

và từ đây suy ra

$$K_i = K_{i-1} + \underline{r}^T[i] \cdot \begin{pmatrix} \underline{a}_r[i-1] \\ 1 \end{pmatrix} a_i[i] \tag{4.32}$$

và

$$0 = \underline{r}_r^T[i] \cdot \begin{pmatrix} 1 \\ \underline{a}[i-1] \end{pmatrix} + K_{i-1} a_i[i] \;\Rightarrow\; \underline{r}_r^T[i] \cdot \begin{pmatrix} 1 \\ \underline{a}[i-1] \end{pmatrix} = -K_{i-1} a_i[i] \tag{4.33}$$

Thay (4.33) vào (4.32) ta có được điều phải chứng minh thứ hai (4.22b):

$$K_i = K_{i-1} - K_{i-1}(a_i[i])^2 \qquad \square$$

Như vậy, định lý 4.4 với hai công thức (4.22a), (4.22b) đã cung cấp cho ta khả năng xác định truy hồi các tham số $a_1[i]$, $a_2[i]$, ... , $a_{i-1}[i]$ và K_i của (4.19) từ các tham số $a_1[i-1]$, $a_2[i-1]$, ... , $a_{i-1}[i-1]$, K_{i-1} của (4.20) với giả thiết rằng đã biết $a_i[i]$ của (4.19). Nói cách khác vấn đề còn lại phải giải quyết là đi tìm $a_i[i]$.

Để tìm $a_i[i]$ từ các tham số $a_1[i-1]$, $a_2[i-1]$, ... , $a_{i-1}[i-1]$, K_{i-1} của (4.20), trước hết ta viết lại (4.22a) như sau:

$$\begin{pmatrix} 1 \\ a_1[i] \\ \vdots \\ a_{i-1}[i] \\ a_i[i] \end{pmatrix} = \begin{pmatrix} 1 \\ a_1[i-1] \\ \vdots \\ a_{i-1}[i-1] \\ 0 \end{pmatrix} + a_i[i] \begin{pmatrix} 0 \\ a_{i-1}[i-1] \\ \vdots \\ a_1[i-1] \\ 1 \end{pmatrix}$$

sau đó thay vào (4.19) sẽ được

$$\begin{pmatrix} K_i \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} = H_i \begin{pmatrix} 1 \\ a_1[i] \\ \vdots \\ a_{i-1}[i] \\ a_i[i] \end{pmatrix} = H_i \begin{pmatrix} 1 \\ a_1[i-1] \\ \vdots \\ a_{i-1}[i-1] \\ 0 \end{pmatrix} + a_i[i] H_i \begin{pmatrix} 0 \\ a_{i-1}[i-1] \\ \vdots \\ a_1[i-1] \\ 1 \end{pmatrix}. \tag{4.34}$$

Nhưng vì

$$H_i = \begin{pmatrix} H_{i-1} & \underline{r}_r[i] \\ \underline{r}_r^T[i] & r_y(0) \end{pmatrix} = \begin{pmatrix} r_y(0) & \underline{r}^T[i] \\ \underline{r}[i] & H_{i-1} \end{pmatrix}$$

trong đó

$$r[i] = \begin{pmatrix} r_y(T_a) \\ \vdots \\ r_y(iT_a) \end{pmatrix}, \qquad r_r[i] = \begin{pmatrix} r_y(iT_a) \\ \vdots \\ r_y(T_a) \end{pmatrix}$$

nên (4.34) có thể viết được thành

$$\begin{pmatrix} K_i \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} H_{i-1} & \underline{r}_r[i] \\ \underline{r}_r^T[i] & r_y(0) \end{pmatrix} \begin{pmatrix} 1 \\ a_1[i-1] \\ \vdots \\ a_{i-1}[i-1] \\ 0 \end{pmatrix} + a_i[i] \begin{pmatrix} r_y(0) & \underline{r}^T[i] \\ \underline{r}[i] & H_{i-1} \end{pmatrix} \begin{pmatrix} 0 \\ a_{i-1}[i-1] \\ \vdots \\ a_1[i-1] \\ 1 \end{pmatrix}$$

$$\Leftrightarrow \quad \begin{pmatrix} K_i \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} H_{i-1} \begin{pmatrix} 1 \\ a_1[i-1] \\ \vdots \\ a_{i-1}[i-1] \end{pmatrix} \\ r_r^T[i] \begin{pmatrix} 1 \\ a_1[i-1] \\ \vdots \\ a_{i-1}[i-1] \end{pmatrix} \end{pmatrix} + a_i[i] \begin{pmatrix} r^T[i] \begin{pmatrix} a_{i-1}[i-1] \\ \vdots \\ a_i[i-1] \\ 1 \end{pmatrix} \\ H_{i-1} \begin{pmatrix} a_{i-1}[i-1] \\ \vdots \\ a_i[i-1] \\ 1 \end{pmatrix} \end{pmatrix} \tag{4.35}$$

Mặt khác từ (4.20) có

$$H_{i-1} \begin{pmatrix} 1 \\ a_1[i-1] \\ \vdots \\ a_{i-1}[i-1] \end{pmatrix} = \begin{pmatrix} K_{i-1} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

và với định lý 4.3 cho trường hợp i–1 thay vào vị trí của i, đẳng thức (4.35) được biến đổi thành

$$\begin{pmatrix} K_i \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} K_{i-1} \\ 0 \\ \vdots \\ 0 \\ r_y(iT_a) + \sum_{k=1}^{i-1} r_y[(i-k)T_a] a_k[i-1] \end{pmatrix} + a_i[i] \begin{pmatrix} r^T[i] \begin{pmatrix} a_{i-1}[i-1] \\ \vdots \\ a_i[i-1] \\ 1 \end{pmatrix} \\ 0 \\ \vdots \\ 0 \\ K_{i-1} \end{pmatrix}$$

So sánh riêng phần tử cuối cùng của hai vế ta được

$$0 = r_y(iT_a) + \sum_{k=1}^{i-1} r_y[(i-k)T_a]a_k[i-1] + a_i[i]K_{i-1}$$

$$\Leftrightarrow \qquad a_i[i] = -\frac{r_y(iT_a) + \sum_{k=1}^{i-1} r_y[(i-k)T_a]a_k[i-1]}{K_{i-1}} \tag{4.36}$$

và đó chính là công thức cho phép xác định $a_i[i]$ từ các tham số $a_1[i–1]$, $a_2[i–1]$, ... , $a_{i-1}[i–1]$, K_{i-1} đã biết.

Trong trường hợp i=0 thì từ (4.19) có ngay được $K_0 = r_y(0)$.

Ta đi đến thuật toán Levinson để giải phương trình Yule–Walker (4.14) như sau:

1) Gán $K_0 = r_y(0)$.

2) Thực hiện các bước sau cho i=1, 2, ... , n_a .

   a) Tính $a_i[i]$ theo (4.36).

   b) Tính $a_1[i]$, $a_2[i]$, ... , $a_{i-1}[i]$ và K_i theo (4.22a) và (4.22b).

Thuật toán trên đã được cài đặt thành hàm **`Levinson()`** viết trên ngôn ngữ C cho dưới đây để tham khảo. Hàm có hai biến hình thức là

– con trỏ **`r`** chỉ đầu mảng **`r[]`** chứa dãy các giá trị hàm tương quan theo thứ tự

  `r[0]`$= r_y(0)$, **`r[1]`**$= r_y(T_a)$, … , **`r[i]`**$= r_y(iT_a)$, …

– biến nguyên **`na`** chứa kích thước của mảng, tức là n_a+1 (bậc của phương trình).

Các giá trị $a_1[n_a]$, $a_2[n_a]$, ... , $a_{n_a}[n_a]$ và K_{n_a} tính được sẽ được hàm đưa ra ngoài thông qua mảng **`a[]`** theo thứ tự

  `a[0]`$= K_{n_a}$, **`a[1]`**$= a_1[n_a]$, … , **`a[na]`**$= a_{n_a}[n_a]$.

Như vậy mảng **`a[]`** ít nhất phải có n_a+1 phần tử. Hàm trả về giá trị nguyên 0 thông báo phương trình (4.14) giải được, tức là ma trận H_{n_a} không suy biến, hoặc 1 khi (4.14) không giải được.

```
int levinson(double *r,int na,double *a)
{
   int i,k,j,ik,p=0;
   double sum,save;
   a[0]=r[0];
   for (i=1;i<=na;i++)
```

```
    {
        sum=0;
        for (k=1;k<i;k++)    sum=sum+r[i-k]*a[k];
        if (a[0]!=0.) a[i]=-(r[i]+sum)/a[0];
           else {p=1; break;}
        j=i/2;
        for (k=1;k<=j;k++)
        {
           ik=i-k; save=a[k];
           a[k]=save+a[i]*a[ik];
           if (k!=ik) a[ik]=a[ik]+a[i]*save;
        }
        a[0]=a[0]*(1.-a[i]*a[i]);
    }
    return(p);
}
```

Ví dụ: Gọi hàm trên bằng lệnh

```
k=levinson(r,na,a);
```

với các giá trị đầu vào **na**=2, **r[0]**=2, **r[1]**=1, **r[2]**=3, ta sẽ nhận được

k= 0, **a[0]**= –2,6667, **a[1]**= 0,3333 và **a[2]**= –1,6667.

Thử lại ta thấy đó là kết quả đúng:

$$\begin{pmatrix} 2 & 1 & 3 \\ 1 & 2 & 1 \\ 3 & 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0{,}3333 \\ -1{,}6667 \end{pmatrix} = \begin{pmatrix} -2{,}6667 \\ 0 \\ 0 \end{pmatrix}.$$ ❑

Dựa vào chương trình **Levinson()** thuật toán xác định các tham số $\underline{a}$, K theo phương pháp Yule–Walker có dạng:

1) Xác định giá trị hàm tự tương quan $r_y(mT_a)$, m=0,1, … , n_a bằng hàm **cor()** đã được trình bày trong chương 2, từ dãy giá trị $\{y_k\}$ đo được của tín hiệu ra.

2) Gọi hàm **Levinson()** để tính $\underline{a}$, K từ $r_y(mT_a)$, m=0,1, … , n_a.

Thuật toán trên đã được cài đặt thành hàm **Yule_Walker()** viết trên C cho sau đây để tham khảo. Hàm **Yule_Walker()** có 5 biến hình thức, bao gồm:

a) Con trỏ **y** chỉ đầu mảng **y[]** chứa các giá trị y_k, k=0,1, … , N–1.

b) Số nguyên **N** chứa độ dài dãy $\{y_k\}$, tức là chứa chỉ số N.

c) Số nguyên **na** chứa bậc mô hình AR.

d) Số nguyên **bias** xác định giá trị hàm tương quan sẽ được tính theo công thức bias (**bias**=1) hay unbias (**bias**≠1).

e) Con trỏ **a** chỉ đầu mảng **a[]** chứa kết quả theo thứ tự:

a[0]= K_{n_a} , **a[1]**= $a_1[n_a]$, … , **a[na]**= $a_{n_a}[n_a]$.

Hàm trả về giá trị báo lỗi bằng 0 nếu $M<N$ hoặc ma trận H_{n_a} không suy biến. Hàm sẽ trả về giá trị 1 khi có lỗi. Hàm không làm thay đổi nội dung của **y[]**.

```
int Yule_Walker(double *y,int N,int na,int bias,double *a)
{
    int k;
    double *r;
    r=new double[na+1];
    if((k=cor(y,y,N,na,bias,r))==0)
       k=levinson(r,na,a);
    delete [] r;
    return(k);
}
```

4.2.4 Phương pháp dự báo điều hòa và thuật toán Burg

Thuật toán giải trực tiếp phương trình Yule–Walker (4.14) dựa vào chương trình con **Levinson()** đã trình bày trong mục 4.2.3 cần phải có một bước trung gian là nhận dạng các giá trị hàm tương quan $r_y(mT_a)$, m=0,1, … , n_a để có được ma trận H_{n_a} và điều này làm cho trong kết quả thu được có lẫn thêm sai số tính toán không cần thiết.

Nhằm tránh các sai số thừa đó, Burg đã dựa vào kết quả của hai định lý 4.1 và 4.2 về sai số dự báo tuyến tính xây dựng lên một thuật toán truy hồi cho phép xác định những tham số a_1 , a_2 , … , a_{n_a} và K của mô hình AR trực tiếp từ các giá trị tín hiệu đo được mà không cần thông qua hàm tương quan. Tương tự như thuật toán Levinson, thuật toán Burg cũng tính truy hồi các tham số $a_1[i]$, $a_2[i]$, … , $a_{i-1}[i]$ và K_i của (4.19) từ các tham số $a_1[i-1]$, $a_2[i-1]$, … , $a_{i-1}[i-1]$, K_{i-1} của (4.20) theo các công thức (4.22a) và (4.22b) vì hai công thức truy hồi này không sử dụng giá trị hàm tương quan $r_y(kT_a)$. Điểm khác cơ bản so với Levinson là Burg không tính $a_i[i]$ theo công thức (4.36) mà dựa theo tính chất về sai số dự báo tuyến tính của phương pháp Yule–Walker.

Để tìm $a_i[i]$ Burg đã dựa vào hai định lý 4.1, 4.2 phát biểu rằng với bộ tham số $a_1[i]$, $a_2[i]$, … , $a_i[i]$ và K_i và xác định được theo thuật toán Yule–Walker, giá trị trung bình của bình phương sai lệch dự báo tuyến tính

$$e_n^f[i] = y_n - y_n^f[i] \tag{4.37a}$$

$$e_n^b[i] = y_{n-i} - y_{n-i}^b[i] \tag{4.37b}$$

là nhỏ nhất. Ký kiệu i trong ngoặc vuông của $e_n^f[i]$, $e_n^b[i]$ chỉ rằng sai lệch đó là ứng với mô hình bậc i. Những giá trị ngoại suy tiến $y_n^f[i]$ và lùi $y_n^f[i]$ tính theo công thức (4.15), (4.16) cho mô hình bậc i có dạng như sau:

$$y_n^f[i] = -\sum_{k=1}^{i} a_k[i] y_{n-k} \tag{4.38}$$

$$y_n^b[i] = -\sum_{k=1}^{i} a_k[i] y_{n+k} \tag{4.39}$$

Nếu giá trị trung bình của bình phương sai lệch $M\left[\left|e_n^f[i]\right|^2\right]$ và $M\left[\left|e_n^b[i]\right|^2\right]$ là nhỏ nhất thì đương nhiên tổng

$$Q = \sum_{n=i}^{N-1} \left(e_n^f[i]^2 + e_n^b[i]^2\right) \tag{4.40}$$

cũng nhỏ nhất. Nhưng trước khi đi tìm điều kiện cho $a_i[i]$ để Q nhận giá trị cực tiểu, ta sẽ tìm mối quan hệ truy hồi giữa $e_n^f[i], e_n^b[i]$ và $e_n^f[i-1], e_n^b[i-1]$.

Định lý 4.5: Giữa $e_n^f[i], e_n^b[i]$ và $e_n^f[i-1], e_n^b[i-1]$ có quan hệ truy hồi

a) $e_n^f[i] = e_n^f[i-1] + a_i[i] e_{n-1}^b[i-1]$ (4.41)

b) $e_n^b[i] = e_{n-1}^b[i-1] + a_i[i] e_n^f[i-1]$ (4.42)

Chứng minh:

Từ định nghĩa về sai số dự báo tuyến tính lùi (4.37b) và giá trị được dự báo (4.39) có

$$\begin{aligned} e_{n-1}^b[i-1] &= y_{n-i} - y_{n-i}^b[i-1] = y_{n-i} + \sum_{k=1}^{i-1} a_k[i-1] y_{n-i+k} \\ &= y_{n-i} + \sum_{k=1}^{i-1} a_{i-k}[i-1] y_{n-k} \quad (\text{thế } k \text{ bởi } i\text{–}k) \end{aligned} \tag{4.43}$$

Mặt khác, với (4.37a) và (4.38) cho công thức dự báo tiến thì

$$\begin{aligned} e_n^f[i] - e_n^f[i-1] &= \sum_{k=1}^{i} a_k[i] y_{n-k} - \sum_{k=1}^{i-1} a_k[i-1] y_{n-k} \\ &= a_i[i] y_{n-i} + \sum_{k=1}^{i-1} \left(a_k[i] - a_k[i-1]\right) \cdot y_{n-k} \end{aligned} \tag{4.44}$$

Thay hiệu $a_k[i] - a_k[i-1]$ từ (4.22a) vào (4.44) được

$$e_n^f[i] - e_n^f[i-1] = a_i[i]\left(y_{n-i} + \sum_{k=1}^{i-1} a_{i-k}[i-1] \cdot y_{n-k} \right) \tag{4.45}$$

So sánh (4.43) với (4.45) ta có điều phải chứng minh (4.41)

$$e_n^f[i] - e_n^f[i-1] = a_i[i] \cdot e_{n-1}^b[i-1].$$

Công thức (4.42) được chứng minh một cách hoàn toàn tương tự. ❑

Quay lại công việc chính là xác định $a_i[i]$ để Q có giá trị nhỏ nhất. Với định lý 4.5, phiếm hàm Q định nghĩa theo công thức (4.40) trở thành

$$Q = \sum_{n=i}^{N-1} \left[\left(1 + a_i[i]^2\right)\left(e_n^f[i-1]^2 + e_{n-1}^b[i]^2\right) + 4a_i[i]e_n^f[i-1]e_{n-1}^b[i-1] \right],$$

hơn nữa, do Q là hàm xác định dương nên để Q cực tiểu thì cần và đủ là

$$0 = \frac{dQ}{da_i[i]} = \sum_{n=i}^{N-1} \left[2a_i[i]\left(e_n^f[i-1]^2 + e_{n-1}^b[i]^2\right) + 4e_n^f[i-1]e_{n-1}^b[i-1] \right]$$

$$\Rightarrow \qquad a_i[i] = \frac{-2\sum_{n=i}^{N-1}\left(e_n^f[i-1]e_{n-1}^b[i-1]\right)}{\sum_{n=i}^{N-1}\left(e_n^f[i-1]^2 + e_{n-1}^b[i-1]^2\right)} \tag{4.46}$$

Như vậy với (4.22a), (4.22b), (4.41), (4.42) và (4.46) ta đã có đầy đủ các công thức truy hồi để xác định $a_1[i]$, $a_2[i]$, ... , $a_i[i]$ và K_i ứng với mô hình bậc i từ các tham số $a_1[i-1]$, $a_2[i-1]$, ... , $a_{i-1}[i-1]$, K_{i-1} của mô hình bậc $i-1$. Song để xây dựng thành thuật toán hoàn chỉnh còn cần phải có những giá trị khởi phát $e_n^f[0]$, $e_n^b[0]$ và K_0.

Khi $i=0$ thì với (4.38), (4.39) có $y_n^f = y_n^b = 0$ và $y_n = K_0 u_n$, do đó

$$e_n^f[0] = e_n^b[0] = y_n \ , \quad n = 0, 1, \ldots, N-1 \tag{4.47a}$$

và

$$K_0 = r_y(0) = \frac{1}{N}\sum_{k=0}^{N-1} y_k^2 \ . \tag{4.47b}$$

Cuối cùng, ta đi đến thuật toán truy hồi như sau:

1) Xác định $e_n^f[0]$, $e_n^b[0]$ và K_0 từ $\{y_n\}$, $n = 0, 1, \ldots, N-1$ theo (4.47).

2) Thực hiện các bước sau lần lượt với $i = 1, 2, \ldots, n_a$:

a) Tính $a_i[i]$ từ $e_n^f[i-1], e_{n-1}^b[i-1]$ theo (4.46).

b) Tính $a_k[i]$ và K_i , k=0,1, … , i theo (4.22a), (4.22b).

c) Tính $e_n^f[i], e_n^b[i]$, n=0,1, … , N–1 theo (4.41), (4.42).

Thuật toán trên đã được cài đặt thành hàm **Burg()** viết trên ngôn ngữ lập trình C cho dưới đây để tham khảo. Hàm này có các biến hình thức sau:

a) Con trỏ **y** chỉ đầu mảng **y[]** chứa các giá trị y_k , k=0,1, … , N–1.

b) Số nguyên **N** chứa độ dài dãy $\{y_k\}$, tức là chứa chỉ số N.

c) Số nguyên **na** chứa bậc mô hình AR.

d) Con trỏ **a** chỉ đầu mảng **a[]** chứa kết quả theo thứ tự:

a[0]= K_{n_a} , **a[1]**= $a_1[n_a]$, … , **a[na]**= $a_{n_a}[n_a]$.

Hàm trả về giá trị báo lỗi bằng 1 nếu không tính được $a_i[i]$ theo (4.46) vì mẫu số bằng 0 hoặc trả về giá trị 0 khi không có lỗi.

Hàm **Burg()** không làm thay đổi nội dung của mảng **y[]**.

```
int Burg(double *y,int N,int na,double *a)
{  int i,k,j,ik;
   double *ef,*eb,sum1=0.,sum2;
   ef=new double[N];
   eb=new double[N];
   for(i=0;i<N;i++)
   {  ef[i]=eb[i]=y[i];
      sum1=sum1+y[i]*y[i];
   }
   a[0]=sum1/(double)N;
   for (i=1;i<=na;i++)
   {  sum1=sum2=0.;
      for (k=i;k<N;k++)
      {  j=k-1;
         sum1=sum1+ef[k]*eb[j];
         sum2=sum2+ef[k]*ef[k]+eb[j]*eb[j];
      }
      if(sum2!=0.) a[i]=-2.*sum1/sum2; else return(1);
      j=i/2;
      for (k=1;k<=j;k++)
      {  ik=i-k;
         sum1=a[k];
         a[k]=sum1+a[i]*a[ik];
         if (k!=ik) a[ik]=a[ik]+a[i]*sum1;
      }
```

```
        a[0]=a[0]*(1.-a[i]*a[i]);
        for (k=N-1;(k>i)&&(i!=na);k--)
        {   j=k-1;
            sum1=ef[k];
            ef[k]=sum1+a[i]*eb[j];
            eb[k]=eb[j]+a[i]*sum1;
        }
    }
    delete [] ef; delete [] eb;
    return(0);
}
```

4.2.5 Kết luận

Mục 4.2.2, cụ thể là hai định lý 4.1 và 4.2, đã chỉ rằng bộ tham số a_1, a_2, … , a_{n_a} của mô hình AR làm cho phiếm hàm mô tả trung bình bình phương sai lệch dự báo tuyến tính:

$$Q_f = M\left[|e_n^f|^2\right] = M\left[\left|y_n - y_n^f\right|^2\right] \to \min!,$$

và $$Q_b = M\left[|e_n^b|^2\right] = M\left[\left|y_n - y_n^b\right|^2\right] \to \min!,$$

đạt giá trị nhỏ nhất chính là nghiệm của phương trình Yule–Walker (4.14).

Tuy nhiên điều khẳng định đó chỉ đúng nếu như tất cả các giá trị hàm tương quan $r_y(mT_a)$, m=0,1, … , n_a thu được là phải hoàn toàn chính xác, song điều này là không hiện thực vì bản thân chúng cũng chỉ được nhận dạng (xấp xỉ) từ dãy các giá trị $\{y_k\}$, k=0,1, … , N–1 của tín hiệu $y(t)$ đo được trong khoảng thời gian quan sát $[0,T)$ tương đối ngắn. Chính vì điều này mà Burg đã đưa ra giải pháp nhận dạng a_1, a_2, … , a_{n_a} không thông qua bước trung gian xác định hàm tương quan bằng cách cực tiểu hóa phiếm hàm sai lệch điều hòa:

$$Q_{Burg} = \sum_{n=n_a}^{N-1}\left(\left[e_n^f\right]^2 + \left[e_n^b\right]^2\right) \to \min!.$$

được xem như một giải pháp dung hòa của hai điều kiện cực tiểu.

Ngoài hai thuật toán trên còn có một số thuật toán khác cũng được sử dụng khá phổ biến để nhận dạng mô hình AR như thuật toán Morf, thuật toán Marple …. Những thuật toán này rất thích hợp với các bài toán có khoảng quan sát lớn (số lượng lớn các giá trị y_k). Độc giả quan tâm có thể tham khảo chúng trong tài liệu [10], [11].

4.3 Nhận dạng chủ động tham số mô hình MA

4.3.1 Thay mô hình MA bằng mô hình AR tương đương

Bài toán nhận dạng chủ động tham số mô hình MA phát biểu như sau: khi đối tượng mô tả bởi

$$\widetilde{G}(z) = 1 + b_1 z^{-1} + \ldots + b_{n_b} z^{-n_b} \tag{4.48}$$

được kích thích chủ động bằng tín hiệu ồn trắng $u(t)$ thỏa mãn

$$m_u = 0 \text{ và } S_u(\omega) = K, \tag{4.49}$$

hãy xác định các vector tham số $\underline{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_{n_b} \end{pmatrix}$ của mô hình (4.48) và K của tín hiệu vào $u(t)$ từ dãy N giá trị đo được $\{y_k\}$, $k = 0,1, \ldots, N-1$ của tín hiệu ra, sao cho sai lệch giữa mô hình và đối tượng là nhỏ nhất.

Phải nói rằng việc xây dựng một công thức tương tự như phương trình Yule–Walker để tìm $\underline{b}$, K là có thể, song khả năng ứng dụng của nó rất thấp do quan hệ giữa vector tham số $\underline{b}$ và sai lệch đối tượng/mô hình có tính phi tuyến mạnh, sinh ra bởi đặc thù mô hình MA. Chính vì vậy ta sẽ không đi tiếp vào hướng giải quyết đó mà tìm cách sử dụng lại những thuật toán đã biết của nhận dạng chủ động mô hình AR phục vụ việc nhận dạng tham số mô hình MA. Ý tưởng này có được trên cơ sở nguyên lý đối ngẫu Markov phát biểu như sau:

Định lý 4.6: Với mỗi một mô hình MA (4.48) bao giờ cũng tồn tại một mô hình AR tương đương bậc vô hạn. Nói cách khác bao giờ cũng tồn tại dãy vô hạn các tham số c_1, c_2, … sao cho

$$1 + \sum_{n=1}^{n_b} b_n z^{-n} = \frac{1}{1 + \sum_{n=1}^{\infty} c_n z^{-n}} \tag{4.50}$$

Giữa các tham số b_1, b_2, … , b_{n_b} và c_1, c_2, … có quan hệ truy hồi

$$c_{n+n_b} = \sum_{i=0}^{n_b-1} b_{i+1} c_{n+i} \text{ , trong đó } n \geq 1 \tag{4.51}$$

Ta sẽ bỏ qua việc chứng minh định lý trên và công nhận chúng như điều hiển nhiên. Những bạn đọc quan tâm có thể tìm thấy phần chứng minh trong các tài liệu [10] hoặc [12].

Dựa theo định lý 4.6 thì việc nhận dạng tham số mô hình MA sẽ được thay thế bằng các thuật toán nhận dạng tham số mô hình AR đã biết, tức là nhận dạng các tham số c_1,

c_2 , … . Sau đó tính ngược b_1 , b_2 , … , b_{n_b} từ c_1 , c_2 , … theo (4.51). Phương hướng giải quyết rất rõ ràng như vậy, song để thực hiện nó ta sẽ gặp các khó khăn gì?.

Trước mắt có thể thấy ngay được hai khó khăn cơ bản sau:

- Mô hình AR tương đương có bậc vô hạn, tức là phải xác định vô hạn các tham số c_1 , c_2 , ….
- Việc tính ngược b_1 , b_2 , … , b_{n_b} từ c_1 , c_2 , … theo (4.51) đồng nghĩa với việc tìm nghiệm của hệ phương trình có số các phương trình (n=1,2, …) nhiều hơn số ẩn số b_1 , b_2 , … , b_{n_b} nên ta thường gặp phải trường hợp hệ vô nghiệm.

Để loại bỏ khó khăn thứ nhất, ta sẽ thay mô hình AR bậc vô hạn bằng một mô hình AR bậc hữu hạn s và cùng với nó (4.50) được sửa đổi thành

$$1 + \sum_{n=1}^{n_b} b_n z^{-n} \approx \frac{1}{1 + \sum_{n=1}^{s} c_n z^{-n}} \tag{4.52}$$

và do đó phải chấp nhận một sai số sinh ra bởi sự thay thế này. Cùng với sự thay thế đó, dãy vô hạn c_1, c_2 , … trở thành dãy hữu hạn c_1, c_2 , … , c_s . Tất nhiên bậc s càng lớn, sai số đó sẽ càng nhỏ.

Để tránh được khó khăn thứ hai ta có hai cách:

1) Chọn s= $2n_b$. Khi đó với n=1,2, … , n_b sẽ có được từ (4.51) đúng n_b phương trình cho n_b ẩn b_1 , b_2 , … , b_{n_b} .

2) Chọn s> $2n_b$. Vậy thì không thể sử dụng (4.51) để xác định b_1 , b_2 , … , b_{n_b} từ c_1, c_2 , …, c_s . Ta sẽ tính trực tiếp b_1 , b_2 , … , b_{n_b} nhờ (4.52), trong đó tham số c_k được xem như bằng 0 khi k>s .

4.3.2 Thuật toán nhận dạng cho trường hợp s=$2n_b$

Sau khi thay mô hình MA (4.48) bởi một mô hình AR tương đương theo công thức (4.52) và đã xác định được các tham số c_1, c_2 , … , c_s của mô hình AR tương đương đó cũng như giá trị mật độ phổ K của $u(t)$ thì việc cuối cùng phải làm là tính bộ tham số b_1 , b_2 , … , b_{n_b} của mô hình MA từ các giá trị c_1, c_2 , … , c_s đã tìm được.

Với giả thiết s= $2n_b$ ta có thể áp dụng trực tiếp công thức Markov (4.51) để tìm b_1 , b_2 , … , b_{n_b} từ c_1, c_2 , … , c_s vì khi đó số các phương trình sẽ đúng bằng số các ẩn số. Viết lại (4.51) lần lượt cho n=1, 2, … , $2n_b$ được:

$$\begin{pmatrix} c_1 & c_2 & \cdots & c_{n_b} \\ c_2 & c_3 & \cdots & c_{n_b+1} \\ \vdots & & & \\ c_{n_b+1} & c_{n_b+2} & \cdots & c_{2n_b} \end{pmatrix} \cdot \begin{pmatrix} b_1 \\ \vdots \\ b_{n_b} \end{pmatrix} = \begin{pmatrix} c_{n_b+1} \\ \vdots \\ c_{2n_b} \end{pmatrix} \tag{4.53}$$

và đó chính là hệ phương trình tuyến tính cho phép xác định b_1, b_2, … , b_{n_b} từ c_1, c_2, … , c_{2n_b}.

Ta đi đến thuật toán thứ nhất:

1) Sử dụng các thuật toán nhận dạng tham số mô hình AR đã biết như thuật toán Levinson (mục 4.2.3) hay thuật toán truy hồi của Burg (mục 4.2.4) để xác định c_1, c_2, … , c_{2n_b} và K từ dãy N giá trị đo được $\{y_k\}$, k=1, … , N–1 của tín hiệu đầu ra.

2) Giải phương trình tuyến tính (4.53) để có b_1, b_2, … , b_{n_b} từ c_1, c_2, … , c_{2n_b}.

Chú ý rằng trong thuật toán vừa trình bày ta phải chấp nhận sai số do giả thiết bậc mô hình AR tương đương là hữu hạn và bản thân thuật toán không làm giảm được sai số này. Riêng trường hợp khi bậc của mô hình MA là n_b lớn thì bậc của AR tương đương là $2n_b$ cũng sẽ lớn, nên sai số đó sẽ giảm.

4.3.3 Thuật toán nhận dạng cho trường hợp $s>2n_b$

Khi $s> 2n_b$ ta không thể sử dụng công thức Markov (4.51) để xác định trực tiếp b_1, b_2, … , b_{n_b} từ c_1, c_2, … , c_s. Thay vào đó ta sử dụng (4.52). Nếu ký hiệu

$$B(z) = 1 + \sum_{n=1}^{n_b} b_n z^{-n} \qquad \text{và} \qquad C(z) = 1 + \sum_{n=1}^{s} c_n z^{-n}$$

ta sẽ thấy $B(z)$ chính là ảnh toán tử $\mathcal{Z}$ của dãy giá trị $\{1, b_1, b_2, \dots, b_{n_b}\}$ và $C(z)$ là ảnh $\mathcal{Z}$ của $\{1, c_1, c_2, \dots, c_s\}$. Do theo (4.52), tích của hai ảnh bằng 1

$$B(z)\cdot C(z) \approx 1$$

nên tích chập của hai dãy giá trị trên phải là hàm dirac

$$b_n * c_n \approx \begin{cases} 1 & \text{nếu} \quad n = 0 \\ 0 & \text{nếu} \quad n \neq 0 \end{cases}$$

Suy ra

$$c_n + \sum_{k=1}^{n_b} b_k c_{n-k} \approx 0 \quad \text{với} \quad n \geq 1. \tag{4.54}$$

Đẳng thức (4.54) chỉ thực sự đúng khi mô hình AR tương đương có bậc vô cùng và các tham số c_1 , c_2 … là chính xác. Nhưng ở đây mô hình AR lại có bậc hữu hạn s và các tham số c_1, c_2 , … , c_s của nó cũng chỉ được nhận dạng bằng thuật toán Yule–Walker hoặc Burg, tức là các giá trị mà ta có chỉ là những giá trị xấp xỉ. Nói cách khác

$$c_n + \sum_{k=1}^{n_b} b_k c_{n-k} = e_n \neq 0 \qquad \text{với} \qquad n \geq 1$$

trong đó e_n là sai số.

Bởi vậy một thuật toán nào đó xác định b_1 , b_2 , … , b_{n_b} từ c_1, c_2 , … , c_s sẽ được gọi là tốt nhất nếu nó mang lại giá trị trung bình của bình phương sai số $M\{e_n^2\}$ nhỏ nhất. Nếu như rằng trong bài toán vừa nêu ta xem 1, c_1, … , c_s như là dãy $\{y_k\}$ thì nó sẽ chính là bài toán dự báo tuyến tính tiến mà ta đã đề cập tại mục 4.2.2. Do đó, theo định lý 4.1, nghiệm b_1 , b_2 , … , b_{n_b} phải thỏa mãn phương trình Yule–Walker (4.14) với vai trò dãy $\{y_k\}$ được thay bằng $\{1, c_1, c_2 , \dots , c_s\}$.

Ta đi đến thuật toán thứ 2 để nhận dạng tham số mô hình MA như sau:

1) Chọn $s > 2n_b$. Cũng có thể chọn $s \geq 2n_b$.
2) Sử dụng các chương trình nhận dạng chủ động tham số mô hình AR đã biết như **`Yule_Walker()`** hay **`Burg()`** để xác định c_1, c_2 , … , c_s và K từ dãy N giá trị đo được $\{y_k\}$, k=1, … , N–1 của tín hiệu đầu ra.
3) Sử dụng các chương trình nhận dạng tham số mô hình AR **`Yule_Walker()`** hoặc **`Burg()`** một lần nữa để xác định b_1 , b_2 , … , b_{n_b} từ dãy giá trị 1, c_1, c_2 , … , c_s .

Dưới đây là hàm **`MA()`** viết trên ngôn ngữ lập trình C mô tả việc cài đặt thuật toán vừa trình bày để tham khảo. Hàm này sử dụng hàm con **`Yule_Walker()`** cho cả hai bước của thuật toán, tức là vừa để nhận dạng tham số c_1, c_2 , … , c_s của mô hình AR cũng như để xác định b_1 , b_2 , … , b_{n_b} từ dãy giá trị $\{1, c_1, c_2 , \dots , c_s\}$.

Hàm **`MA()`** có các biến hình thức sau:

a) Con trỏ **`y`** chỉ đầu mảng **`y[]`** chứa các giá trị y_k , k=0,1, … , N–1.
b) Số nguyên **`N`** chứa độ dài dãy $\{y_k\}$, tức là chứa chỉ số N.
c) Số nguyên **`nb`** chứa bậc mô hình MA.

d) Số nguyên **s** chứa bậc mô hình AR được thay thế tạm thời cho mô hình MA.

e) Số nguyên **bias** xác định giá trị hàm tương quan sẽ được tính theo công thức bias (**bias**=1) hay unbias (**bias**≠1).

f) Con trỏ **b** chỉ đầu mảng **b[]** chứa kết quả theo thứ tự:

b[0]=K , **b[1]**= b_1 , … , **b[nb]**= b_{n_b} .

Hàm trả về giá trị báo lỗi bằng 1 nếu **Yule_Walker()** có lỗi hoặc 2 khi giá trị đầu vào s của **s** và n_b của **nb** không thỏa mãn điều kiện $s \geq 2n_b$. Trường hợp không có lỗi, hàm sẽ trả về giá trị 0 và kết quả được lưu giữ trong mảng **b[]**. Hàm **MA()** không làm thay đổi nội dung của mảng **y[]**.

```
int MA(double *y,int N,int nb,int s,int bias,double *b)
{  int i;
   if (s<2*nb) return(2);
   double K,*c;
   c=new double[s+1];
   i=Yule_Walker(y,N,s,bias,c);
   if(i==0)
   {  K=c[0];
      c[0]=1.;
      i=Yule_Walker(c,s+1,nb,bias,b);
      b[0]=K;
   }
   delete [] c;
   return(i);
}
```

Chú ý: Trong quá trình xây dựng thuật toán thứ hai ta đã không sử dụng giả thiết $s > 2n_b$ nên chương trình **MA()** hoàn toàn vẫn sử dụng được cho cả trường hợp $s = 2n_b$. Thực tế ứng dụng đã chỉ rằng ngay cả khi $s = 2n_b$ thì **MA()** cũng cho ra kết quả tốt hơn là áp dụng trực tiếp thuật toán thứ nhất để giải hệ phương trình (4.53).

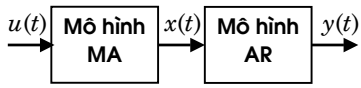
4.4 Nhận dạng chủ động tham số mô hình ARMA

Việc nhận dạng trực tiếp tham số mô hình ARMA (4.10), trong đó tín hiệu đầu vào $u(t)$ được giả thiết là ngẫu nhiên egodic với

$$m_u = 0 \quad \text{và} \quad S_u(\omega) = K$$

có thể nói là không khả thi vì quan hệ giữa hàm sai lệch Q của mô hình–đối tượng với các tham số mô hình rất phức tạp và mang tính phi tuyến mạnh. Tuy nhiên sau khi đã có các thuật toán nhận dạng tham số mô hình AR và MA, trong đó về thực chất bài toán nhận dạng MA cũng đã được chuyển về bài toán nhận dạng AR tương đương, thì việc

nhận dạng mô hình ARMA cũng sẽ đơn giản hơn và có tính ứng dụng cao hơn nếu ta chuyển bài toán đó thành hai bài toán nhận dạng AR và MA riêng biệt.



Hình 4.3: Chuyển bài toán nhận dạng ARMA thành hai bài toán AR và MA

4.4.1 Nhận dạng tham số AR của mô hình ARMA

Tương tự như đã làm ở mục 4.2, ta nhân cả hai vế của mô hình (4.10) với y_{n-k} về phía trái sau đó lập giá trị trung bình cho cả hai vế, sẽ được

$$r_y(kT_a) + \sum_{i=1}^{n_a} a_i r_y((k-i)T_a) = r_{yu}(kT_a) + \sum_{i=1}^{n_b} b_i r_{yu}((k-i)T_a)\,.$$

Do đối tượng có hệ số khuếch đại bằng 1 nên nếu gọi $g_k = g(kT_a)$ là giá trị hàm hàm trọng lượng của mô hình ARMA thì theo giả thiết (4.9) có

$$g_k = \begin{cases} 1 & \text{khi} \quad k = 0 \\ 0 & \text{khi} \quad k < 0 \end{cases} \qquad \text{và} \qquad r_u(kT_a) = \begin{cases} K & \text{khi} \quad k = 0 \\ 0 & \text{khi} \quad k \neq 0 \end{cases}$$

Nhưng vì

$$r_{yu}(kT_a) = r_{uy}(-kT_a) = r_u(-kT_a) * g_{-k} = \sum_{i=-\infty}^{\infty} r_u(-iT_a) g_{-k-i} = Kg_{-k}$$

nên

$$r_{yu}(kT_a) = \begin{cases} K & \text{khi} \quad k = 0 \\ 0 & \text{khi} \quad k > 0 \\ Kg_{-k} & \text{khi} \quad k < 0 \end{cases}$$

Bởi vậy với $k > n_b$ sẽ được

$$r_y(kT_a) + \sum_{i=1}^{n_a} a_i r_y((k-i)T_a) = 0. \tag{4.54}$$

Tuy nhiên (4.54) cũng chỉ là công thức gần đúng. Lý do là các giá trị $r_y(kT_a)$ của hàm tương quan chỉ có thể là các giá trị xấp xỉ, chúng có được là bởi ta đã áp dụng các thuật toán nhận dạng hàm tương quan từ dãy $\{y_k\}$ đo được của tín hiệu đầu ra (ví dụ như với hàm `cor()`). Nói cách khác

$$r_y(kT_a) + \sum_{i=1}^{n_a} a_i r_y((k-i)T_a) = e_k \neq 0$$

Bởi vậy các tham số $a_1, a_2, \dots, a_{n_a}$ cần phải được xác định sao cho giá trị trung bình của bình phương sai lệch $M\{e_n^2\}$ là nhỏ nhất. So sánh với định lý 4.1 thì ở đây $a_1, a_2, \dots, a_{n_a}$ chính là nghiệm của bài toán Yule–Walker ứng với các giá trị đầu vào $r_y[(n_b+1)T_a]$, $r_y[(n_b+2)T_a], \cdots, r_y(MT_a)$ thay cho dãy $\{y_k\}$, trong đó M là số Lag ta đã chọn khi sử dụng thuật toán nhận dạng hàm tương quan.

Ta đi đến thuật toán nhận dạng tham số AR của mô hình ARMA như sau:

1) Chọn số Lag $M \gg n_a$ và sử dụng chương trình nhận dạng hàm tương quan **`cor()`** để xác định dãy giá trị $r_y(kT_a)$, $k = 0,1, \dots, M$.

2) Sử dụng các chương trình nhận dạng off–line tham số mô hình AR đã biết như **`Yule_Walker()`** hay **`Burg()`** để xác định $a_1, a_2, \dots, a_{n_a}$ từ dãy các giá trị $r_y[(n_b+1)T_a], r_y[(n_b+2)T_a], \cdots, r_y(MT_a)$ của hàm tương quan.

4.4.2 Nhận dạng tham số MA của mô hình ARMA

Bước tiếp theo sau khi đã có các tham số $a_1, a_2, \dots, a_{n_a}$ là áp dụng chương trình nhận dạng tham số mô hình MA (ví dụ như **`MA()`**) để tính các tham số $b_1, b_2, \dots, b_{n_b}$. Muốn như vậy ta cần phải có các giá trị của tín hiệu đầu ra $x(t)$ của khối MA tương ứng trong mô hình ARMA (hình 4.3) và tín hiệu này cũng chính là tín hiệu đầu vào của khối AR nên được xác định theo (4.11), tức là

$$x_k = y_k + \sum_{i=1}^{n_a} a_i y_{k-i} \tag{4.55}$$

Như vậy việc tính các tham số MA của mô hình ARMA sẽ gồm hai bước:

1) Xác định dãy $\{x_k\}$ từ $\{y_k\}$ và $a_1, a_2, \dots, a_{n_a}$ theo (4.55)

2) Áp dụng chương trình **`MA()`** để tính $b_1, b_2, \dots, b_{n_b}$ với dãy đầu vào là $\{x_k\}$.

Nhưng do dãy $\{y_k\}$ có độ dài là N nên dãy $\{x_k\}$ cũng chỉ có thể có độ dài lớn nhất là $N+n_a$, tức là khi $k \geq N+n_a-1$ thì $x_k \equiv 0$. Ngoài ra do tổng bên vế phải của (4.55) có dạng một tích chập của hai dãy giá trị $\{1, a_1, a_2, \dots, a_{n_a}\}$, $\{y_k\}$ và ảnh Fourier của một tích chập chính là tích của hai ảnh Fourier nên ta có thể áp dụng **`dft()`** để tính nhanh $\{x_k\}$ như sau

1) Áp dụng chương trình **dft()** để tính ảnh $A(jk\Omega\lambda)$ của dãy $\{1, a_1, a_2, \dots, a_{n_a}\}$ và $Y(jk\Omega\lambda)$ của $\{y_k\}$ với $\Omega\lambda = \dfrac{2\pi}{\lambda T_a}$.

2) Tính $X(jk\Omega\lambda) = A(jk\Omega\lambda) \cdot Y(jk\Omega\lambda)$.

3) Áp dụng chương trình **invdft()** để tính ngược x_k từ $X(jk\Omega\lambda)$.

Chú ý: Tích $X(jk\Omega\lambda)=A(jk\Omega\lambda)\cdot Y(jk\Omega\lambda)$ chỉ có ý nghĩa khi $\{A(jk\Omega\lambda)\}$, $\{Y(jk\Omega\lambda)\}$ có cùng độ dài λ, trong đó λ là số nguyên dạng lũy thừa 2 nhỏ nhất nhưng không nhỏ hơn N và n_a. Do $N > n_a$ nên trước khi sử dụng **dft()** ở bước 1 ta phải thêm các phần tử 0 vào dãy $\{1, a_1, a_2, \dots, a_{n_a}\}$ sao cho những dãy mới này có đúng N phần tử (xem thêm hàm **dft()** đã giới thiệu ở mục 2.1.7 trong chương 2).

4.4.3 Thuật toán nhận dạng tham số mô hình ARMA

Tổng kết hai phần trên lại ta có thuật toán chung để nhận dạng tham số mô hình ARMA như sau:

1) Chọn số Lag $M \gg n_a$ và sử dụng hàm **cor()** để nhận dạng hàm tương quan, tức là xác định dãy giá trị $r_y(kT_a)$, k=0,1, … , M.

2) Sử dụng các chương trình nhận dạng chủ động tham số mô hình AR đã biết như **Yule_Walker()** hay **Burg()** để xác định $a_1, a_2, \dots, a_{n_a}$ từ dãy các giá trị $r_y[(n_b+1)T_a]$, $r_y[(n_b+2)T_a]$, ⋯ , $r_y(MT_a)$ của hàm tương quan.

3) Xác định dãy $\{x_k\}$ từ $\{y_k\}$ và $a_1, a_2, \dots, a_{n_a}$ trực tiếp theo (4.55) hoặc có thể tính nhanh theo các bước:

   a) Lập dãy $\{1, a_1, \dots, a_{n_a}, 0, \dots, 0\}$ bằng cách thêm các giá trị 0 vào cuối dãy sao cho dãy mới có đúng N phần tử.

   b) Tính ảnh Fourier $A(jk\Omega\lambda)$ của $\{1, a_1, \dots, a_{n_a}, 0, \dots, 0\}$ và ảnh $Y(jk\Omega\lambda)$ của $\{y_k\}$ nhờ **dft()**.

   c) Lập tích $X(jk\Omega\lambda) = A(jk\Omega\lambda) \cdot Y(jk\Omega\lambda)$, k=0,1, … , $\lambda-1$.

   d) Áp dụng **invdft()** với dãy đầu vào $\{\overline{X(jk\Omega_\lambda)}\}$ để tính ngược $\{x_k\}$ từ $\{X(jk\Omega\lambda)\}$.

4) Áp dụng hàm **MA()** để tính $K, b_1, b_2, \ldots, b_{n_b}$ từ $\{x_k\}$.

Một hàm **ARMA()** viết trên ngôn ngữ lập trình C mô tả việc cài đặt thuật toán trên được cho dưới đây để tham khảo.

Hàm **ARMA()** có các biến hình thức sau:

a) Con trỏ **y** chỉ đầu mảng **y[]** chứa dãy các giá trị tín hiệu $\{y_k\}$, k = 0,1, … , N–1 đo được.

b) Số nguyên **N** chứa số các giá trị y_k, tức là chứa N.

c) Số nguyên **na** chứa bậc của đa thức mẫu số của mô hình, tức là n_a.

d) Số nguyên **nb** chứa bậc của đa thức tử số của mô hình, tức là n_b.

e) Số nguyên **s** chứa bậc mô hình AR được thay thế tạm thời cho mô hình MA của mô hình ARMA cần nhận dạng.

f) Số nguyên **bias** xác định giá trị hàm tương quan sẽ được tính theo công thức bias (**bias**=1) hay unbias (**bias**≠1).

g) Số nguyên **M** chứa chỉ số Lag cần thiết cho việc nhận dạng hàm tương quan.

h) Con trỏ **a** chỉ đầu mảng **a[]** có độ dài n_a+1 chứa kết quả theo thứ tự:

a[0]=1 , **a[1]**= a_1, … , **a[na]**= a_{n_a} .

i) Con trỏ **b** chỉ đầu mảng **b[]** có độ dài n_b+1 chứa kết quả theo thứ tự:

b[0]=K , **b[1]**= b_1, … , **b[nb]**= b_{n_b} .

Hàm **ARMA()** trả về giá trị báo lỗi 3 nếu $M \geq N$, giá trị 4 khi không thực hiện được hàm **Yule_Walker()** và giá trị 1 hoặc 2 khi hàm **MA()** có lỗi. Trường hợp không có lỗi, hàm **ARMA()** sẽ trả về giá trị 0.

Hàm **ARMA()** sử dụng thuật toán Yule–Walker, tức là hàm **Yule_Walker()** cho bước 2) để xác định $a_1, a_2, \ldots, a_{n_a}$ với các giá trị đầu vào $r_y[(n_b+1)T_a]$, $r_y[(n_b+2)T_a]$, ⋯ , $r_y(MT_a)$ thay cho dãy $\{y_k\}$.

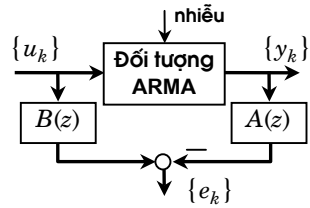
```
int ARMA(double *y,int N,int na,int nb,int s,int bias,int M,
         double *a,double *b)
{  int i,k,err;
   double *x;
   x=new double[N];
   err=2+cor(y,y,N,M,bias,x);
   if(err==2)
      err=3+Yule_Walker(x+nb+1,M-nb,na,bias,a);
```

```
    if(err==3)
    {   a[0]=1.;
        for(k=0;k<N;k++)
        {   x[k]=0.;
            for(i=0;(i<=na)&&(i<=k);i++) x[k]+=a[i]*y[k-i];
        }
        err=MA(x,N,nb,s,bias,b);
    }
    delete [] x;
    return(err);
}
```

4.5 Nhận dạng bị động tham số mô hình ARMA

Bây giờ ta xét bài toán tổng quát hơn cho việc nhận dạng tham số mô hình ARMA. Đó là bài toán nhận dạng bị động (còn gọi là passive hay on–line). Trong bài toán này, tín hiệu đầu vào $u(t)$ không được giả thiết là đã biết trước, nó có thể là một tín hiệu bất kỳ và như vậy để nhận dạng ta phải đo cả tín hiệu vào $u(t)$ lẫn tín hiệu ra $y(t)$.

Bài toán được phát biểu như sau: Cho đối tượng mô tả bởi mô hình rời rạc, được giả thiết là tuyến tính, dưới dạng phương trình vi sai phân (4.1). Từ các giá trị đã đo được của tín hiệu đầu vào là $\{u_k\}$ và của tín hiệu đầu ra là $\{y_k\}$, k=0, … , N, hãy xác định các tham số K, a_1, … , a_{n_a} và b_1, … , b_{n_b} của mô hình (4.1) sao cho sai lệch giữa mô hình và đối tượng là nhỏ nhất.



Hình 4.4: Nhận dạng bị động tham số mô hình ARMA.

Để giải quyết bài toán vừa nêu ta có hai cách:

– Nhận dạng các tham số K, a_1, … , a_{n_a} và b_1, … , b_{n_b} trực tiếp từ dãy các giá trị đo được $\{u_k\}$, $\{y_k\}$ sao cho tổng bình phương sai lệch mở rộng giữa mô hình và đối tượng là nhỏ nhất (hình 4.4).

– Chuyển về bài toán nhận dạng chủ động và sử dụng các thuật toán nhận dạng chủ động tham số mô hình ARMA đã biết.

4.5.1 Nhận dạng bị động khi các tín hiệu vào ra là tiền định

Để đơn giản, ta chuyển mô hình (4.1) về dạng

$$G(z)=\frac{\widetilde{b}_0+\widetilde{b}_1z^{-1}+\cdots+\widetilde{b}_{n_b}z^{-n_b}}{1+a_1z^{-1}+\cdots+a_{n_a}z^{-n_a}}=\frac{B(z)}{A(z)}. \tag{4.56}$$

Như vậy thì so với (4.1), trong (4.56) có

$$\widetilde{b}_i = Kb_i \qquad \text{với} \qquad i= 0,1 \ldots , \ n_b \ \text{ và } \ b_0=1.$$

Dạng tương đương của (4.54) viết trực tiếp theo quan hệ vào ra của tín hiệu trong miền thời gian, là

$$y_k+a_1y_{k-1}+\cdots+a_{n_a}y_{k-n_a}=\widetilde{b}_0u_k+\widetilde{b}_1u_{k-1}+\cdots+\widetilde{b}_{n_b}u_{k-n_b} \tag{4.57}$$

Mô hình (4.57) trên chỉ có thể đúng nếu như $n(t)\equiv 0$ và các giá trị đo được $\{u_k\}$, $\{y_k\}$ là chính xác. Song vì không có một sự đảm bảo nào cho rằng điều đó thực hiện được nên trong thực tế giữa vế phải và vế trái tồn tại một sai số, tức là

$$y_k+\sum_{i=1}^{n_a}a_iy_{k-i}\neq\sum_{i=0}^{n_b}\widetilde{b}_iu_{k-i}$$

Ký hiệu sai số đó là e_k, thì

$$e_k=y_k+\sum_{i=1}^{n_a}a_iy_{k-i}-\sum_{i=0}^{n_b}\widetilde{b}_iu_{k-i} \tag{4.58}$$

Giá trị tổng bình phương sai lệch mở rộng khi đó được viết thành:

$$Q=\sum_{k=n_a}^{N}e_k^2=\sum_{k=n_a}^{N}\left(y_k+\sum_{i=1}^{n_a}a_iy_{k-i}-\sum_{i=0}^{n_b}\widetilde{b}_iu_{k-i}\right)^2 \tag{4.59}$$

trong đó, để có thể chỉ những giá trị u_k, y_k đã đo được trong khoảng $k= 0,1, \ldots , N$, tham gia vào thuật toán nhận dạng thì e_k phải có $k=n_a , \ \ldots \ , N$ và như vậy điều kiện để ứng dụng được thuật toán sẽ là $N \geq n_a$.

Bài toán nhận dạng bây giờ được phát biểu như sau: Trên cơ sở quan sát các tín hiệu vào ra, hãy xác định $a_1 , \ldots , a_{n_a}$ và $\widetilde{b}_0, \ \ldots \ , \widetilde{b}_{n_b}$ sao cho $Q\rightarrow$ min.

Viết lại (4.59) với các ký hiệu vector

$$\underline{y} = \begin{pmatrix} y_{n_a} \\ \vdots \\ y_N \end{pmatrix},\ \underline{p} = \begin{pmatrix} -a_1 \\ \vdots \\ -a_{n_a} \\ \widetilde{b}_0 \\ \vdots \\ \widetilde{b}_{n_b} \end{pmatrix},\ M = \begin{pmatrix} y_{n_a-1} & \cdots & y_0 & u_{n_a} & \cdots & u_{n_a-n_b} \\ y_{n_a} & \cdots & y_1 & u_{n_a+1} & \cdots & u_{n_a-n_b+1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ y_{N-1} & \cdots & y_{N-n_a} & u_N & \cdots & u_{N-n_b} \end{pmatrix} \tag{4.60}$$

ta sẽ được

$$Q = (\underline{y} - M\underline{p})^T(\underline{y} - M\underline{p}) = \underline{y}^T\underline{y} - \underline{y}^T M\,\underline{p} - \underline{p}^T M^T\underline{y} + \underline{p}^T M^T M\,\underline{p}.$$

Giả thiết thêm ma trận vuông $S=M^TM$ *không suy biến*, vậy thì

$$Q = \underline{y}^T\underline{y} - \underline{y}^T MS^{-1}M^T\underline{y} + (M^T\underline{y} - S\,\underline{p})^T S^{-1}(M^T\underline{y} - S\,\underline{p}).$$

Đẳng thức trên có thành phần thứ ba bên vế phải là thành phần duy nhất chứa vector $\underline{p}$ phải tìm. Bởi vậy, do S *xác định dương* nên Q sẽ nhận giá trị nhỏ nhất khi và chỉ khi

$$\underline{0} = M^T\underline{y} - S\,\underline{p} \quad \Leftrightarrow \quad \underline{p} = S^{-1}M^T\underline{y} \tag{4.61}$$

và đó chính là công thức xác định các tham số của mô hình ARMA. Ta đi đến thuật toán:

1) Lập vector $\underline{y}$ và ma trận M, S theo (4.60) từ dãy các giá trị $\{u_k\}$, $\{y_k\}$, k=0, … , N đã đo được của tín hiệu.
2) Kiểm tra tính không suy biến cũng như tính xác định dương của ma trận S.
3) Tính vector $\underline{p}$ của tham số mô hình theo (4.61). Có thể sử dụng hàm **`cholesky()`** đã trình bày trong chương 3 để giải hệ phương trình tuyến tính này.

Chú ý: Trong khi vector tham số $\underline{p}$ có số chiều cố định là n_a+n_b+1 thì số chiều của ma trận M và vector $\underline{y}$ phụ thuộc vào số mẫu tín hiệu trích được. Số lượng mẫu tín hiệu trích được càng lớn, khoảng thời gian quan sát tín hiệu sẽ càng rộng, lượng thông tin thu được về đối tượng càng nhiều, song chiều của ma trận M và vector $\underline{y}$ cũng vì thế mà càng tăng kéo theo số lượng các phép tính phải thực hiện lớn, làm cho sai số tính toán cũng lớn. Bởi vậy ta cần phải chọn khoảng thời gian quan sát không nên quá ngắn nhưng cũng không quá lâu và kết quả nhận dạng sẽ phụ thuộc đáng kể vào khoảng thời gian quan sát tín hiệu được chọn. Ngoài ra thuật toán trên không có khả năng lọc nhiễu nên chỉ có thể ứng dụng cho các bài toán nhận dạng mà sự ảnh hưởng của nhiễu vào đối tượng là không đáng kể và có thể bỏ qua được. Cũng như vậy thuật toán cũng sẽ không sử dụng được khi tín hiệu vào/ra là những tín hiệu ngẫu nhiên.

Nếu gọi s_{ij} với i, j =1, 2, … , n_a+n_b+1 là các phần tử của S=$M^T M$ thì từ (4.60) có:

$$s_{ij} = \begin{cases} (N - n_a + 1) \cdot r_y((i-j)T_a) & \text{nếu} \quad 1 \le j \le i \le n_a \\ (N - n_a + 1) \cdot r_{yu}((i-j-n_a-1)T_a) & \text{nếu} \quad 1 \le j \le n_a < i \le n_a + n_b + 1 \\ (N - n_a + 1) \cdot r_u((i-j)T_a) & \text{nếu} \quad n_a < j \le i \le n_a + n_b + 1 \end{cases}$$

trong đó $r_y(mT_a)$, $r_{yu}(mT_a)$, $r_u(mT_a)$ là giá trị hàm tương quan được xác định theo công thức bias (2.45) và (2.46) và T_a là khoảng thời gian trích mẫu tín hiệu. Cũng như vậy nếu ký hiệu k_i với i =1, 2, … , n_a+n_b+1 là những phần tử của vector $\underline{k}$=M^T $\underline{y}$ thì:

$$k_i = \begin{cases} (N - n_a + 1) \cdot r_y(iT_a) & \text{nếu} \quad 1 \le i \le n_a \\ (N - n_a + 1) \cdot r_{uy}((i-n_a-1)T_a) & \text{nếu} \quad n_a < i \le n_a + n_b + 1 \end{cases}.$$

Hai công thức này là công cụ để xây dựng dãy giá trị đầu vào cho hàm **cholesky()** khi cài đặt thuật toán trên. Hàm **online_d()** viết trên ngôn ngữ lập trình C cho dưới đây là một ví dụ về việc cài đặt thuật toán nhận dạng bị động tham số mô hình ARMA khi các tín hiệu vào ra là tiền định.

Hàm **online_d()** sử dụng thêm hàm **cor()** phục vụ việc thực hiện hai công thức tính s_{ij} , k_i , $j \le i$ =1, 2, … , n_a+n_b+1 và có các biến hình thức sau:

a) Con trỏ **u** chỉ đầu mảng thực **u[]** chứa dãy các giá trị $\{u_k\}$, k = 0,1, … , N đo được của tín hiệu vào.

b) Con trỏ **y** chỉ đầu mảng thực **y[]** chứa dãy các giá trị $\{y_k\}$, k = 0,1, … , N đo được của tín hiệu ra.

c) Số nguyên **N** chứa số các giá trị tín hiệu đo được, tức là chứa N+1.

d) Số nguyên **na** chứa bậc n_a của đa thức mẫu số của mô hình.

e) Số nguyên **nb** chứa bậc n_b của đa thức tử số của mô hình.

f) Con trỏ **p** chỉ đầu mảng phức **p[]** chứa tham số tìm được theo thức tự

p[0]= $-a_1$, … , **p[na-1]**= $-a_{n_a}$, **p[na]**= $\tilde{b}_0$, … , **p[na+nb]**= $\tilde{b}_{n_b}$.

Mặc dù tham số $\underline{a}$, $\underline{b}$ mô hình chỉ là những số thực, song để tiện cho việc sử dụng hàm **cholesky()** chúng được khai báo ở đây là các giá trị phức. Phần ảo của kết quả nếu khác 0 có thể được xem như sai số tính toán của chương trình.

Hàm trả về giá trị báo lỗi ≠ 0 khi $N \le n_a + n_b$ hoặc hàm con **cor()** hay **cholesky()** có lỗi, trường hợp ngược lại sẽ trả về giá trị 0. Hàm **online_d()** không làm thay đổi nội dung hai mảng **u[]** và **y[]**.

```
int online_d(double *u,double *y,int N,int na,int nb,complex *p)
{
   if(N<=na+nb) return(1);
   int n=na+nb+1,i,j,q,l,err;
   double *ru,*ruy,*ryu,*ry;
   complex *s;
   ry=new double [na+1];
   ru=new double [nb+1];
   ruy=new double [nb+1];
   ryu=new double [na+1];
   s=new complex [n*(n+1)/2];
   err = cor(y,y,N+1,na,1,ry)+cor(u,y,N+1,nb,1,ruy);
   err += cor(y,u,N+1,na,1,ryu)+cor(u,u,N+1,nb,1,ru);
   if(err==0)
   {  for(i=1;i<=n;i++)
      {  q=i*(i-1)/2-1;
         p[i-1]=(i<=na)? complex(ry[i]):complex(ruy[i-na-1]);
         for(j=1;j<=i;j++)
         {  if(i<=na) s[q+j]=complex(ry[i-j]);
            else if(j<=na)
            {  l=i-j-na-1;
               s[q+j]=(l<0)? complex(ryu[-l]):complex(ruy[l]);
            }
            else s[q+j]=complex(ru[i-j]);
         }
      }
      err=cholesky(n,s,p);
   }
   delete[]ru; delete[]ruy; delete[]ryu; delete[]ry; delete[]s;
   return(err);
}
```

4.5.2 Nhận dạng bị động với các tín hiệu vào ra là ngẫu nhiên

Xét bài toán nhận dạng bị động tham số mô hình ARMA với các tín hiệu vào/ra là những hàm ngẫu nhiên. Không mất tính tổng quát nếu giả thiết thêm rằng $u(t)$, $y(t)$ và nhiễu là những tín hiệu có phần tĩnh bằng 0. Khi đó, sau khi nhân cả hai vế của (4.57) với với u_{n-m}, rồi lập giá trị trung bình cả hai vế, sẽ được

$$r_{uy}(mT_a)+\sum_{i=1}^{n_a}a_i r_{uy}((m-i)T_a)=\sum_{i=0}^{n_b}\tilde{b}_i r_u((m-i)T_a) \tag{4.62}$$

Hai phương trình (4.57) và (4.62) như vậy là tương đương, trong đó vị trí y_k và u_k trong (4.57) nay đã được thay thế bởi $r_{uy}(mT_a)$ và $r_u(mT_a)$. Do đó, nếu ta lý luận đúng như đã làm với (4.57) để đi đến (4.61) thì khi xuất phát từ (4.62) cũng sẽ được công thức xác

định các tham số $a_1, \ldots, a_{n_a}$ và $\tilde{b}_0, \ldots, \tilde{b}_{n_b}$ theo nguyên tắc cực tiểu hóa tổng bình phương các sai lệch như sau:

$$(\tilde{M}^T\tilde{M})\begin{pmatrix} -a_1 \\ \vdots \\ -a_{n_a} \\ \tilde{b}_0 \\ \vdots \\ \tilde{b}_{n_b} \end{pmatrix} = \tilde{M}^T\begin{pmatrix} r_{uy}(n_aT_a) \\ \vdots \\ r_{uy}(NT_a) \end{pmatrix} \tag{4.63}$$

với $\tilde{M} = \begin{pmatrix} r_{uy}((n_a-1)T_a) & \cdots & r_{uy}(0) & r_u(n_aT_a) & \cdots & r_u((n_a-n_b)T_a) \\ r_{uy}(n_aT_a) & \cdots & r_{uy}(T_a) & r_u((n_a+1)T_a) & \cdots & r_u((n_a-n_b+1)T_a) \\ \vdots & & & & & \\ r_{uy}((N-1)T_a) & \cdots & r_{uy}((N-n_a)T_a) & r_u(NT_a) & \cdots & r_u((N-n_b)T_a) \end{pmatrix}$

và cuối cùng, ta có được thuật toán (dạng thô):

1) Sử dụng chương trình **`cor()`** để tính giá trị các hàm tương quan $r_{uy}(mT_a)$ và $r_u(mT_a)$ từ những giá trị tín hiệu đo được.

2) Xác định các tham số $a_1, \ldots, a_{n_a}$ và $\tilde{b}_0, \ldots, \tilde{b}_{n_b}$ bằng cách giải hệ phương trình (4.63). Ta có thể sử dụng chương trình **`cholesky()`** đã giới thiệu trong chương 3 để giải hệ phương trình này.

Việc áp dụng thuần túy thuật toán trên để xác định tham số thường dẫn tới kết quả có sai số lớn. Lý do nằm ở sai số nhận dạng hàm tương quan $r_{uy}(mT_a)$ và $r_u(mT_a)$ từ N+1 giá trị tín hiệu thu được $\{u_k\}$, $\{y_k\}$, k=0,1, … , N. Như đã trình bày trong chương 2 thì sai số trong $r_{uy}(mT_a)$ và $r_u(mT_a)$ càng lớn khi chỉ số m càng cao. Bởi vậy để loại trừ những sai số này, ta nên áp dụng lại việc dùng số Lag M trong khoảng 5%÷20% của N+1 số các giá trị tín hiệu nhận được nhằm hạn chế sự tham gia tiếp tục của những giá trị $r_{uy}(mT_a)$, $r_u(mT_a)$ có sai số lớn (chương 2, mục 2.2.1). Nói cách khác ta sẽ chỉ sử dụng trong phương trình (4.63) những giá trị $r_{uy}(mT_a)$, $r_u(mT_a)$ có chỉ số m= 0,1, … , M.

Áp dụng chỉ số Lag, phương trình (4.63) trở thành:

$$(\hat{M}^T\hat{M})\begin{pmatrix} -a_1 \\ \vdots \\ -a_{n_a} \\ \tilde{b}_0 \\ \vdots \\ \tilde{b}_{n_b} \end{pmatrix} = \hat{M}^T\begin{pmatrix} r_{uy}(n_aT_a) \\ \vdots \\ r_{uy}(MT_a) \end{pmatrix} \tag{4.64a}$$

với

$$\widehat{M}=\begin{pmatrix} r_{uy}((n_a-1)T_a) & \cdots & r_{uy}(0) & r_u(n_aT_a) & \cdots & r_u((n_a-n_b)T_a) \\ r_{uy}(n_aT_a) & \cdots & r_{uy}(T_a) & r_u((n_a+1)T_a) & \cdots & r_u((n_a-n_b+1)T_a) \\ \vdots & & & & & \\ r_{uy}((M-1)T_a) & \cdots & r_{uy}((M-n_a)T_a) & r_u(MT_a) & \cdots & r_u((M-n_b)T_a) \end{pmatrix} \tag{4.64b}$$

Ta đi đến thuật toán dạng tinh như sau:

1) Chọn số Lag $M \gg n_a$ và sử dụng chương trình nhận dạng hàm tương quan **`cor()`** để xác định dãy giá trị $r_{uy}(mT_a)$ và $r_u(mT_a)$, m=0,1, … , M.

2) Xây dựng ma trận $\widehat{M}$ theo (4.64b) và kiểm tra tính không suy biến của $(\widehat{M}^T\widehat{M})$.

3) Giải hệ phương trình (4.64a) để xác định a_1, … , a_{n_a} và $\widetilde{b}_0$, … , $\widetilde{b}_{n_b}$.

Thêm nữa, thuật toán này sẽ không bị ảnh hưởng bởi nhiễu $n(t)$ tác động tại đầu ra nếu như nhiễu đó không tương quan với tín hiệu đầu vào, vì khi đó có $r_{nu}(\tau)$=0.

Như vậy, thuật toán trên hoàn toàn tương tự như hàm **`online_d()`**. Điều khác biệt duy nhất ở đây là vai trò của dãy $\{u_k\}$, $\{y_k\}$, k=0, … , N nay được thay bằng $r_{uy}(mT_a)$, $r_u(mT_a)$, m= 0,1, … , M.

Hàm **`online_s()`** cho dưới đây minh họa việc cài đặt thuật toán. Hàm có các biến hình thức:

a) Con trỏ **`u`** chỉ đầu mảng thực **`u[]`** chứa dãy các giá trị $\{u_k\}$, k = 0,1, … , N đo được của tín hiệu vào $u(t)$.

b) Con trỏ **`y`** chỉ đầu mảng thực **`y[]`** chứa dãy các giá trị $\{y_k\}$, k = 0,1, … , N đo được của tín hiệu ra $y(t)$.

c) Biến nguyên **`N`** chứa số các giá trị tín hiệu đo được là N+1.

d) Biến nguyên **`M`** chứa chỉ số Lag là M.

e) Biến nguyên **`bias`** xác định giá trị hàm tương quan sẽ được tính theo công thức bias (**`bias`**=1) hay unbias (**`bias`**≠1).

f) Biến nguyên **`na`** chứa bậc n_a của đa thức mẫu số của mô hình ($n_a > 0$).

g) Biến nguyên **`nb`** chứa bậc n_b của đa thức tử số của mô hình ($n_b \le n_a$).

h) Con trỏ **`p`** chỉ đầu mảng phức **`p[]`** chứa tham số tìm được theo thức tự

`p[0]`= $-a_1$, … , **`p[na-1]`**= $-a_{n_a}$, **`p[na]`**= $\widetilde{b}_0$, … , **`p[na+nb]`**= $\widetilde{b}_{n_b}$.

Thực chất các tham số mô hình $a_1, \dots, a_{n_a}$, $\tilde{b}_0, \dots, \tilde{b}_{n_b}$ chỉ là phần thực của **p[]**.

Phần ảo của **p[]** có thể được xem như sai số tính toán của chương trình.

Hàm **online_s()** sử dụng thêm hai hàm con **cor()**, **online_d()**. Hàm báo lỗi bằng giá trị trả về ≠ 0 khi các hàm con này có lỗi, trường hợp ngược lại sẽ trả về giá trị 0. Hàm **online_s()** không làm thay đổi nội dung hai mảng **u[]** và **y[]**.

```
int online_s(double *u,double *y,int N,int M,int bias,
             int na,int nb,complex *p)
{
   if((na+nb==0)||(na<nb)) return(-1);
   int err;
   double *ru,*ruy;
   ru=new double[M+1];
   ruy=new double[M+1];
   err=cor(u,u,N+1,M,bias,ru)+cor(u,y,N+1,M,bias,ruy);
   if(err==0) err=online_d(ru+na-nb,ruy,M,na,nb,p);
   delete [] ru; delete [] ruy;
   return(err);
}
```

4.5.3 Chuyển về bài toán nhận dạng chủ động

Trong các mục 4.2, 4.3, 4.4 ta đã làm quen với các thuật toán nhận dạng chủ động tham số mô hình AR, MA và ARMA khi tín hiệu đầu vào $u(t)$ là ồn trắng, tức là tín hiệu ngẫu nhiên có

$m_u = 0$ và $S_u(\omega)$ = hằng số.

Mục đích chính của mục này là tìm cách ứng dụng những thuật toán đã biết đó để nhận dạng bị động (còn gọi là trực tuyến/online) tham số mô hình ARMA

$$G(z) = \frac{Y(z)}{U(z)} = K\frac{1 + b_1 z^{-1} + \cdots + b_{n_b} z^{-n_b}}{1 + a_1 z^{-1} + \cdots + a_{n_a} z^{-n_a}} = K\frac{B(z)}{A(z)}, \tag{4.65}$$

cho một tín hiệu vào $u(t)$ ngẫu nhiên bất kỳ.

Nếu $u(t)$ là một tín hiệu bất kỳ thì như ngay ở chương 1 mở đầu đã nói, ảnh $\mathcal{Z}$ của nó có thể mô tả được bởi:

$$U(z) = \tilde{K}\frac{C(z)}{F(z)}\tilde{U}(z)$$

trong đó

$$C(z) = 1 + c_1 z^{-1} + \cdots + c_{n_c} z^{-n_c}, \qquad F(z) = 1 + f_1 z^{-1} + \cdots + f_{n_f} z^{-n_f}$$

và $\widetilde{U}(z)$ là ảnh $\mathcal{Z}$ của tín hiệu $\widetilde{u}(t)$ ồn trắng có $m_{\widetilde{u}} = 0$ và $S_{\widetilde{u}}(\omega) = 1$ (hình 4.5). Khi đó bài toán nhận dạng bị động tham số mô hình ARMA sẽ được chuyển về bài toán nhận dạng chủ động với các bước như sau:

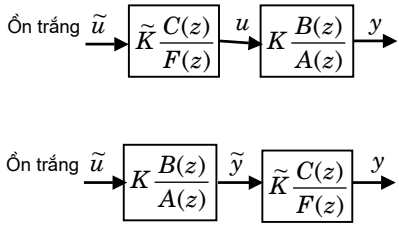
1) Nhận dạng các tham số mô hình $c_1, c_2, \ldots, c_{n_c}, f_1, f_2, \ldots, f_{n_f}$ và $\widetilde{K}$ của tín hiệu ngẫu nhiên đầu vào $u(t)$ từ dãy các giá trị $\{u_k\}$, $k = 0,1, \ldots, N$ đo được của nó bằng các thuật toán nhận dạng chủ động đã biết.

2) Xem mô hình $\widetilde{K}\dfrac{C(z)}{F(z)}$ của tín hiệu đầu vào như một khâu mắc nối tiếp ngay sau đối tượng $K\dfrac{B(z)}{A(z)}$ cần nhận dạng (hình 4.5), thì tín hiệu đầu vào $\widetilde{y}(t)$ của $\widetilde{K}\dfrac{C(z)}{F(z)}$ sẽ đồng thời là tín hiệu ra của đối tượng khi đối tượng được chủ động kích thích bằng tín hiệu ồn trắng $\widetilde{u}(t)$. Xác định dãy các giá trị $\{\widetilde{y}_k\}$ từ $\{y_k\}$, $k = 0,1, \ldots, N$ và mô hình $\widetilde{K}\dfrac{C(z)}{F(z)}$ theo công thức

$$y_k + f_1 y_{k-1} + \cdots + f_{n_f} y_{k-n_f} = \widetilde{K}\left(\widetilde{y}_k + c_1\widetilde{y}_{k-1} + \cdots + c_{n_c}\widetilde{y}_{k-n_c}\right)$$

3) Xác định $a_1, a_2, \ldots, a_{n_a}, b_1, b_2, \ldots, b_{n_b}$ và K của $K\dfrac{B(z)}{A(z)}$ từ dãy $\{\widetilde{y}_k\}$ bằng các thuật toán nhận dạng chủ động (off–line) đã biết.

Như vậy điều kiện để có thể áp dụng được thuật toán trên là thông tin A–priori phải cho ta biết thêm bậc n_c, n_f của mô hình tín hiệu đầu vào hoặc ít nhất cũng phải nhận dạng được chúng trước khi áp dụng thuật toán.



Hình 4.5: Chuyển bài toán on–line thành off–line.

Do mô hình ARMA của đối tượng và tín hiệu vào là dạng tổng quát của các mô hình AR, MA nên tùy theo các điều kiện cho trước của thông tin A–priori khác nhau về bậc của mô hình đối tượng cũng như của tín hiệu vào $u(t)$ mà ta sẽ có được những hình thức khác nhau của thuật toán vừa nêu. Ví dụ:

a) Thuật toán AR_AR nếu $n_c = n_b = 0$, tức là tín hiệu vào $u(t)$ và đối tượng có cùng mô hình AR.

b) Thuật toán AR_MA nếu $n_c = n_a = 0$ hay mô hình tín hiệu vào $u(t)$ là AR còn mô hình đối tượng là MA.

c) Thuật toán AR_ARMA nếu $n_c = 0$ tức là tín hiệu vào $u(t)$ có mô hình AR còn đối tượng có mô hình ARMA.

d) Thuật toán MA_AR nếu $n_b = n_f = 0$, nói cách khác tín hiệu vào $u(t)$ có mô hình MA còn mô hình đối tượng là AR.

e) Thuật toán MA_MA nếu $n_a = n_f = 0$, tức là cả tín hiệu vào $u(t)$ và đối tượng đều mô tả được bởi mô hình MA.

f) Thuật toán MA_ARMA nếu $n_f = 0$, hay cả tín hiệu vào $u(t)$ và đối tượng đều mô tả được bởi mô hình MA.

g) Thuật toán ARMA_AR nếu $n_b = 0$, tức là tín hiệu vào $u(t)$ có mô hình ARMA còn mô hình đối tượng là AR.

h) Thuật toán ARMA_MA cho trường hợp tín hiệu vào $u(t)$ có mô hình ARMA còn đối tượng có mô hình MA ($n_a = 0$).

i) Thuật toán ARMA_ARMA cho trường hợp cả tín hiệu vào $u(t)$ và đối tượng cùng có mô hình ARMA.

Hai dạng b) và d) có nét đặc biệt riêng. Mô hình chung của tín hiệu vào và đối tượng ở những dạng này là một mô hình ARMA có kích thích đầu vào là tín hiệu ồn trắng, trong đó bản thân từng đa thức tử số hay mẫu số là một mô hình. Do đó với những bài toán có mô hình chung dạng AR_MA hay MA_AR như vậy thì xác định dãy các giá trị đầu ra $\{ \tilde{y}_k \}$ là không cần thiết.

Sau đây ta sẽ minh họa việc cài đặt các bước của thuật toán trên cho dạng c), tức là cho mô hình chung dạng AR_ARMA. Những dạng còn lại của thuật toán có thể được cài đặt một cách tương tự.

Giả sử thông tin A–priori còn cho biết thêm rằng tín hiệu vào $u(t)$ mô tả được bởi

$$U(z) = \frac{\tilde{K} \cdot \tilde{U}(z)}{1 + f_1 z^{-1} + \cdots + f_{n_f} z^{-n_f}}$$

trong đó $\tilde{U}(z)$ là ảnh $\mathcal{Z}$ của tín hiệu $\tilde{u}(t)$ ồn trắng có $m_{\tilde{u}} = 0$ và $S_{\tilde{u}}(\omega) = 1$, thì các bước của thuật toán chung nêu trên sẽ được chi tiết hóa thành:

1) Nhận dạng tham số $f_2, \ldots, f_{n_f}$ và $\tilde{K}$ từ dãy các giá trị $\{u_k\}$, $k = 0,1, \ldots, N$ đo được của $u(t)$ nhờ hàm **`Yule_Walker()`** hoặc **`Burg()`**.

2) Xác định dãy $\{\ \widetilde{y}_k\ \}$ từ $\{y_k\}$, k = 0,1, … , N và f_2 , … , f_{n_f} , $\widetilde{K}$ theo công thức

$$y_k + f_1 y_{k-1} + \cdots + f_{n_f} y_{k-n_f} = \widetilde{K}\ \widetilde{y}_k\ .$$

3) Xác định các tham số a_1, a_2 , … , a_{n_a} , b_1, b_2 , … , b_{n_b} và K của mô hình đối tượng từ dãy $\{\ \widetilde{y}_k\ \}$ nhờ hàm **`ARMA()`**.

Thuật toán chi tiết này đã được cài đặt thành hàm **`AR_ARMA()`** cho dưới đây để tham khảo. Hàm sử dụng hàm con **`Burg()`** cho bước 1 và có các biến hình thức sau:

a) Con trỏ **`u`** chỉ đầu mảng thực **`u[]`** chứa dãy giá trị $\{u_k\}$, k = 0,1, … , N đo được của tín hiệu vào $u(t)$.

b) Con trỏ **`y`** chỉ đầu mảng thực **`y[]`** chứa dãy giá trị $\{y_k\}$, k = 0,1, … , N đo được của tín hiệu ra $y(t)$.

c) Biến nguyên **`N`** chứa số các giá trị tín hiệu đo được là N+1.

d) Biến nguyên **`nf`** chứa bậc n_f của mô hình tín hiệu vào $u(t)$.

e) Biến nguyên **`na`** chứa bậc n_a của đa thức mẫu số mô hình ARMA cho đối tượng.

f) Biến nguyên **`nb`** chứa bậc n_b của đa thức tử số mô hình ARMA cho đối tượng.

g) Số nguyên **`s`** chứa bậc mô hình AR được thay thế tạm thời cho mô hình MA của mô hình ARMA cho đối tượng cần nhận dạng.

h) Số nguyên **`bias`** xác định giá trị hàm tương quan sẽ được tính theo công thức bias (**`bias`**=1) hay unbias (**`bias`**≠1).

i) Số nguyên **`M`** chứa chỉ số Lag cần thiết cho việc nhận dạng hàm tương quan.

j) Con trỏ **`a`** chỉ đầu mảng **`a[]`** có độ dài its nhất là max$\{n_a,n_f\}$+1 chứa kết quả theo thứ tự:

`a[0]`=1 , **`a[1]`**= a_1 , … , **`a[na]`**= a_{n_a} .

k) Con trỏ **`b`** chỉ đầu mảng **`b[]`** có độ dài n_b+1 chứa kết quả theo thứ tự:

`b[0]`=K , **`b[1]`**= b_1 , … , **`b[nb]`**= b_{n_b} .

Hàm **`AR_ARMA()`** trả về giá trị báo lỗi 3 nếu $M \geq N$, giá trị 4 khi không thực hiện được hàm **`Burg()`** và giá trị 1 hoặc 2 khi hàm **`MA()`** có lỗi. Trường hợp không có lỗi, hàm sẽ trả về giá trị 0.

Hàm **`AR_ARMA()`** không làm thay đổi nội dung hai mảng **`u[]`** và **`y[]`**.

```
int AR_ARMA(double *u,double *y,int N,int nf,int na,int nb,
            int s,int bias,int M,double *a,double *b)
{
```

```
    int i,k,err;
    double *x,K;
    x=new double [N];
    err=3+Burg(u,N,nf,a);
    if(err==3)
    {
       K=a[0];
       a[0]=1.;
       for(k=0;k<N;k++)
       {
          x[k]=0.;
          for(i=0;(i<=nf)&&(i<=k);i++) x[k]+=a[i]*y[k-i];
          x[k]=x[k]/K;
       }
       err=ARMA(x,N,na,nb,s,bias,M,a,b);
    }
    delete [] x;
    return(err);
}
```

Câu hỏi ôn tập và bài tập

1. Chứng minh rằng nếu hằng số K của mô hình AR bậc n_a là một số dương và được xác định theo thuật toán Yule–Walker thì nó sẽ chính là giá trị trung bình bình phương sai lệch nhỏ nhất:

$$Q_{min} = \min M\left[\left|y_n - y_n^f\right|^2\right] = K, \text{ trong đó } y_n^f = -\sum_{k=1}^{n_a} a_k y_{n-k} .$$

2. Từ kết quả bài 1 người ta thấy khi bậc của mô hình AR là n_a cũng là một tham số cần nhận dạng thì n_a có thể được xác định sao cho K là một số dương càng nhỏ càng tốt. Hãy căn cứ vào công thức (4.22b)

$$K_i = K_{i-1}\left(1 - a_i[i]^2\right)$$

   cũng như tính truy hồi của thuật toán Levinson hoặc Burg để xây dựng một thuật toán nhận dạng bậc n_a cho mô hình AR.

3. Thuật toán nhận dạng chủ động tham số mô hình MA trình bày tại mục 4.3.3 có bốn cách cài đặt, phụ thuộc vào việc chọn hàm **`Yule–Walker()`** hay **`Burg()`** để thực hiện bước 2) và 3). Hàm **`MA()`** đã giới thiệu ở mục 4.3.3 chỉ là một trong 4 cách cài đặt đó. Hãy viết những chương trình thể hiện các cách cài đặt còn lại và so sánh kết quả thử nghiệm với hàm đã cho **`MA()`**.

4. Cùng với 4 cách cài đặt thuật toán nhận dạng tham số mô hình MA đã nói tới trong bài 3, ở thuật toán nhận dạng chủ động tham số mô hình ARMA (mục 4.4.3) ta cũng

sẽ có 8 cách cài đặt (do có thêm hai cách lựa chọn ở bước 2 cho việc nhận dạng tham số $a_1, a_2, \ldots, a_{n_a}$ theo **`Yule–Walker()`** hay **`Burg()`**). Hãy viết những chương trình thể hiện các cách cài đặt còn lại.

5. Với những tín hiệu ngẫu nhiên egodic phức $u(t)$, $y(t)$ người ta định nghĩa hàm tương quan như sau:

$$r_{uy}(\tau) = M[\,\overline{u}(t)y(t+\tau)\,] = \lim_{T\to\infty}\frac{1}{2T}\int_{-T}^{T}\overline{u}(t)y(t+\tau)dt$$

và do đó $r_{uy}(\tau)$ cũng là hàm phức. Gọi $\underline{a}[i] = \begin{pmatrix} a_1[i] \\ \vdots \\ a_i[i] \end{pmatrix}$ là vector nghiệm phức của phương trình Yule–Walker có bậc là i (4.19) với các phần tử phức $r_{uy}(\tau)$. Chứng minh rằng:

a) Nghiệm $\underline{a}[i]$ cũng thỏa mãn

$$H_i\begin{pmatrix} \overline{a}_i[i] \\ \vdots \\ \overline{a}_1[i] \\ 1 \end{pmatrix} = \begin{pmatrix} \underline{0} \\ K_i \end{pmatrix}.$$

b) Giữa nghiệm $\underline{a}[i]$ của phương trình Yule–Walker bậc i và $\underline{a}[i-1]$ bậc i–1 có mối quan hệ

$$a_k[i] = a_k[i-1] + a_i[i]\,\overline{a}_{i-k}[i-1] \qquad \text{với} \quad k = 1, 2, \ldots, i-1.$$

c) Giữa sai lệch dự báo tuyến tính phức của mô hình bậc i là $e_n^f[i]$, $e_n^b[i]$ và của mô hình bậc i–1 là $e_n^f[i-1]$, $e_n^b[i-1]$ có quan hệ truy hồi

$$e_n^f[i] = e_n^f[i-1] + a_i[i]e_{n-1}^b[i-1].$$

$$e_n^b[i] = e_{n-1}^b[i-1] + \overline{a}_i[i]e_n^f[i-1].$$

6. Hãy dựa vào kết quả của bài 5 để xây dựng thuật toán Levinson giải phương trình Yule–Walker phức, tức là ma trận H_{n_a} có các phần tử phức $r_y(mT_a)$ và vector nghiệm $\underline{a} = \begin{pmatrix} a_1 \\ \vdots \\ a_{n_a} \end{pmatrix}$ của nó cũng là vector có những phần tử a_i phức.

7. Cũng dựa vào kết quả của bài 5 câu c) mà chứng minh rằng tham số phức $a_i[i]$ của mô hình AR bậc i làm cho phiếm hàm sai lệch điều hòa

$$Q = \sum_{n=n_a}^{N-1} \left(\left[e_n^f \right]^2 + \left[e_n^b \right]^2 \right)$$

có giá trị nhỏ nhất sẽ là

$$a_i[i] = \frac{-2 \sum_{n=i}^{N-1} \left(\overline{e_n^f[i-1]} \cdot e_{n-1}^b[i-1] \right)}{\sum_{n=i}^{N-1} \left(e_n^f[i-1]^2 + e_{n-1}^b[i-1]^2 \right)}.$$

8. Theo kết quả bài 5, câu c) và bài 7 hãy xây dựng thuật toán truy hồi xác định các tham số phức a_1, a_2 , … , a_{n_a} cho mô hình AR khi tín hiệu vào là hàm ngẫu nhiên phức $u(t)$ kiểu ồn trắng ($m_u = 0$, $S_u(\omega) = 1$) trên cơ sở đo tín hiệu ra $y(t)$, tức là trên cơ sở dãy giá trị $\{y_k\}$, $k = 0,1, \ldots, N-1$.

5 NHỮNG KỸ THUẬT BỔ TRỢ

5.1 DFT thời gian ngắn (SFT)

Trong các chương 2, 3 và 4 ta đã làm quen với những phương pháp nhận dạng có sử dụng kỹ thuật DFT để xử lý phổ tín hiệu đo được nhằm loại bỏ ảnh hưởng của nhiễu và để có được giá trị hàm tương quan hoặc mật đổ phổ tín hiệu. Kỹ thuật DFT này có một nhược điểm là chỉ áp dụng được sau khi đã có đủ các giá trị tín hiệu $x(kT_a)$, k=0,1, … , N–1 trong một khoảng thời gian quan sát $[0, T)$ đủ lớn. Nếu khoảng thời gian quan sát (thời gian đo) tín hiệu quá ngắn, ta sẽ không có đủ lượng thông tin cần thiết cho việc phân tích phổ, song nếu thời gian đo quá lâu thì ta sẽ làm mất tính thời gian thực của phương pháp, hơn nữa khi kéo dài thời gian đo một tín hiệu ngẫu nhiên ta đã vô hình chung giả thiết rằng đó là tín hiệu ngẫu nhiên dừng (*stationary*) mà điều này không phải lúc nào cũng được thỏa mãn.

Phương pháp SFT (viết tắt của từ tiếng Anh *Short time Fourier Transformation*), hay còn gọi là phương pháp DFT thời gian ngắn, trình bày sau đây sẽ khắc phục được nhược điểm trên của DFT kinh điển. Đặc biệt nữa, phương pháp SFT này còn thường được sử dụng để nhận dạng các đối tượng có tham số mô hình thay đổi theo thời gian.

5.1.1 Tư tưởng của phương pháp

Mục đích của nhận dạng phổ tín hiệu là kết quả mật độ phổ $X_a(j\omega)$ thu được phải phản ánh một cách đầy đủ toàn bộ thông tin tần số có trong tín hiệu $x(t)$ trên toàn bộ khoảng thời gian $0 \le t < \infty$:

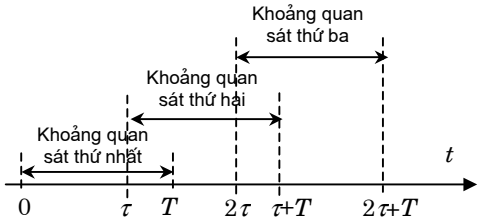
$$X_a(j\omega) = \sum_{k=0}^{\infty} x_k e^{-jkT_a\omega}, \qquad x_k = x(kT_a). \tag{5.1}$$

Nhưng làm thế nào để đạt được mục đích đó trong khi khoảng thời gian quan sát lại bị khống chế bởi T. Tất nhiên rằng ta không thể đi theo hướng như đã làm ở chương 2 là chờ đợi cho tới khi có được hết tất cả các giá trị x_k , k=0, … , ∞ rồi mới tính $X_a(j\omega)$ mà phải suy nghĩ tới một hướng giải quyết khác. Hướng đầu tiên có nhiều khả năng phù hợp là ứng dụng kỹ thuật truy hồi bằng cách chia toàn bộ khoảng thời gian $0 \le t < \infty$ thành những khoảng quan sát $[m\,\tau,\ m\,\tau+T)$, m=0, … , ∞ rồi từ những giá trị x_k đo được trong

khoảng quan sát thứ nhất $[0,T)$ ta tính ngay phổ $X_a^1(j\omega)$ và nó được xem như là giá trị xấp xỉ đầu tiên của $X_a(j\omega)$. Tiếp theo với các giá trị tín hiệu x_k đo được trong khoảng quan sát kế sau $[\tau,T+\tau)$ cũng như $X_a^1(j\omega)$ đã có ta hiệu chỉnh lại $X_a^1(j\omega)$ thành $X_a^2(j\omega)$ theo nghĩa $X_a^2(j\omega)$ gần $X_a(j\omega)$ hơn so với $X_a^1(j\omega)$, trong đó τ là một giá trị thỏa mãn $0\leq\tau<T$ (hình 5.1). Tiếp tục, từ x_k trong khoảng $[2\tau, 2\tau+T)$ và $X_a^2(j\omega)$ ta lại hiệu chỉnh $X_a^2(j\omega)$ thành $X_a^3(j\omega)$ … . Kéo dài quá trình truy hồi đó đến vô hạn ta sẽ được dãy $\{ X_a^i(j\omega) \}$ thỏa mãn

$$\lim_{i\to\infty} X_a^i(j\omega) = X_a(j\omega) \tag{5.2}$$

và đó chính là tư tưởng của phương pháp SFT.



Hình 5.1: Các khoảng quan sát tín hiệu cho từng bước truy hồi.

5.1.2 Thuật toán SFT với hàm cửa sổ Bartlett

Để biểu diễn (5.1) dưới dạng truy hồi trước hết ta xem $x(t)$ trong (5.1) như tích của chính nó với hàm cửa sổ lý tưởng $w_\infty(t)$

$$x(t)=w_\infty(t)x(t) \qquad \text{với} \qquad w_\infty(t)\equiv 1$$

rồi tìm cách biểu diễn $w_\infty(t)$ thông qua các hàm cửa sổ $w_i(t)$ khác có

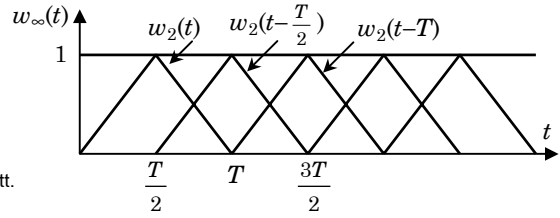
$$\text{supp}\, w_i(t)=[0,T),$$

(mục 2.1.6) ứng với từng khoảng quan sát $[m\tau, m\tau+T)$, $m=0, \dots, \infty$. Một trong những cách biểu diễn như vậy với hàm Bartlett $w_2(t)$ được mô tả trong hình 5.2. Khi đó

$$w_\infty(t) = \sum_{m=-\infty}^{\infty} w_2(t-m\frac{T}{2}).$$

Suy ra

$$x(t) = \sum_{m=-\infty}^{\infty} w_2(t-\frac{mT}{2})x(t) \quad \Rightarrow \quad x(kT_a) = x_k \sum_{m=-\infty}^{\infty} w_2((k-\frac{mN}{2})T_a) \tag{5.3}$$



Hình 5.2: Xây dựng hàm cửa sổ lý tưởng thông qua hàm Bartlett.

Thay (5.3) vào (5.1) và $T=NT_a$ được

$$X_a(j\omega) = \sum_{k=0}^{\infty}\left[\sum_{m=-\infty}^{\infty} x_k w_2((k-\frac{mN}{2})T_a)\right]e^{-jkT_a\omega}$$

$$= \sum_{m=-\infty}^{\infty}\left[\sum_{k=\frac{iN}{2}}^{\frac{mN}{2}+N-1} x_k\, w_2((k-\frac{mN}{2})T_a)\right]e^{-jkT_a\omega}$$

vì $w_2(t)=0$ khi $t \notin [0,T)$, tức là $w_2(k-\frac{mN}{2})T_a) = 0$ khi $k<\frac{mN}{2}$ và $k\geq\frac{mN}{2}+N$.

Trong tổng thứ hai ta thay $q=k-\frac{mN}{2}$ sẽ có

$$X_a(j\omega) = \sum_{m=-\infty}^{\infty}\left[\sum_{q=0}^{N-1} x_{q+\frac{mN}{2}} w_2(qT_a)e^{-jqT_a\omega}\right]e^{-j\frac{mN}{2}T_a\omega} \tag{5.4}$$

Nếu gọi $\{\tilde{x}_q^m\}$ là dãy giá trị tín hiệu $\{x_k\}$ trong khoảng quan sát thứ m, tức là trong khoảng $[m\frac{T}{2},(m+2)\frac{T}{2})$, đã được nhân với hàm cửa sổ Bartlett nhằm giảm sai số rò rỉ:

$$\tilde{x}_q^m = x_{q+\frac{mN}{2}} w_2(qT_a), \quad q=0,1,\ldots,N-1$$

thì (5.4) viết được thành

$$X_a(j\omega) = \sum_{m=-\infty}^{\infty}\left[\sum_{q=0}^{N-1}\tilde{x}_q^m e^{-jqT_a\omega}\right]e^{-j\frac{mN}{2}T_a\omega} = \sum_{m=-\infty}^{\infty}\tilde{X}_a^m(j\omega)e^{-j\frac{mT}{2}\omega} \tag{5.5}$$

trong đó

$$\tilde{X}_a^m(j\omega) = \sum_{q=0}^{N-1}\tilde{x}_q^m e^{-jqT_a\omega} \tag{5.6}$$

là ảnh Fourier của dãy N phần tử $\{ \tilde{x}_q^m \} = \{ \tilde{x}_{\frac{mN}{2}}, \tilde{x}_{\frac{mN}{2}+1}, \dots, \tilde{x}_{\frac{mN}{2}+N-1} \}$.

Công thức (5.5) chính là công thức truy hồi xác định dãy $\{ X_a^i(j\omega) \}$ thỏa mãn điều kiện (5.2) với

$$X_a^i(j\omega) = \sum_{m=-\infty}^{i} \tilde{X}_a^m(j\omega) e^{-j\frac{mT}{2}\omega} = \sum_{m=-\infty}^{i-1} \tilde{X}_a^m(j\omega) e^{-j\frac{mT}{2}\omega} + \tilde{X}_a^i(j\omega) e^{-j\frac{iT}{2}\omega}$$

$$\Rightarrow \qquad X_a^i(j\omega) = X_a^{i-1}(j\omega) + \tilde{X}_a^i(j\omega) e^{-j\frac{iT}{2}\omega}. \tag{5.7}$$

Nếu sử dụng hàm **`fft()`** để tính nhanh công thức (5.6) ta sẽ có thuật toán nhận dạng

$$X(jn\Omega) \approx T_a X_a^i(jn\Omega)$$

theo công thức truy hồi (5.7), trong đó $\Omega = \frac{2\pi}{NT_a}$, như sau:

1) Đặt giá trị khởi phát $X_a^{-1}(jn\Omega) = 0$, n=0,1, … ,N–1 và $\Omega = \frac{2\pi}{NT_a}$.

2) Thực hiện các bước sau ứng với khoảng quan sát $[i\tau,\ i\tau+T)$, i = 0, 1, 2, …

   a) Đo tín hiệu $x(t)$ trong khoảng thời gian $[\frac{iT}{2}, \frac{(i+2)T}{2})$ và gọi dãy gồm N các giá trị nhận được là $\{ x_k^i \} = \{ x_{\frac{iN}{2}}, x_{\frac{iN}{2}+1}, \dots, x_{\frac{iN}{2}+N-1} \}$.

   b) Nhân từng phần tử của dãy trên với hàm cửa sổ Bartlett $w_2(t)$ xác định trong khoảng thời gian $0 \le t < T$ theo công thức

   $\tilde{x}_k^i = x_{k+\frac{iN}{2}} w_2(kT_a)$, k=0,1, … , N–1

   và gọi dãy mới là $\{ \tilde{x}_k^i \} = \{ \tilde{x}_0^i, \tilde{x}_1^i, \dots, \tilde{x}_{N-1}^i \}$.

   c) Sử dụng **`fft()`** để tính $\{ \tilde{X}_a^i(jn\Omega) \}$, n=0,1, … ,N–1 của dãy $\{ \tilde{x}_k^i \}$.

   d) Tính $X_a^i(jn\Omega) = X_a^{i-1}(jn\Omega) + \tilde{X}_a^i(jn\Omega) e^{-jin\pi}$, n=0,1, … ,N–1.

   e) Cho ra kết quả $X(jn\Omega) = T_a X_a^i(jn\Omega)$, n=0,1, … ,N–1 và đó được xem như kết quả truy hồi tại bước thứ i cho ảnh Fourier của $x(t)$.

5.1.3 Thuật toán SFT với một hàm cửa sổ bất kỳ

Thuật toán vừa nêu hoàn toàn có thể mở rộng ra cho một hàm cửa sổ $w(t)$ bất kỳ chứ không phải chỉ riêng cho hàm Bartlett $w_2(t)$. Với một hàm cửa sổ $w(t)$ nào đó, xác định trong khoảng $[0, T)$ và ngoài khoảng này được xem như bằng 0

$$w(t) = 0 \qquad \text{khi} \quad t \notin [0, T)$$

thì hàm $w_\infty(t)$ được biểu diễn thông qua $w(t)$ như sau

$$w_\infty(t) = K \sum_{m=-\infty}^{\infty} w(t - m\tau),$$

trong đó τ được gọi là bước trượt, tức là bước mà các khoảng thời gian quan sát tín hiệu $[m\tau, m\tau+T)$, $m=0, \ldots, \infty$ trượt dọc theo trục thời gian.

Vấn đề đặt ra ở đây là bước trượt τ và hằng số K phải thỏa mãn điều kiện gì để có

$$K \sum_{m=-\infty}^{\infty} w(t - m\tau) = 1 \quad \text{với mọi } t \tag{5.8}$$

và để trả lời câu hỏi này, trước hết ta chứng minh định lý sau:

Định lý 5.1: (Công thức Poisson) Một tín hiệu $x(t)$ có ảnh Fourier $X(j\omega)$ luôn thỏa mãn

$$\sum_{m=-\infty}^{\infty} x(t - m\tau) = \frac{1}{\tau} \sum_{n=-\infty}^{\infty} X(jn\Omega_\tau) e^{-jn\Omega_\tau t} \quad \text{với} \quad \Omega_\tau = \frac{2\pi}{\tau}. \tag{5.9}$$

Chứng minh: Nếu gọi $s_\tau(t)$ là hàm răng lược với khoảng cách bước răng τ, tức là

$$s_\tau(t) = \sum_{m=-\infty}^{\infty} \delta(t - m\tau)$$

thì theo định lý 2.2 đã được trình bày ở mục 2.1.4, nó sẽ có ảnh Fourier

$$S_\tau(j\omega) = \Omega_\tau \sum_{n=-\infty}^{\infty} \delta(\omega - n\Omega_\tau).$$

Mặt khác, nếu xem vé trái của (5.9) là hàm mở rộng biểu diễn dãy các giá trị của $x(t)$ được trích mẫu với thời gian trích mẫu τ:

$$\sum_{m=-\infty}^{\infty} x(t - m\tau) = \sum_{m=-\infty}^{\infty} \int_{-\infty}^{\infty} x(t - t')\delta(t' - m\tau)dt' = x(t) * s_\tau(t)$$

ta sẽ được

$$\sum_{m=-\infty}^{\infty} x(t - m\tau) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\omega) S_\tau(j\omega) e^{j\omega\tau} d\omega$$

$$= \frac{\Omega_\tau}{2\pi} \int_{-\infty}^{\infty} X(j\omega) \sum_{n=-\infty}^{\infty} \delta(\omega - n\Omega_\tau) e^{j\omega\tau} d\omega$$

$$= \frac{1}{\tau} \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} X(j\omega) e^{j\omega\tau} \delta(\omega - n\Omega_\tau) d\omega$$

$$= \frac{1}{\tau} \sum_{n=-\infty}^{\infty} X(jn\Omega_\tau) e^{jn\Omega_\tau \tau} . \qquad \text{(đ.p.c.m.)} \qquad \square$$

Bây giờ ta quay lại bài toán tìm điều kiện cho τ và hằng số K để có (5.8). Theo định lý 5.1 thì (5.8) sẽ được thỏa mãn nếu

$$\frac{1}{\tau} \sum_{n=-\infty}^{\infty} W(jn\Omega_\tau) e^{jn\Omega_\tau \tau} = \frac{1}{K}$$

$$\Leftrightarrow \qquad \frac{1}{\tau} \left(W(0) + \sum_{\substack{n=-\infty \\ n\neq 0}}^{\infty} W(jn\Omega_\tau) e^{jn\Omega_\tau \tau} \right) = \frac{1}{K} \qquad (5.10)$$

trong đó $W(j\omega)$ là ký hiệu chỉ ảnh Fourier của hàm cửa sổ $w(t)$ có supp $w(t)=[0,T)$ đã chọn.

Giả sử tiếp rằng ω_τ là tần số giới hạn của $W(j\omega)$, tức là

$$|W(j\omega)| = 0 \quad \text{khi } |\omega| > \omega_\tau$$

(hoặc ít ra khi $|\omega| > \omega_\tau$ có $|W(j\omega)|$ đủ nhỏ để có thể bỏ qua được) thì khi chọn bước trượt τ thỏa mãn

$$\Omega_\tau > \omega_\tau \qquad \Leftrightarrow \qquad \frac{2\pi}{\omega_\tau} > \tau \qquad (5.11a)$$

cũng như chọn hằng số K với

$$K = \frac{\tau}{W(0)} \qquad (5.11b)$$

sẽ được

$$w_\infty(t) = K \sum_{m=-\infty}^{\infty} w(t - m\tau) = \frac{K}{\tau} \left(W(0) + \sum_{\substack{n=-\infty \\ n\neq 0}}^{\infty} W(jn\Omega_\tau) e^{jn\Omega_\tau \tau} \right) = \frac{K}{\tau} W(0) = 1.$$

Ta đi đến kết luận:

Định lý 5.2: Nếu hàm cửa sổ $w(t)$, $t \in [0,T)$ được chọn có $|W(j\omega)| \approx 0$ khi $|\omega| > \omega_\tau$ thì với bước trượt τ thỏa mãn (5.11a) cũng như hằng số K thỏa mãn (5.11b) sẽ có

$$w_\infty(t) = K \sum_{m=-\infty}^{\infty} w(t - m\tau) \approx 1. \qquad (5.12)$$

Sau khi đã có hàm $w_\infty(t)$, ta thay vào công thức DFT tính ảnh Fourier của $x(t)$

$$X_a(j\omega) = \sum_{k=0}^{\infty} x_k e^{-j\omega kT_a} = K\sum_{k=0}^{\infty} x_k w_\infty(kT_a) e^{-j\omega kT_a}$$

$$= K\sum_{k=0}^{\infty} x_k \left[\sum_{m=-\infty}^{\infty} w((k-mD)T_a)\right] e^{-j\omega kT_a}$$

trong đó D là số nguyên lớn nhất thỏa mãn $D \le \frac{\tau}{T_a}$, rồi đổi vị trí hai dấu tổng sẽ có

$$X_a(j\omega) = K\sum_{m=-\infty}^{\infty} \left[\sum_{k=0}^{\infty} x_k w((k-mD)T_a)\right] e^{-j\omega kT_a}$$

Nhưng vì $w(t)$=0 khi $t \notin [0, T)$ nên

$$X_a(j\omega) = K\sum_{m=-\infty}^{\infty} \left[\sum_{k=mD}^{mD+N-1} x_k w((k-mD)T_a)\right] e^{-j\omega kT_a}$$

$$= K\sum_{m=-\infty}^{\infty} \left[\sum_{k=0}^{N-1} x_{k+mD} w(kT_a)\right] e^{-j\omega(k+mD)T_a}$$

trong đó N là số nguyên nhỏ nhất thỏa mãn $N \ge \frac{T}{T_a}$.

Nếu xem các giá trị tín hiệu $\{ x_k^m \} = \{ x_{mD}, x_{mD+1}, \ldots, x_{mD+N-1} \}$ đo được trong khoảng quan sát thứ m, tức là trong khoảng thời gian $[m\tau, m\tau+T)$ và đã được nhân với giá trị hàm cửa sổ $w(kT_a)$ là dãy

$$\{ \tilde{x}_k^m \} = \{ x_{mD} w(0), x_{mD+1} w(T_a), \ldots, x_{mD+N-1} w((N-1)T_a) \}.$$

thì công thức trên trở thành

$$X_a(j\omega) = K\sum_{m=-\infty}^{\infty} \left[\sum_{k=0}^{N-1} \tilde{x}_k e^{-j\omega kT_a}\right] e^{-j\omega mDT_a}$$

$$\Rightarrow \quad X_a(jn\Omega) = K\sum_{m=-\infty}^{\infty} \left[\sum_{k=0}^{N-1} \tilde{x}_k e^{-jkn\frac{2\pi}{N}}\right] e^{-jmnD\frac{2\pi}{N}}$$

trong đó $\omega = n\Omega = \frac{2\pi n}{NT_a}$. Với ký hiệu

$$\tilde{X}_a^m(jn\Omega) = \sum_{k=0}^{N-1} \tilde{x}_k e^{-jkn\frac{2\pi}{N}}$$

và

$$X_a^m(jn\Omega) = \sum_{l=-\infty}^{m}\left[\sum_{k=0}^{N-1}\widetilde{x}_k e^{-jkn\frac{2\pi}{N}}\right]e^{-jnlD\frac{2\pi}{N}}$$

công thức trên trở thành công thức truy hồi

$$X_a^m(jn\Omega) = X_a^{m-1}(jn\Omega) + \widetilde{X}_a^{m-1}(jn\Omega)e^{-jmnD\frac{2\pi}{N}}$$

thỏa mãn điều kiện

$$K\lim_{m\to\infty} X_a^m(jn\Omega) = X_a(jn\Omega) \qquad \Leftrightarrow \qquad KT_a\lim_{m\to\infty} X_a^m(jn\Omega) = X(jn\Omega).$$

Ta đi đến thuật toán:

1) Chọn một hàm cửa sổ $w(t)$, $t\in[0,T)$.

2) Chọn bước trượt τ thỏa mãn (5.11a) và hằng số K thỏa mãn (5.11b).

3) Đặt giá trị khởi phát $X_a^{-1}(jn\Omega) = 0$, n=0,1, … , N–1 và $\Omega = \dfrac{2\pi}{NT_a}$.

4) Thực hiện lần lượt các bước sau cho từng khoảng quan sát tín hiệu $[m\tau,\ m\tau+T)$, m=0, 1, 2, …

   a) Đo tín hiệu $x(t)$ trong khoảng thời gian $[m\tau, m\tau+T)$ và gọi dãy gồm N các giá trị nhận được là $\{ x_k^m \}=\{ x_{mD}, x_{mD+1}, \dots, x_{mD+N-1} \}$.

   b) Nhân từng phần tử của dãy trên với hàm cửa sổ $w(t)$ theo công thức

   $\widetilde{x}_k^m = x_{k+mD}w(kT_a)$, k = 0,1, … , N–1.

   và gọi dãy mới là $\{ \widetilde{x}_k^m \}=\{ \widetilde{x}_0^m, \widetilde{x}_1^m, \dots, \widetilde{x}_{N-1}^m \}$.

   c) Sử dụng **`fft()`** để tính $\{ \widetilde{X}_a^m(jn\Omega) \}$, n=0,1, … ,N–1 của dãy $\{ \widetilde{x}_k^m \}$.

   d) Tính $X_a^m(jn\Omega) = X_a^{m-1}(jn\Omega) + \widetilde{X}_a^{m-1}(jn\Omega)e^{-jmnD\frac{2\pi}{N}}$, n=0,1, … ,N–1.

   e) Cho ra kết quả $X(jn\Omega) = KT_aX_a^m(jn\Omega)$, n=0,1, … ,N–1 và đó được xem như kết quả truy hồi tại bước thứ m cho ảnh Fourier của $x(t)$.

Điều kiện chọn τ cho trong (5.11a) chính là tiêu chuẩn Shannon đã nói tới tại mục 2.1.5, định lý 2.4, nhưng ở đây là ứng với giá trị tần số giới hạn ω_τ của $w(t)$. Nói cách khác, khi áp dụng SFT ta phải *sử dụng tiêu chuẩn Shannon hai lần*: một lần cho thời gian trích mẫu tín hiệu T_a và một lần nữa cho bước trượt cửa sổ quan sát tín hiệu τ.

Thuật toán trên đã được cài đặt thành chương trình **sft()** viết trên C cho dưới đây để tham khảo. Chương trình **sft()** này có các biến hình thức:

a) Số nguyên **m** chứa chỉ số khoảng quan sát tín hiệu $[m\tau, m\tau+T)$.

b) Số nguyên **w** chứa chỉ số hàm cửa sổ $w_i(t)$, $0 \le i \le 7$ (mục 2.1.6). Nếu nội dung của **w** là một số ngoài khoảng [0,7], hàm **sft()** sẽ sử dụng hàm cửa sổ $w_0(t)$.

c) Số nguyên **D** chứa hệ số D của bước trượt τ, tức là $\tau = DT_a$.

d) Số nguyên **Nexp** chứa số mũ lũy thừa 2 của độ dài dãy tín hiệu $\{ x_k^m \}$ đo được, trong khoảng $[m\tau, m\tau+T)$, tức là độ dài dãy tín hiệu sẽ phải là $N=2^{\mathbf{Nexp}}$.

e) Số thực **Ta** chứa chu kỳ thời gian trích mẫu tín hiệu T_a.

f) Con trỏ **x** chỉ đầu mảng số phức **x[]** chứa các giá trị x_k^m, k=0,1, … , N–1 theo thứ tự **x[0]**$=x_{mD}$, **x[1]**$=x_{mD+1}$, … , **x[N-1]**$=x_{mD+N-1}$. Sau khi chương trình **sft()** được thực hiện xong, nội dung của mảng này sẽ được thay bởi các giá trị ảnh Fourier $\{ \widetilde{X}_a^m(jn\Omega) \}$, n=0,1, … , N–1 của dãy $\{ \widetilde{x}_k^m \}$, trong đó $\widetilde{x}_k^m = x_{k+mD} w_i(kT_a)$, k=0,1, … , N–1.

g) Con trỏ **X** chỉ đầu mảng số phức **X[]** chứa các giá trị đầu vào $X_a^{m-1}(jn\Omega)$. Sau khi chương trình **sft()** được thực hiện xong mảng này sẽ chứa dãy kết quả $X_a^m(jn\Omega)$, n=0,1, … , N–1.

```
void sft(int m,int w,int D,int Nexp,double Ta,complex *x,
         complex *X)
{  int i,k,p,N=pow(2,Nexp);
   double s,T=Ta*N;
   complex e;
   k=N/2;
   for (i=0;i<=k;i++)
   {  s=fw(w,Ta*(double)(i-k),T);
      x[i]=x[i]*s;
      if((i>0)&&(i!=(p=(N-i)))) x[p]=x[p]*s;
   }
   fft(Nexp,x);
   for(i=0;i<N;i++)
      X[i]=X[i]+x[i]*exp(complex(0.,-M_PI*2*D*m*i/N));
}
```

5.1.4 Ứng dụng để nhận dạng mô hình có tham số thay đổi

Các thuật toán nhận dạng mô hình đối tượng được trình bày trong quyển sách này phần lớn đều dựa vào việc phân tích phổ tín hiệu. Chẳng hạn toàn bộ các thuật toán nói

tới trong chương 2 về việc nhận dạng mô hình không tham số để đi tới công thức (2.52) nhận dạng giá trị đường đặc tính tần $\{G(jn\Omega_\lambda)\}$, n=0,1, … , M

$$G(jn\Omega_\lambda) = \frac{\widetilde{S}_{uy}(jn\Omega_\lambda)}{\widetilde{S}_u(n\Omega_\lambda)},$$

hoàn toàn được xây dựng trên cơ sở nhận dạng hàm mật độ phổ $S_u(j\omega)$, $S_{uy}(j\omega)$ từ các giá trị tín hiệu quan sát được. Cũng như vậy, những thuật toán nhận dạng tham số mô hình nói tới ở mục 3.2 thuộc chương 3 dựa chủ yếu vào mô hình không tham số đã có, tức là dựa vào dãy $\{G(jn\Omega_\lambda)\}$, n=0,1, … , M. Bởi vậy nếu như người ta nhận dạng truy hồi được các giá trị $G(jn\Omega_\lambda)$ sao cho sau mỗi khoảng quan sát $[m\tau, m\tau+T)$, m=0, 1, 2, … chúng lại được hiệu chỉnh cho phù hợp và chính xác hơn thì cùng với nó các thuật toán trên sẽ trở thành những thuật toán nhận dạng mô hình đối tượng theo nguyên tắc truy hồi và sau từng khoảng quan sát $[m\tau, m\tau+T)$, m=0, 1, 2, … mô hình đối tượng hiện có lại được hiệu chỉnh để có được một mô hình mới chính xác hơn. Những thuật toán nhận dạng truy hồi mô hình đối tượng như vậy có rất nhiều ý nghĩa ứng dụng cho những đối tượng có tham số thay đổi.

Vấn đề còn lại là làm sao chuyển được thuật toán nhận dạng hàm mật độ phổ $S_u(j\omega)$, $S_{uy}(j\omega)$ đã có ở chương 2 (hàm **`nonpar()`**) sang dạng truy hồi?. Câu trả lời là dựa vào kỹ thuật SFT với hàm **`sft()`** đã được cài đặt trên C tại mục 5.1.3.

Trước hết ta chứng minh định lý sau:

Định lý 5.3: Hàm mật độ phổ định nghĩa theo công thức Wiener–Chitchin dạng rời rạc

$$\widetilde{S}_{uy}(jn\Omega_\lambda) = T_a \sum_{m=-N+1}^{N-1} \widetilde{r}_{uy}(mT_a) e^{-jnm\frac{2\pi}{\lambda}} \tag{5.13a}$$

với λ là số nguyên lũy thừa 2 nhỏ nhất nhưng không nhỏ hơn $2N$–1 và

$$\widetilde{r}_{uy}(mT_a) \approx \frac{1}{p} \sum_{k=0}^{N-|m|-1} \widetilde{u}_k \widetilde{y}_{k+m}, \qquad p = \begin{cases} N & \text{nếu nhận dạng bias} \\ N-m & \text{nếu nhận dạng unbias} \end{cases} \tag{5.13b}$$

$$\widetilde{u}_k = \begin{cases} u_k & \text{nếu } 0 \le k \le N-1 \\ 0 & \text{nếu } N \le k \le \lambda - 1 \end{cases}$$

$$\widetilde{y}_k = \begin{cases} y_k & \text{nếu } 0 \le k \le N-1 \\ 0 & \text{nếu } N \le k \le \lambda - 1 \end{cases}$$

trong đó u_k , y_k , k=0,1, … ,N–1 là những giá trị đo được của tín hiệu $u(t)$, $y(t)$ trong khoảng thời gian [0, T) với chu kỳ lấy mẫu T_a ($T = NT_a$), $\Omega_\lambda = \frac{2\pi}{\lambda T_a}$, λ là số

nguyên lũy thừa 2 nhỏ nhất nhưng không nhỏ hơn $2N-1$, có thể được tính theo công thức *Periodogram* như sau

$$\widetilde{S}_{uy}(jn\Omega_\lambda)=\frac{T_a}{p}\overline{\widetilde{U}_a(jn\Omega_\lambda)}\widetilde{Y}_a(jn\Omega_\lambda)\,,\ \ n=0,1,\ \ldots\ ,\lambda-1 \tag{5.14}$$

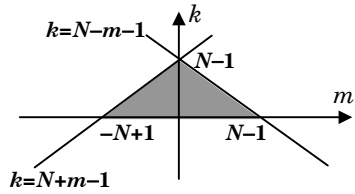
với $\widetilde{U}_a(jn\Omega_\lambda)$ là ảnh Fourier của dãy $\{\,\widetilde{u}_k\,\}$ và $\widetilde{Y}_a(jn\Omega_\lambda)$ là ảnh Fourier của dãy $\{\,\widetilde{y}_k\,\}$, $k=0,1,\ \ldots\ ,N-1$ thu được nhờ **`fft()`**.

Chứng minh: Thay (5.13b) vào (5.13a) có

$$\widetilde{S}_{uy}(jn\Omega_\lambda)=\frac{T_a}{p}\sum_{m=-N+1}^{N-1}\ \sum_{k=0}^{N-|m|-1}\widetilde{u}_k\widetilde{y}_{k+m}e^{-jnm\frac{2\pi}{\lambda}} \tag{5.15}$$

tiếp theo đổi vị trí hai dấu tổng với sự trợ giúp của hình 5.3 biểu diễn miền lấy tổng được

$$\widetilde{S}_{uy}(jn\Omega_\lambda)=\frac{T_a}{p}\sum_{k=0}^{N-1}\left(\sum_{m=k-N+1}^{N-1-k}\widetilde{u}_k\widetilde{y}_{k+m}e^{-jnm\frac{2\pi}{\lambda}}\right)$$



Hình 5.3: Miền lấy tổng của công thức xác định hàm mật độ phổ (5.15).

Suy ra

$$\widetilde{S}_{uy}(jn\Omega_\lambda)\ =\frac{T_a}{p}\sum_{k=0}^{N-1}\widetilde{u}_k e^{jnk\frac{2\pi}{\lambda}}\left(\sum_{m=k-N+1}^{N-1-k}\widetilde{y}_{k+m}e^{-jn(k+m)\frac{2\pi}{\lambda}}\right)$$

$$=\frac{T_a}{p}\sum_{k=0}^{N-1}\widetilde{u}_k e^{jnk\frac{2\pi}{\lambda}}\left(\sum_{l=2k-N+1}^{N-1}\widetilde{y}_l e^{-jnl\frac{2\pi}{\lambda}}\right),\qquad l=k+m$$

Nhưng do $\widetilde{u}_k=\widetilde{y}_l=0$ khi $k,l\notin[0,N)$ nên

$$\widetilde{S}_{uy}(jn\Omega_\lambda)\ =\frac{T_a}{p}\sum_{k=0}^{N-1}\widetilde{u}_k e^{jnk\frac{2\pi}{\lambda}}\left(\sum_{l=0}^{N-1}\widetilde{y}_l e^{-jnl\frac{2\pi}{\lambda}}\right)$$

$$= \frac{T_a}{p} \sum_{k=0}^{2N-2} \tilde{u}_k e^{jnk\frac{2\pi}{\lambda}} \left(\sum_{l=0}^{2N-2} \tilde{y}_l e^{-jnl\frac{2\pi}{\lambda}} \right) = \frac{T_a}{p} \overline{\tilde{U}_a(jn\Omega_\lambda)} \tilde{Y}_a(jn\Omega_\lambda)$$

và đó chính là điều phải chứng minh. ❑

Dựa vào định lý 5.3 và công thức $G(jn\Omega\lambda) = \dfrac{\tilde{S}_{uy}(jn\Omega_\lambda)}{\tilde{S}_u(n\Omega_\lambda)}$ ta thấy việc nhận dạng đường đặc tính tần $G(j\omega)$ có thể được thực hiện truy hồi thông qua kỹ thuật SFT áp dụng cho cả hai tín hiệu $u(t)$, $y(t)$ như sau:

1) Chọn hàm cửa sổ $w(t)$, $t \in [0, T)$, bước trượt τ theo (5.11a), hằng số K theo (5.11b).

2) Đặt giá trị khởi phát $\tilde{U}_a^{-1}(jn\Omega_\lambda) = \tilde{Y}_a^{-1}(jn\Omega_\lambda) = 0$ cho tất cả các chỉ số n=0,1, … , λ–1, trong đó $\Omega_\lambda = \dfrac{2\pi}{\lambda T_a}$, λ là số nguyên lũy thừa 2 nhỏ nhất nhưng không nhỏ hơn 2N–1 và $N = \dfrac{T}{T_a}$ với T_a là chu kỳ trích mẫu.

3) Thực hiện lần lượt các bước sau cho từng khoảng quan sát tín hiệu $[m\tau,\ m\tau+T)$, m=0, 1, 2, …

   a) Đo tín hiệu $u(t)$, $y(t)$ trong khoảng thời gian $[m\tau, m\tau+T)$ và gọi dãy gồm N các giá trị nhận được là

   $\{ u_{mD}, u_{mD+1}, \dots, u_{mD+N-1} \}$.

   $\{ y_{mD}, y_{mD+1}, \dots, y_{mD+N-1} \}$.

   b) Nhân từng phần tử của hai dãy trên với hàm cửa sổ $w(t)$ sau đó gắn thêm các giá trị 0 sao cho mỗi dãy có đúng λ phần tử:

   $\{ \tilde{u}_k^m \} = \{ u_{mD}w(0), u_{mD+1}w(T_a), \dots, u_{mD+N-1}w((N-1)T_a), 0, \dots, 0 \}$

   $\{ \tilde{y}_k^m \} = \{ y_{mD}w(0), y_{mD+1}w(T_a), \dots, y_{mD+N-1}w((N-1)T_a), 0, \dots, 0 \}$

   k=0,1, … ,λ–1.

   c) Sử dụng `fft()` để tính $\{ \tilde{U}_a^m(jn\Omega_\lambda) \}$ của dãy $\{ \tilde{u}_k^m \}$ cũng như $\{ \tilde{Y}_a^m(jn\Omega_\lambda) \}$ của dãy $\{ \tilde{y}_k^m \}$.

   d) Tính

   $$U_a^m(jn\Omega_\lambda) = U_a^{m-1}(jn\Omega_\lambda) + \tilde{U}_a^{m-1}(jn\Omega_\lambda) e^{-jmnD\frac{2\pi}{\lambda}}$$

   $$Y_a^m(jn\Omega_\lambda) = Y_a^{m-1}(jn\Omega_\lambda) + \tilde{Y}_a^{m-1}(jn\Omega_\lambda) e^{-jmnD\frac{2\pi}{\lambda}}$$

e) Tính

$$G(jn\Omega_\lambda) = \frac{\widetilde{S}_{uy}(jn\Omega_\lambda)}{\widetilde{S}_u(n\Omega_\lambda)} = \frac{Y_a^m(jn\Omega_\lambda)}{U_a^m(jn\Omega_\lambda)},\ n = 0,1,\ \dots\ ,\ \lambda-1$$

và đó là giá trị hàm truyền đạt đã được hiệu chỉnh tại khoảng quan sát thứ m .

Một phần thuật toán trên đã được cài đặt thành hàm **renonpar()** viết trên C cho dưới đây để tham khảo. Hàm **renonpar()** có các biến hình thức sau:

a) Số nguyên **m** chứa chỉ số khoảng quan sát tín hiệu $[m\tau, m\tau+T)$.

b) Số nguyên **w** chứa chỉ số hàm cửa sổ $w_i(t)$, $0 \le i \le 7$ (mục 2.1.6) dùng để trượt khoảng quan sát tín hiệu. Nếu nội dung của **w** là một số ngoài khoảng [0,7], hàm **renonpar()** sẽ sử dụng hàm cửa sổ $w_0(t)$.

c) Số nguyên **D** chứa hệ số D của bước trượt τ, tức là $\tau = DT_a$.

d) Số nguyên **Nexp** chứa số mũ lũy thừa 2 của độ dài dãy tín hiệu $\{u_{mD+k}\}$, $\{y_{mD+k}\}$ đo được trong khoảng thời gian quan sát tín hiệu $[m\tau, m\tau+T)$ là N.

e) Con trỏ **u** chỉ đầu mảng thực **u[]** chứa các giá trị u_{mD+k} , k=0,1, … , N–1 theo thứ tự **u[0]**= u_{mD}, **u[1]**= u_{mD+1}, … , **u[N-1]**= u_{mD+N-1}.

f) Con trỏ **y** chỉ đầu mảng thực **y[]** chứa các giá trị y_{mD+k} , k=0,1, … , N–1 theo thứ tự **y[0]**= y_{mD}, **y[1]**= y_{mD+1}, … , **y[N-1]**= y_{mD+N-1}.

g) Con trỏ **U** chỉ đầu mảng số phức **U[]** chứa các giá trị đầu vào $\{ U_a^{m-1}(jn\Omega_\lambda) \}$. Sau khi chương trình **renonpar()** thực hiện xong mảng này sẽ chứa $\{ U_a^m(jn\Omega_\lambda) \}$, n=0,1, … ,λ–1.

h) Con trỏ **Y** chỉ đầu mảng số phức **Y[]** chứa các giá trị đầu vào $\{ U_a^{m-1}(jn\Omega_\lambda) \}$. Sau khi chương trình **renonpar()** thực hiện xong mảng này sẽ chứa $\{ Y_a^m(jn\Omega_\lambda) \}$, n=0,1, … ,N–1.

i) Số thực **Ta** chứa khoảng thời gian trích mẫu tín hiệu T_a.

j) Con trỏ **G** chỉ đầu mảng số phức **G[]** chứa kết quả $\{G(jn\Omega_\lambda)\}$, n=0,1, … , N–1.

```
void renonpar(int m,int w,int D,int Nexp,double *u,double *y,
              complex *U,complex *Y,double Ta,complex *G)
{
   int k,N=pow(2,Nexp);
   complex *x;
   x=new complex[N];
   for(k=0;k<N;k++) x[k]=complex(u[k]);
   sft(m,w,D,Nexp,Ta,x,U);
```

```
    for(k=0;k<N;k++) x[k]=complex(y[k]);
    sft(m,w,D,Nexp,Ta,x,Y);
    for(k=0;k<N;k++) G[k]=Y[k]/U[k];
    delete [] x;
}
```

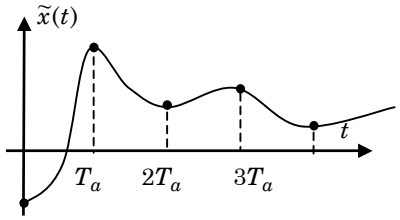
5.2 Nội suy

Nhớ lại thuật toán nhận dạng $G(jn\Omega_\lambda)$, n=0,1, … ,2M đã trình bày trong mục 2.3.1 mà ở đó $G(jn\Omega_\lambda)$ được tính gián tiếp thông qua $S_u(n\Omega_\lambda)$, $S_{uy}(jn\Omega_\lambda)$, n=0,1, … ,λ–1 theo công thức (2.53), ta thấy thuật toán chỉ có thể cho ra được các giá trị gần đúng của hàm đặc tính tần $G(j\omega)$ trong khoảng tần số từ 0 đến

$$(2M+1)\Omega_\lambda = \frac{2(2M+1)\pi}{\lambda T_a}$$

Làm thế nào để mở rộng khoảng tần số này. Có thể thấy ngay rằng chỉ có một giải pháp là làm nhỏ T_a, tức là tăng tần số trích mẫu tín hiệu.

Nhưng trong thực tế, do hạn chế của kỹ thuật hoặc do tốc độ xử lý của thiết bị không cho phép, ta không thể chọn được T_a đủ nhỏ để có thể thu được một kết quả đo có độ phân giải như ý muốn. Ở những trường hợp như vậy người ta thường áp dụng phương pháp nội suy tín hiệu. Điều này có nghĩa là sau khi đo tín hiệu vào hoặc ra được gọi chung là $x(t)$ và có được dãy N giá trị $x(kT_a)$, k=0,1, … , N–1 của nó, ta phải xác định một hàm liên tục $\tilde{x}(t)$ sao cho hàm đó đi qua tất cả N các điểm mẫu $x(kT_a)$ đó (hình 5.4). Khi đã có hàm liên tục $\tilde{x}(t)$ ta có thể lấy lại giá trị tín hiệu $\tilde{x}(kT_b)$, k=0,1, … , $\tilde{N}$ –1 với một chu kỳ lấy mẫu mới T_b nhỏ tùy ý.



Hình 5.4 : Mô tả phương pháp nội suy.

5.2.1 Nội suy cổ điển

Với phương pháp nội suy cổ điển, người ta chọn trước một đa thức bậc N–1

$$\tilde{x}(t) = a_1 + a_2 t + \ \dots \ + a_N t^{N-1},$$

trong đó N là số các giá trị mẫu đã có. Nhiệm vụ của nội suy là xác định các hệ số $a_1, a_2, \ldots, a_N$ của đa thức này, sao cho đường cong của *đa thức mô hình* $\tilde{x}(t)$ đi qua các giá trị mẫu $x(0), \ldots, x((N-1)T_a)$ của hàm cần nội suy, tức là phải thỏa mãn điều kiện

$$\tilde{x}(kT_a) = x_k = x(kT_a) \text{ với } \quad k = 0, 1, \ldots, N-1. \tag{5.15}$$

Viết lại (5.15) lần lượt cho tất cả các giá trị k dưới dạng ma trận sẽ có

$$\underbrace{\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 1 & T_a & \cdots & T_a^{N-1} \\ \vdots & & & \\ 1 & (N-1)T_a & \cdots & \left[(N-1)T_a\right]^{N-1} \end{pmatrix}}_{A} \cdot \underbrace{\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{pmatrix}}_{\underline{a}} = \underbrace{\begin{pmatrix} x_0 \\ x_2 \\ \vdots \\ x_{N-1} \end{pmatrix}}_{\underline{g}}.$$

Suy ra

$$\underline{a} = A^{-1}\,\underline{g} \tag{5.16}$$

và đó chính là công thức xác định vector tham số $\underline{a}$ cho đa thức $\tilde{x}(t)$ từ các giá trị đo được x_k , $k = 0, 1, \ldots, N-1$.

5.2.2 Nội suy spline

Phương pháp nội suy cổ điển thường chỉ thích hợp với những bài toán nội suy bậc thấp, tức là số các giá trị đã có của hàm cần nội suy nhỏ. Nó có ưu điểm là sai số giữa mô hình nội suy được $\tilde{x}(t)$ và tín hiệu gốc $x(t)$ tại các điểm lấy mẫu bằng 0. Khi số lượng các điểm mẫu lớn, bậc của đa thức nội suy $\tilde{x}(t)$ cũng phải tăng theo, làm cho hàm $\tilde{x}(t)$ càng có nhiều điểm cực trị và đồ thị của nó càng có nhiều "lượn sóng" do đa thức bậc càng cao thì càng có nhiều điểm cực trị. Điều này dẫn đến nhược điểm chính hạn chế ứng dụng của phương pháp là người ta hoàn toàn không có khả năng kiểm soát và khống chế được sai số giữa các điểm lấy mẫu và không khẳng định được rằng đa thức mô hình đã nội suy được có hội tụ đến tín hiệu gốc hay không. Thậm chí, người ta còn chỉ ra được rằng với bất cứ một tập các điểm mẫu nào cho trước, bao giờ cũng tồn tại một hàm số liên tục nhận các điểm mẫu đó làm giá trị mà bài toán nội suy cổ điển áp dụng cho hàm số đó lại phân kỳ.

Để khắc phục nhược điểm trên của bài toán nội suy cổ điển, người ta sẽ không nội suy với toàn bộ tập hợp các giá trị mẫu nữa mà chỉ nội suy từng khoảng *cục bộ*, để có thể làm giảm bậc của đa thức mô hình. Chẳng hạn người ta có thể chọn những *đa thức mô hình cục bộ* bậc 2 là

$$f_0(t) = a_{00} + a_{01}t + a_{02}t^2 ,$$

$$f_1(t) = a_{10} + a_{11}t + a_{12}t^2 \ ,$$

$$\vdots$$

cho từng khoảng 3 giá trị mẫu $\{0, T_a, 2T_a\}$, $\{T_a, 2T_a, 3T_a\}$, … của tín hiệu và xác định hệ số của các đa thức này sao cho

$$f_0(kT_a) = x_k \quad \text{với} \quad k = 0, 1, 2$$

$$f_1(kT_a) = x_k \quad \text{với} \quad k = 1, 2, 3$$

$$\vdots$$

thay cho một đa thức chung bậc N–1 như phương pháp cổ điển vẫn làm. Sau đó những đa thức nội suy cục bộ này sẽ lại được "dán" với nhau tại để tạo thành một đường cong chung $\tilde{x}(t)$ đi qua tất cả các điểm mẫu và có đạo hàm (bậc đạo hàm càng cao càng tốt) tại những điểm nối, tức là tạo ra một đường cong trơn khắp nơi chứ không chỉ riêng trong khoảng cục bộ được nội suy. Đường cong $\tilde{x}(t)$ là đường nội suy của tín hiệu đã cho và phương pháp nội suy như vậy gọi là *nội suy spline*.

Bậc của đa thức được chọn sẽ quyết định kích thước miền cục bộ được nội suy, chẳng hạn nếu chọn đa thức bậc hai thì do có 3 tham số phải xác định, miền nội suy cục bộ sẽ chỉ chứa 3 điểm mẫu. Đa thức được chọn để nội suy cục bộ đó có tên là *hàm mô hình cục bộ*.

5.2.3 Nội suy B–spline

Một trong những phương pháp nội suy spline có ý nghĩa ứng dụng trong phân tích phổ tín hiệu và có họ hàng gần gũi với mô hình toán học của *khối hồi phục tín hiệu* (holding) trong điều khiển kỹ thuật là phương pháp *nội suy B–spline*.

Phương pháp nội suy B-spline, cũng giống như phương pháp spline nói chung, được xây dựng dựa trên các hàm mô hình cục bộ mà ta sẽ gọi là hàm *B–spline gốc*. Ký hiệu $f_m(t)$ là hàm B–spline gốc bậc m thì $f_m(t)$ được xác định theo công thức truy hồi (định nghĩa của Bezier):

a) $$f_0(t) = \frac{1(t) - 1(t - T_a)}{T_a} \tag{5.17a}$$

b) $$f_m(t) = f_0(t) * f_{m-1}(t). \tag{5.17b}$$

Định lý 5.3: Hàm $f_m(t)$ có khoảng thời gian sống supp $f_m(t) = [0, (m+1)T_a]$, tức là

$$f_m(t) \equiv 0 \quad \text{khi} \quad t < 0 \quad \text{hoặc} \quad t > (m+1)T_a.$$

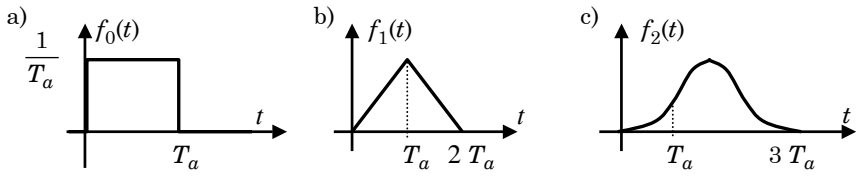
Chứng minh:

Ta chứng minh bằng phương pháp quy nạp. Từ định nghĩa ta thấy điều khẳng định đúng cho m=0. Giả sử cũng đúng cho m–1. Vậy từ công thức định nghĩa tích chập

$$f_m(t) = f_0(t) * f_{m-1}(t) = \int_{-\infty}^{\infty} f_0(\tau) f_{m-1}(t-\tau) d\tau = \int_0^{T_a} f_0(\tau) f_{m-1}(t-\tau) d\tau$$

hàm $f_m(t)$ chỉ tồn tại với những giá trị t thỏa mãn $0<t-\tau<mT_a$, trong đó $0<\tau<T_a$, nói cách khác:

$$0 \le t \le (m+1)T_a$$ ❑



Hình 5.5: a) Hàm B–spline gốc $f_0(t)$
b) Hàm B–spline gốc $f_1(t)$
c) Hàm B–spline gốc $f_2(t)$

Định lý 5.4: Một hàm B–spline gốc $f_m(t)$ bậc m có mô hình

$$f_m(t) = \frac{m+1}{T_a^{m+1}} \sum_{k=0}^{m+1} (-1)^k \frac{(t-kT_a)^m 1(t-kT_a)}{k!(m+1-k)!} \tag{5.18}$$

trong đó ký hiệu $1(t)$ chỉ hàm Heaviside.

Chứng minh:

Ta chứng minh (5.18) theo phương pháp quy nạp. Hiển nhiên (5.18) đúng cho trường hợp m=0. Giả sử (5.18) đã đúng cho m–1. Ta phải chỉ rằng nó cũng đúng với m. Bắt đầu từ vế phải với hình 5.5a) minh họa cho $f_0(t)$ sẽ có

$$f_0(t) * f_{m-1}(t) = \int_{-\infty}^{\infty} f_0(\tau) f_{m-1}(t-\tau) d\tau = -\frac{1}{T_a} \int_0^{T_a} f_{m-1}(t-\tau) d\tau$$

Thay (5.18) cho trường hợp m–1 vào công thức trên

$$f_0(t) * f_{m-1}(t) = \frac{1}{T_a^{m+1}} \int_0^{T_a} m \sum_{k=0}^{m} (-1)^k \frac{(t-kT_a-\tau)^{m-1} 1(t-kT_a-\tau)}{k!(m-k)!} d\tau$$

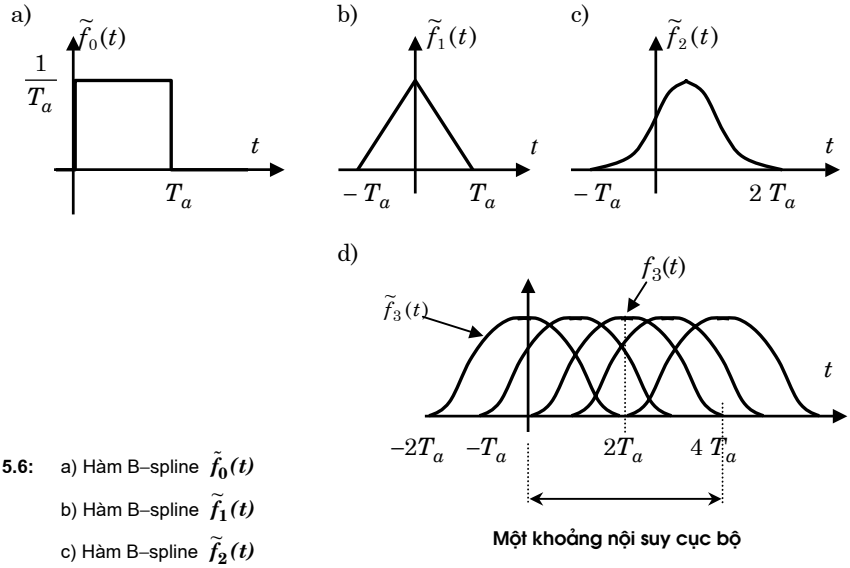
$$= \frac{1}{T_a^{m+1}} \sum_{k=0}^{m} \frac{(-1)^k}{k!(m-k)!} \left[(t-kT_a)^m 1(t-kT_a) - (t-(k+1)T_a)^m 1(t-(k+1)T_a) \right]$$

sau đó tách thành hai tổng và trong tổng thứ hai thay k+1=l thì được

$$f_0(t)* f_{m-1}(t) = \frac{1}{T_a^{m+1}}\left[\sum_{k=0}^{m}\frac{(-1)^k}{k!(m-k)!}(t-kT_a)^m 1(t-kT_a) + \right.$$
$$\left. + \sum_{l=1}^{m+1}\frac{(-1)^l}{(l-1)!(m+1-l)!}(t-lT_a)^m 1(t-lT_a)\right]$$
$$= \frac{1}{T_a^{m+1}}\left[\frac{(-1)^m}{m!}t^m 1(t) + \sum_{k=1}^{m}\frac{(-1)^k(t-kT_a)^m 1(t-kT_a)}{(k-1)!(m-k)!}\left(\frac{1}{k}+\frac{1}{m+1-k}\right)+\right.$$
$$\left. + \frac{(-1)^{m+1}}{m!}(t-(m+1)T_a)^m 1(t-(m+1)T_a)\right]$$
$$= \frac{m+1}{T_a^{m+1}}\sum_{k=0}^{m+1}(-1)^k\frac{(t-kT_a)^m 1(t-kT_a)}{k!(m+1-k)!}$$
$$= f_m(t)$$

và đó chính là điều phải chứng minh. ❑

Định lý 5.4 còn chỉ rằng $f_m(t)$ nhận giá trị cực đại tại $\frac{m+1}{2}T_a$.



Hình 5.6: a) Hàm B–spline $\tilde{f}_0(t)$
b) Hàm B–spline $\tilde{f}_1(t)$
c) Hàm B–spline $\tilde{f}_2(t)$
d) Nội suy với hàm B–spline gốc bậc 3

Quay lại vấn đề chính là nội suy dãy hàm trọng lượng từ các giá trị mẫu x_k , k=1, 2, … , N. Gọi $\tilde{x}(t)$ là hàm nội suy theo phương pháp B–spline với hàm B–spline gốc $f_m(t)$ bậc m thu được bằng cách "dán" các hàm B–spline gốc này lại với nhau sao cho $\tilde{x}(t)$ đi qua các điểm mẫu x_k .

Để trong mỗi khoảng nội suy cục bộ có được nhiều hàm $f_m(t)$ tham gia, ta sẽ dịch $f_m(t)$ sang trái một khoảng $\tau = \left\lceil \frac{m}{2} \right\rceil T_a$, trong đó ký hiệu $\lceil x \rceil$ chỉ một số nguyên nhỏ nhất nhưng không nhỏ hơn x, sao cho hàm nhận được (hình 5.6)

$$\tilde{f}_m(t) = f_m(t+\tau)$$

có khoảng thời gian sống (gần) đối xứng qua 0 là $\left(-\left\lceil \frac{m}{2} \right\rceil T_a, \left\lceil \frac{m+1}{2} \right\rceil T_a \right)$. Vậy thì khi dán các hàm $\tilde{f}_m(t)$ này lại với nhau để có $\tilde{x}(t)$ ta được

$$\tilde{x}(t) = \sum_{n=0}^{N-1} a_n \tilde{f}_m(t - nT_a) , \tag{5.19}$$

với a_n , n=1, 2, … , N–1 là những tham số được xác định sao cho

$$\tilde{x}(kT_a) = x_k , \quad k=0, 1, \dots , N-1. \tag{5.20}$$

Hàm $\tilde{f}_m(t)$ có tên gọi là hàm B–spline để phân biệt với hàm B–spline gốc $f_m(t)$.

Thay (5.19) vào (5.20) rồi viết chung N phương trình (k=0, … , N–1) lại với nhau dưới dạng ma trận, ta sẽ có

$$\underbrace{\begin{pmatrix} \tilde{f}_m(0) & \cdots & \tilde{f}_m((-N+1)T_a) \\ \vdots & \ddots & \vdots \\ \tilde{f}_m((N-1)T_a) & \cdots & \tilde{f}_m(0) \end{pmatrix}}_{A} \cdot \underbrace{\begin{pmatrix} a_0 \\ \vdots \\ a_{N-1} \end{pmatrix}}_{\underline{a}} = \underbrace{\begin{pmatrix} x_0 \\ \vdots \\ x_{N-1} \end{pmatrix}}_{\underline{g}}$$

$$\Rightarrow \qquad \underline{a} = A^{-1} \cdot \underline{g} \tag{5.21}$$

và đó chính là công thức xác định vector tham số $\underline{a}$ cho hàm nội suy $\tilde{x}(t)$.

Ví dụ 1: Xác định vector tham số $\underline{a}$ cho $\tilde{x}(t)$ được nội suy theo phương pháp B–spline bậc 1 từ các giá trị mẫu x_k , k=0, 1, … , N–1.

Từ định nghĩa $\tilde{f}_0(t)$ xác định theo $f_0(t)$ ta có $\tilde{f}_0(t) = f_0(t)$. Bởi vậy

$$\tilde{f}_0(0) = \frac{1}{T_a} \quad \text{và} \quad \tilde{f}_0(kT_a) \equiv 0 \quad \text{khi} \quad k \neq 0 .$$

Suy ra

$$a_n = T_a x_n\,,\ \ n=0,\ 1,\ \dots\ ,\ N-1.$$ ❑

Ví dụ 2: Xác định vector tham số $\underline{a}$ cho $\tilde{x}(t)$ được nội suy theo phương pháp B–spline bậc 3 từ các giá trị mẫu x_k , k=0, 1, … , N–1.

Theo công thức (5.18) của định lý 5.4, hàm B–spline gốc bậc 3 là $f_3(t)$ có dạng

$$f_3(t) \ = \frac{4}{T_a^4}\sum_{k=0}^{4}(-1)^k\,\frac{(t-kT_a)^3 1(t-kT_a)}{k!(4-k)!}$$

do đó

$$f_3(T_a)=\frac{1}{6T_a}\,,\qquad f_3(2T_a)=\frac{4}{6T_a}\,,\qquad f_3(3T_a)=\frac{1}{6T_a}\,,$$

$f_3(kT_a)\equiv 0$ khi $k\geq 4$ hoặc $k\leq 0$.

Với $\tau=\left\lceil\frac{3}{2}\right\rceil=2$ suy ra được

$$\tilde{f}_3(-T_a)=\frac{1}{6T_a}\,,\qquad \tilde{f}_3(0)=\frac{4}{6T_a}\,,\quad \tilde{f}_3(T_a)=\frac{1}{6T_a}\,,\qquad \tilde{f}_3(kT_a)\equiv 0\qquad \text{khi } |k|>1.$$

Vậy vector tham số $\underline{a}$ cần tìm sẽ là nghiệm của

$$\frac{1}{6T_a}\begin{pmatrix} 4 & 1 & 0 & \cdots & 0 & 0\\ 1 & 4 & 1 & \cdots & 0 & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots\\ 0 & 0 & 0 & \cdots & 4 & 1\\ 0 & 0 & 0 & \cdots & 1 & 4 \end{pmatrix}\cdot\begin{pmatrix} a_0\\ \vdots\\ a_{N-1}\end{pmatrix}=\begin{pmatrix} x_0\\ \vdots\\ x_{N-1}\end{pmatrix}.$$ ❑

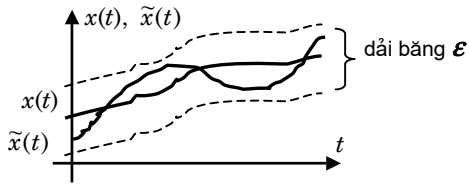
5.2.4 Sai số phổ của nội suy B–spline

Trong phần này ta sẽ nghiên cứu sai số phổ của nội suy B–spline. Phép nội suy này thực chất chính là quá trình biến đổi một tín hiệu số thành tín hiệu tương tự (khối D/A) nhờ các khâu holding (hình 5.6a).

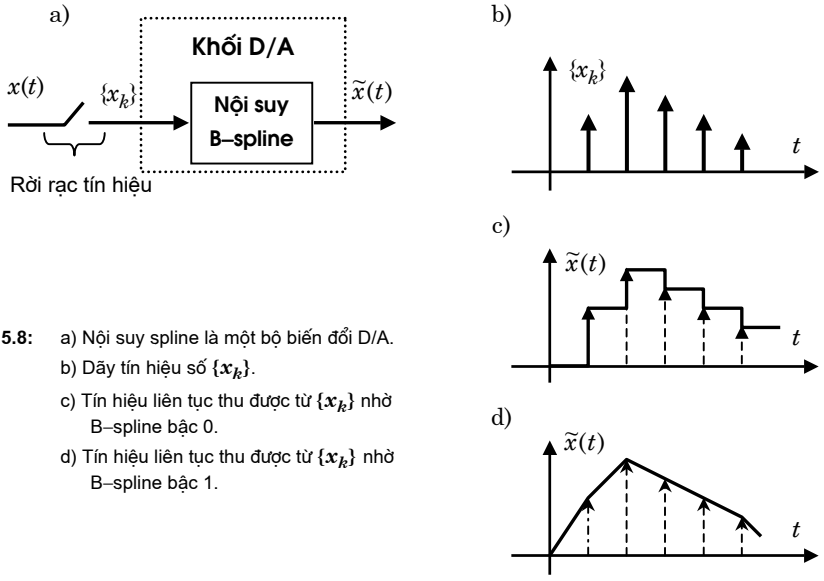
Tại sao lại phải đặt ra vấn đề nghiên cứu sai số phổ của pháp nội suy?. Trong ngành toán số nói chung, khi không có cách nào có thể xác định được một cách chính xác hàm $x(t)$ có khoảng thời gian sống hữu hạn và phải áp dụng những phương pháp gần đúng, người ta thu được $\tilde{x}(t)$ và nó được xem như một kết quả xấp xỉ của $x(t)$. Người ta sẽ rất thỏa mãn nếu như sai lệch giữa $\tilde{x}(t)$ và $x(t)$ trong khoảng thời gian được nội suy là không đáng kể, nói cách khác nếu $\tilde{x}(t)$ nằm trong một "dải băng" ε nào đó của hàm cần xác định $x(t)$ (hình 5.7)

$$\sup_t \left| \tilde{x}(t) - x(t) \right| < \varepsilon \tag{5.22}$$

trong đó ε là một số thực dương đủ nhỏ.



Hình 5.7: Yêu cầu về sai số nội suy.



Hình 5.8: a) Nội suy spline là một bộ biến đổi D/A.
b) Dãy tín hiệu số $\{x_k\}$.
c) Tín hiệu liên tục thu được từ $\{x_k\}$ nhờ B–spline bậc 0.
d) Tín hiệu liên tục thu được từ $\{x_k\}$ nhờ B–spline bậc 1.

Đối với việc xử lý tín hiệu và nhận dạng thì vấn đề đặt ra có hơi khác một chút. Bên cạnh điều kiện (5.22) người ta còn phải quan tâm xem tín hiệu xấp xỉ $\tilde{x}(t)$ có phản ánh thông tin tần số của $x(t)$ một cách gần đúng sao cho có thể chấp nhận được hay không?. Tức là có tồn tại hay không một số thực dương η cũng đủ nhỏ để được

$$\sup_\omega \left| \tilde{X}(j\omega) - X(j\omega) \right| < \eta \tag{5.23}.$$

Đây là một vấn đề khá lý thú. Đã không ít người đã ngộ nhận rằng ở bài toán nội suy spline hai điều kiện (5.22), (5.23) là tương đương. Sau đây, chỉ riêng trong bài toán nội suy B–spline, ta sẽ chỉ rằng sự ngộ nhận đó hoàn toàn sai. Cụ thể là khi áp dụng nội suy B–spline bậc 2 hoặc 4 thì tồn tại những điểm tần số ω mà tại đó sự sai lệch giữa $\widetilde{X}(j\omega)$ và $X(j\omega)$ là vô cùng lớn [12].

Trước hết ta xem quá trình biến đổi x_0, x_1, … , x_{N-1} thành $\tilde{x}(t)$ bằng kỹ thuật B–spline được mô tả ở hình 5.8a như một khối D/A có tín hiệu đầu vào là hàm mở rộng $x_a(t)=\{x_k\}$ và tín hiệu ra là $\tilde{x}(t)$.

Ký hiệu quá trình biến đổi $x_a(t) \mapsto \tilde{x}(t)$ đó là ánh xạ T, thì do

$$T[x_a(t)] = \tilde{x}(t) = \sum_{n=0}^{N-1} a_n \hat{f}_m(t - nT_a)$$

và các hệ số a_0, a_1, … , a_{N-1} có quan hệ tuyến tính với đầu vào x_0, x_1, … , x_{N-1} (xem công thức (5.19)), nên T có tính chất tuyến tính, tức là

$$T\,[\alpha x_a(t)+\beta y_a(t)] = \alpha T\,[x_a(t)]+\beta T\,[y_a(t)]\;;\; \forall \alpha, \beta \in \mathbb{R}$$

trong đó $x_a(t)$, $y_a(t)$ là hai tín hiệu đầu vào khác nhau. Điều này chỉ rằng khối D/A theo kỹ thuật B–spline là một khối điều khiển tuyến tính.

Gọi $G(s)$ là hàm truyền đạt của khối D/A đó. Vậy thì

Định lý 5.5: Hàm đặc tính tần $G(j\omega)$ của khối D/A theo kỹ thuật B–spline bậc m có giá trị là

$$G(j\omega) = \frac{\widetilde{X}(j\omega)}{X_a(j\omega)} = \frac{\dfrac{d^m}{d\omega^m}\left(\dfrac{1}{\omega}\right)}{j\dfrac{d^m}{d\omega^m}\left(\dfrac{1}{1-e^{-j\omega T_a}}\right)} \tag{5.24}$$

Chứng minh:

Theo công thức nội suy (5.19) ta có

$$\widetilde{X}(j\omega) = \widetilde{F}_m(j\omega)\sum_{n=0}^{N-1} a_n e^{-j\omega n T_a} \tag{5.25}$$

Ngoài ra, do tín hiệu liên tục tại đầu đầu ra $\tilde{x}(t)$ cũng nhận các giá trị x_k , k= 0,1, ... , N–1 làm N giá trị mẫu nên theo định lý 2.3 và công thức (2.16) trong chương 2 ta đi đến

$$T_a X_a(j\omega) = \sum_{l=-\infty}^{\infty} \widetilde{X}\left[j(\omega - l\Omega_a)\right] = \sum_{l=-\infty}^{\infty}\left[\widetilde{F}_m(j(\omega - l\Omega_a))\sum_{n=0}^{N-1} a_n e^{-j\omega n T_a} e^{jl\Omega_a n T_a}\right]$$

Nhưng vì $\Omega_a T_a$=2π, nên $e^{jl\Omega_a nT_a} = e^{j2\pi nl} = 1$. Bởi vậy

$$T_a X_a(j\omega) = \sum_{l=-\infty}^{\infty} \widetilde{F}_m(j(\omega - l\Omega_a)) \sum_{n=0}^{N-1} a_n e^{-j\omega nT_a} \tag{5.26}$$

Chia hai vế của (5.25) và (5.26) cho nhau ta được

$$T_a G(j\omega) = \frac{\widetilde{F}_m(j\omega)}{\sum_{l=-\infty}^{\infty} \widetilde{F}_m(j(\omega - l\Omega_a))} \tag{5.27}$$

Mặt khác, từ định nghĩa về hàm B–spline gốc $f_m(t)$, ảnh Fourier $F_m(j\omega)$ của nó sẽ là

$$F_m(j\omega) = \left(\frac{1 - e^{-j\omega T_a}}{j\omega T_a} \right)^{m+1}$$

Do đó ảnh Fourier $\widetilde{F}_m(j\omega)$ của hàm B–spline $\widetilde{f}_m(t)$ được tính như sau

$$\widetilde{F}_m(j\omega) = F_m(j\omega) e^{j\omega\tau} = \left(\frac{1 - e^{-j\omega T_a}}{j\omega T_a} \right)^{m+1} e^{j\omega\tau} \text{ với } \tau = \left\lceil \frac{m}{2} \right\rceil T_a . \tag{5.28}$$

Lại để ý tiếp $e^{j2\pi k} = 1$ với mọi k nguyên nên

$$\widetilde{F}_m(j(\omega - l\Omega_a)) = \left(\frac{1 - e^{-j\omega T_a} e^{jl2\pi}}{j(\omega - l\Omega_a) T_a} \right)^{m+1} e^{j\omega\tau} e^{jl\left\lceil \frac{m}{2} \right\rceil 2\pi} = \left(\frac{1 - e^{-j\omega T_a}}{j(\omega - l\Omega_a) T_a} \right)^{m+1} e^{j\omega\tau} \tag{5.29}$$

Bởi vậy khi thay (5.28), (5.29) vào (5.26) có

$$G(j\omega) = \frac{\dfrac{1}{\omega^{m+1}}}{T_a \sum_{l=-\infty}^{\infty} \dfrac{1}{(\omega - l\Omega_a)^{m+1}}} = \frac{\dfrac{d^m}{d\omega^m}\left(\dfrac{1}{\omega}\right)}{\dfrac{d^m}{d\omega^m}\left(T_a \sum_{l=-\infty}^{\infty} \dfrac{1}{\omega - l\Omega_a} \right)} \tag{5.30}$$

Xét riêng trường hợp m=0 thì từ (5.30) được

$$G(j\omega) = \frac{1}{\omega T_a \sum_{l=-\infty}^{\infty} \dfrac{1}{\omega - l\Omega_a}} \tag{5.31}$$

Cũng với m=0 cho (5.25)

$$\widetilde{X}(j\omega) = \widetilde{F}_0(j\omega) \sum_{n=0}^{N-1} a_n e^{-j\omega nT_a} = \frac{1 - e^{-j\omega T_a}}{j\omega T_a} e^{j\omega\tau} \sum_{n=0}^{N-1} a_n e^{-j\omega nT_a}$$

và cho phương trình (5.21) được $a_n = T_a x_n$ (xem kết quả của ví dụ 1 trong mục 5.2.3). Do đó

$$\sum_{n=0}^{N-1} a_n e^{-j\omega n T_a} = T_a \underbrace{\sum_{n=0}^{N-1} x_n e^{-j\omega n T_a}}_{X_a(j\omega)}$$

Suy ra

$$\widetilde{X}(j\omega) = \frac{1-e^{-j\omega T_a}}{j\omega} e^{j\omega\tau} X_a(j\omega)$$

hay $$G(j\omega) = \frac{1-e^{-j\omega T_a}}{j\omega} \tag{5.32}$$

So sánh (5.31) và (5.32) được

$$T_a \sum_{l=-\infty}^{\infty} \frac{1}{\omega - l\Omega_a} = \frac{j}{1-e^{-j\omega T_a}} \tag{5.33}$$

Cuối cùng, thay (5.33) vào (5.30) ta sẽ có điều phải chứng minh. ❑

Bây giờ ta đi vào việc chính là xét sai số phổ sau khi tín hiệu rời rạc được chuyển đổi thành liên tục nhờ phương pháp B–spline và sẽ đưa ra kết luận sai số đó có chấp nhận được hay không?.

Từ định lý trên ta suy ra:

1) khi m=2 thì $$G(j\omega) = \frac{T_a}{\cos\dfrac{\omega T_a}{2}} \left(\frac{\sin\dfrac{\omega T_a}{2}}{\dfrac{\omega T_a}{2}} \right)^3 \tag{5.34}$$

2) khi m=4 thì $$G(j\omega) = \frac{6T_a}{\cos\dfrac{\omega T_a}{2}\left[5+\cos(\omega T_a)\right]} \left(\frac{\sin\dfrac{\omega T_a}{2}}{\dfrac{\omega T_a}{2}} \right)^5 \tag{5.35}$$

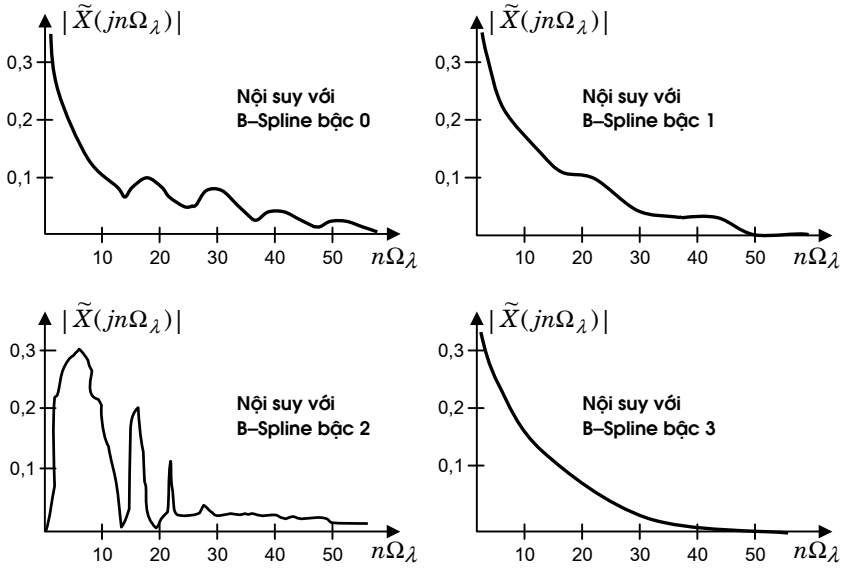
Hai công thức (5.34), (5.35) chỉ rằng hàm đặc tính tần $G(j\omega)$ sẽ tiến tới $\pm\infty$ tại các điểm tần số $\omega T_a = (2k+1)\pi$ với k nguyên. Điều đó dẫn đến sự phân kỳ của $\widetilde{X}(j\omega)$ tại những điểm tần số đó nếu sử dụng kỹ thuật B–spline bậc 2 và 4. Theo kinh nghiệm, bao giờ người ta cũng chỉ dùng các hàm B–spline $\widetilde{f}_m(t)$ bậc 0, hoặc bậc lẻ.

Ví dụ: Để minh họa ta xét một ví dụ. Cho tín hiệu

$$x(t) = e^{-3t}$$

Tín hiệu này có ảnh Fourier là

$$X(j\omega) = \frac{1}{3+j\omega}$$



Hình 5.9: Minh họa sự ảnh hưởng của phép nội suy B–Spline vào kết quả nhận dạng.

Từ 8 giá trị tín hiệu $x_k = x(kT_a)$, k = 0, 1, … ,7 và T_a = 0,2s ban đầu người ta đã sử dụng hàm B–spline bậc i= 0, 1, 2, 3 để chuyển nó thành tín hiệu liên tục $\tilde{x}(t)$ theo công thức (5.19). Tiếp tục, người ta trích mẫu với chu kỳ lấy mẫu mới nhỏ hơn là T_b = 0,01s để có 128 giá trị $\tilde{x}_k = \tilde{x}(kT_b)$, k = 0,1, … ,127 rồi sử dụng **`dft()`** sẽ được $\widetilde{X}(jn\Omega_\lambda)$ với $\Omega_\lambda = 4{,}9s^{-1}$.

Biểu diễn các giá trị $|\widetilde{X}(jn\Omega_\lambda)|$ đó theo n dưới dạng đồ thị có hình 5.9. Nhìn vào đường đồ thị của $\widetilde{X}(jn\Omega_\lambda)$ ứng với i =2 ta có ngay được kết luận rằng phép nội suy B–spline bậc 2 là không thể sử dụng được trong việc nhận dạng phổ tín hiệu. ❑

5.3 Ngoại suy

Thuật toán nhận dạng hàm đặc tính tần $G(j\omega)$ trình bày trong mục 2.3.1 cho ra kết quả là dãy $G(jn\Omega_\lambda)$, n=0,1, … ,2M, trong đó $\Omega_\lambda = \dfrac{2\pi}{\lambda T_a}$ xác định độ phân giải của kết quả và λ phải là một số nguyên lũy thừa của 2 nhỏ nhất nhưng không nhỏ hơn 2N–1. Như

vậy độ *phân giải của kết quả* phụ thuộc vào N là số các giá trị tín hiệu đo được, tức là vào khoảng thời gian quan sát các tín hiệu vào/ra $T = NT_a$. Khi N càng lớn, độ phân giải của kết quả càng cao.

Nhưng không phải bài toán nhận dạng nào trong thực tế cũng có khả năng đáp ứng được những điều kiện cần thiết cho phép thu được một kết quả có độ phân giải như ý muốn. Chẳng hạn do điều kiện khách quan không cho phép quan sát tín hiệu vào/ra trong một khoảng thời gian T đủ lớn (tín hiệu vào/ra là những quá trình ngẫu nhiên không dừng), nên số mẫu tín hiệu lấy được quá ít, kéo theo số lượng các giá trị tín hiệu nhận được không đủ nhiều để có thể phản ánh đầy đủ các thông tin cần thiết về đối tượng. Trong những trường hợp như vậy, thông thường người ta phải ngoại suy để tăng số mẫu tín hiệu thu được nhằm tăng độ phân giải cho kết quả.

Có rất nhiều các phương pháp ngoại suy tín hiệu khác nhau. Song phù hợp hơn cả với bài toán nhận dạng có tín hiệu vào/ra ngẫu nhiên là phương pháp cực đại *entropie*. Bài toán ngoại suy entropie phát biểu như sau: Giả sử rằng trong khoảng thời gian quan sát $[0, NT_a)$ ta đã thu được dãy giá trị $\{x_k\}$, với $x_k = x(kT_a)$, $k = 0, 1, \dots, N-1$ của tín hiệu $x(t)$. Vấn đề đặt ra ở đây là từ dãy gồm chỉ có N giá trị đó phải suy được ra những giá trị chưa có của tín hiệu nằm ngoài khoảng quan sát $[0, NT_a)$, tức là phải ngoại suy để được x_k với $k \geq N$, sao cho cùng với những giá trị ngoại suy này lượng thông tin entropie của tín hiệu $x(t)$ thu được từ dãy mới $\{x_k\}$, $k = 0, 1, \dots$ là lớn nhất.

5.3.1 Cực đại entropie loại 1

Gọi $\{x_k\}$, $k = 0, 1, \dots, N-1$ là dãy gồm N giá trị tín hiệu đo được trong khoảng quan sát $[0, NT_a)$. Theo kết quả của mục 4.2.2 thì sai lệch giữa giá trị đo được x_k và giá trị dự báo tuyến tính x_k^f tính theo

$$x_k^f = -\sum_{k=1}^{n_a} a_k x_{n-k}, \quad k = n_a, \dots, N-1 \tag{5.36}$$

sẽ có trung bình bình phương nhỏ nhất nếu bộ tham số $a_1, a_2, \dots, a_{n_a}, K$ được nhận dạng theo phương pháp Yule–Walker. Cũng với bộ tham số này, đại lượng entropie loại 1 của tín hiệu

$$H_1 = \int_{-\infty}^{\infty} \ln S_y(\omega) d\omega \tag{5.37}$$

có giá trị cực đại, trong đó mật độ phổ $S_x(\omega)$ của $\{x_k\}$ được tính bởi

$$S_x(\omega) = \frac{K}{1 + \sum_{k=1}^{n_a} a_k e^{-j\omega k T_a}} \tag{5.38}$$

Như vậy, phép ngoại suy x_k , $k \geq N$ nào đó không làm thay đổi giá trị trung bình bình phương các sai lệch $|x_k - x_k^f|$ cũng sẽ làm cho giá trị mật độ phổ (5.38) không thay đổi và do đó làm cho entropie H_1 (5.37) giữ được được giá trị cực đại của nó. Nhận xét này đưa ta đến phép ngoại suy đó phải là:

$$x_k = -\sum_{k=1}^{n_a} a_k x_{n-k} , \quad k \geq N \tag{5.39}$$

Ta có được thuật toán *ngoại suy cực đại entropie loại* 1 các giá trị x_k , $k \geq N$ như sau:

1) Chọn bậc n_a mô hình AR mô tả tín hiệu $x(t)$. Ta cũng có thể sử dụng kết quả bài tập 2 thuộc chương 4 để nhận dạng n_a.

2) Xác định a_1 , a_2, … , a_{n_a} từ $\{x_k\}$, k = 0, 1, … , N–1 bằng các thuật toán nhận dạng tham số mô hình AR đã biết như Yule–Walker hoặc Burg.

3) Thực hiện lần lượt với $k = N, N+1$, … công thức (5.39) để tính x_k với $k \geq N$.

5.3.2 Cực đại entropie loại 2

Bên cạnh thuật toán ngoại suy theo nguyên tắc cực đại entropie loại 1 vừa trình bày ta còn có một hình thức ngoại suy cực đại entropie khác để xác định x_k với $k \geq N$ là đi từ công thức entropie loại 2:

$$H_2 = -\int_{-\infty}^{\infty} S_x(\omega) \ln S_x(\omega) d\omega \tag{5.40}$$

Trước hết ta đặt vấn đề nhận dạng mật độ phổ $S_x(\omega)$ của $\{x_k\}$, k = 0, 1, … , N–1, là với $S_x(\omega)$ nhận dạng được, giá trị entropie loại 2 H_2 (5.40) sẽ lớn nhất. Tất nhiên mật độ phổ $S_x(\omega)$ có được đó phải thỏa mãn điều kiện biên

$$S_x(\omega) = T_a \sum_{m=-\infty}^{\infty} r_x(mT_a) e^{-jmT_a\omega}$$

$$\Leftrightarrow \quad r_x(mT_a) = \frac{1}{2\pi} \int_{-\infty}^{\infty} S_x(\omega) e^{jmT_a\omega} d\omega \tag{5.41}$$

trong đó $r_x(mT_a)$ là các giá trị hàm tương quan của tín hiệu $x(t)$ được tính theo công thức nhận dạng *bias* (2.45) hoặc *unbias* (2.47). Theo định lý 2.7 của chương 2 thì chỉ số m trong điều kiện biên (5.41) chỉ cần chạy từ $-N+1$ đến $N-1$ là đủ.

Như vậy bài toán tìm cực đại cho (5.40) với điều kiện biên (5.41) vừa nêu là một bài toán tối ưu tĩnh và nghiệm của nó có thể được tìm một cách đơn giản theo nguyên lý của Lagrange. Trước hết ta lập hàm

$$Q = H_2 + \sum_{m=-N+1}^{N-1} \lambda_m \left[\int_{-\infty}^{\infty} S_x(\omega) e^{jmT_a\omega} d\omega - 2\pi \cdot r_x(mT_a) \right] \tag{5.42}$$

trong đó λ_m , $m=-N+1, \dots, N-1$ là những hằng số Lagrange. Sau đó ta đạo hàm hai vế của (5.42) theo biến $S_x(\omega)$:

$$\frac{\partial Q}{\partial S_x(\omega)} = \int_{-\infty}^{\infty} \left[-\left(1 + \ln|S_x(\omega)|\right) + \sum_{m=-N+1}^{N-1} \lambda_m e^{jmT_a\omega} \right] d\omega$$

rồi tìm điều kiện để đạo hàm đó bị triệt tiêu sẽ được:

$$\sum_{m=-N+1}^{N-1} \lambda_m e^{jmT_a\omega} = 1 + \ln|S_x(\omega)| \tag{5.43}$$

$$\Leftrightarrow \qquad S_x(\omega) = \exp\left(-1 + \sum_{m=-N+1}^{N-1} \lambda_m e^{jmT_a\omega} \right) \tag{5.44}$$

Vấn đề còn lại là xác định các hệ số λ_m , $m=-N+1, \dots, N-1$. Có nhiều cách xác định λ_m với những ưu nhược điểm khác nhau. Ở đây ta sẽ đi theo phương pháp của Wu giới thiệu trong tài liệu [18].

Nếu gọi $\{c_m\}$, $c_m = c(mT_a)$ là dãy các giá trị của hàm $c(t)$ có ảnh Fourier

$$C(j\omega) = \ln|S_x(\omega)|$$

thì khi chuyển ngược công thức (5.43) sang miền thời gian sẽ có

$$\sum_{m=-N+1}^{N-1} \lambda_m \delta(t - mT_a) = \delta(t) + c(t) , \tag{5.45}$$

vì ảnh Fourier của $\delta(t-mT_a)$ là $e^{jmT_a\omega}$. Do $C(j\omega)$ là hàm thuần thực nên $c(t)$ phải là một hàm chẵn, tức là $c(t) = c(-t)$. Suy ra $\lambda_m = \lambda_{-m}$.

Viết lại (5.45) lần lượt cho $t=mT_a$, $m=0,1, \dots, N-1$ được:

- khi $m=0$: $\lambda_0 = 1+c_0$

– khi m=1: $\lambda_1 = c_1$

⋮

– khi m=N–1: $\lambda_{N-1} = c_{N-1}$

và đó chính là những công thức xác định hệ số λ_m. Ta đi đến thuật toán nhận dạng mật độ phổ $S_x(\omega)$ của $x(t)$ từ dãy $\{x_k\}$, k = 0, 1, … , N–1 theo nguyên tắc cực đại entropie loại 2 như sau:

1) Sử dụng hàm **spec()** đã giới thiệu trong chương 2 để xác định $\widetilde{S}_x(n\Omega_\lambda)$ n=0,1, … , λ–1 với λ=2N–1 và chỉ số Lag M=N–1.

2) Tính $C(n\Omega\lambda) = \ln\left|\widetilde{S}_x(n\Omega_\lambda)\right|$, n=0,1, … , λ–1.

3) Chuyển ngược dãy $\{C(n\Omega\lambda)\}$ nhờ hàm **invdt()** để có $\{c_m\}$, m=– N+1, … , N–1.

4) Xác định $\lambda_m = \begin{cases} 1 + c_0 & \text{khi} \quad m = 0 \\ c_m & \text{khi} \quad m = 1, \cdots, N-1 \\ \lambda_{-m} & \text{khi} \quad m < 0 \end{cases}$.

5) Tính $S_x(\omega)$ theo công thức (5.44).

Tiếp tục, theo định lý 5.3 cho mật độ phổ $S_x(\omega)$, tức là:

$$S_x(\omega) = \frac{T_a}{p}\left|X_a(j\omega)\right|^2 \qquad \text{với } p = \begin{cases} N & \text{nếu nhận dạng bias } r_x(mT_a) \\ N-m & \text{nếu nhận dạng unbias } r_x(mT_a) \end{cases}$$

N. Wu đã đưa ra trong tài liệu [18] mối quan hệ giữa c_m và x_k khi $x_0 > 0$ như sau:

$$c_m = \begin{cases} \ln x_0 & \text{khi} \quad m = 0 \\ \dfrac{x_m}{x_0} - \displaystyle\sum_{k=0}^{m-1} \frac{k c_k x_{m-k}}{m x_0} & \text{khi} \quad m > 0 \end{cases}$$

$$x_k = \begin{cases} e^{c_0} & \text{khi} \quad k = 0 \\ c_k x_0 + \displaystyle\sum_{m=0}^{k-1} \frac{m c_m x_{k-m}}{k} & \text{khi} \quad k > 0 \end{cases}$$

Sử dụng quan hệ đó ta có được thuật toán ngoại suy x_k , $k \geq N$ gồm các bước:

1) Xác định $\widetilde{S}_x(n\Omega_\lambda)$ n=0,1, … , λ–1 nhờ hàm **spec()** với λ=2N–1 và chỉ số Lag M=N–1.

2) Tính $C(n\Omega\lambda) = \ln\left|\widetilde{S}_x(n\Omega_\lambda)\right|$, n=0,1, … , λ–1 và chuyển ngược dãy $\{C(n\Omega\lambda)\}$ sang miền thời gian nhờ hàm **`invdt()`** để có $\{c_m\}$, m=– N+1, … , N–1.

3) Ngoại suy x_k lần lượt với $k = N, N+1, \ldots$ theo công thức

$$x_k = \sum_{m=0}^{k-1} \frac{mc_m x_{k-m}}{k} \quad \text{với} \quad c_m=0 \text{ khi } m \geq N.$$

5.4 Lý thuyết hàm mở rộng

5.4.1 Định nghĩa

Ngay đầu chương 2 ta đã khẳng định rằng nhờ có lý thuyết hàm mở rộng mà bản chất toán học của "hàm số" dirac $\delta(t)$ mới được hiểu một cách chặt chẽ và tổng quát. Khái niệm "hàm số" của $\delta(t)$ ở đây không được đúng như định nghĩa toán học kinh điển của nó (nên nó được viết trong dấu ngoặc kép). Thực chất $\delta(t)$ là một *phiếm hàm* (functional) liên tục, tuyến tính trên $D \subset C^\infty(\mathbb{R})$, hay còn gọi là một hàm mở rộng, trong đó $C^\infty(\mathbb{R})$ là ký hiệu chỉ tập hợp các hàm thực liên tục, có đạo hàm vô hạn lần trên $\mathbb{R}$ và D là một tập con của $C^\infty(\mathbb{R})$ gồm các hàm có thời gian sống hữu hạn, tức là hàm có

$$\text{supp } \varphi(t)=\{ t \in \mathbb{R} \mid \varphi(t) \neq 0\}$$

giới nội. Ví dụ như hàm

$$\varphi(t) = \begin{cases} a \exp\left(\dfrac{1}{t^2 - b^2}\right) & \text{khi } |t| \leq b \\ 0 & \text{khi } |t| > b \end{cases} \tag{5.46}$$

là một phần tử của D. Ví dụ này cũng chứng tỏ rằng D không rỗng. Theo thuật ngữ kỹ thuật thì hàm mở rộng được xem như là một quá trình biến đổi tín hiệu $\varphi(t) \in D$ thành hằng số.

Định nghĩa 5.1: Gọi D là tập con của $C^\infty(\mathbb{R})$ gồm các hàm có thời gian sống hữu hạn. Khi đó một phiếm hàm liên tục, tuyến tính $r(t)$: $D \to \mathbb{R}$ sẽ được gọi là *hàm mở rộng* nếu sự hội tụ $\varphi_n(t) \to \varphi(t)$ của một dãy $\{\varphi_n(t)\}$ bất kỳ trên D thỏa mãn:

a) với $k \in \mathbb{N}$ tùy ý thì hợp của tất cả k miền thời gian sống supp$\varphi_n(t)$ của các hàm $\varphi_n(t)$; n =1,2, ... , k cũng giới nội trên $\mathbb{R}$,

b) $\varphi_n(t) \to \varphi(t) \Leftrightarrow \dfrac{d^m}{dt^m}\varphi_n(t) \to \dfrac{d^m}{dt^m}\varphi(t)$; $\forall m$ nguyên và $m \geq 0$, trong đó $\dfrac{d^m}{dt^m}$ là ký hiệu của đạo hàm bậc m của một hàm thường (hội tụ đều),

Tập hợp tất cả các phiếm hàm liên tục, tuyến tính trên D được ký hiệu bằng D' (hay *không gian đối ngẫu*). Khái niệm hội tụ trên D là cần thiết, vì chỉ khi đó chúng ta mới có được định nghĩa về sự liên tục của một phiếm hàm trên D.

Định nghĩa 5.2: Một hàm mở rộng $r(t) \in D'$ có các phép biến đổi giống như một tích phân

$$\varphi(t) \xrightarrow{r(t)} \int_{-\infty}^{\infty} r(t)\varphi(t)dt \,;\forall \varphi(t) \in \boldsymbol{D} \tag{5.47}$$

được gọi là *hàm mở rộng đều* (regular).

Chú ý rằng dấu tích phân trong công thức (5.47) chỉ có ý nghĩa hình thức. Nó được sử dụng ở đây để nói rằng $r(t)$ có các phép biến đổi, như phép cộng, phép lấy đạo hàm … , giống các phép biến đổi của một tích phân, chứ bản thân nó không phải là một tích phân theo nghĩa toán học thông thường như tích phân Riemann hay tích phân Lebesgue.

Xét hàm Heaviside $1(t)$. Với tích phân

$$\int_{-\infty}^{\infty} 1(t)\varphi(t)dt = \int_{0}^{\infty} \varphi(t) < \infty \;;\forall \varphi(t) \in \boldsymbol{D}$$

thì rõ ràng hàm Heaviside $1(t)$ cũng là một hàm mở rộng đều. Từ đó suy ra *mọi hàm thực khác, nếu liên tục từng đoạn và bị chặn, thì cũng được xem như là hàm mở rộng đều*, tức là cũng thuộc D'.

Định nghĩa 5.3: Một hàm mở rộng đều $r(t)$, nếu $\int_{-\infty}^{\infty} r(t)\varphi(t)dt = 0$

a) với mọi $\varphi(t)=0$ có $a \le t \le b$ thì được gọi là có *miền xác định* $a \le t \le b$ (đồng nhất bằng 0 ngoài khoảng kín $[a,b]$),

b) với mọi $\varphi(t)=0$ có $t \notin [a,b]$ thì được gọi là *đồng nhất bằng* 0 trong khoảng kín $[a,b]$.

Định nghĩa 5.4: Hàm mở rộng dirac $\delta(t)$ là một hàm mở rộng đều thỏa mãn tính chất

$$\int_{-\infty}^{\infty} \delta(t)\varphi(t)dt = \varphi(0) \;,\; \forall \varphi(t) \in D. \tag{5.48}$$

Từ nay về sau hàm mở rộng dirac $\delta(t)$ sẽ được gọi đơn giản là *hàm delta*. Với mọi hàm $\varphi(t) \in D$ có $\varphi(0) = 0$ thì

$$\int_{-\infty}^{\infty} \delta(t)\varphi(t)dt = \varphi(0) = 0,$$

do đó mà hàm delta, theo định nghĩa 5.4, có thời gian sống hữu hạn (miền xác định của hàm delta chỉ chứa có một điểm là 0).

Định nghĩa 5.5: Đạo hàm $\frac{dr}{dt}$ của hàm mở rộng đều $r(t)$ là một hàm mở rộng đều của D' thỏa mãn

$$\int_{-\infty}^{\infty} \frac{dr(t)}{dt}\varphi(t)dt = -\int_{-\infty}^{\infty} r(t)\frac{d\varphi(t)}{dt}dt \ , \forall \varphi(t)\in D. \tag{5.49}$$

Với định nghĩa này thì mọi hàm liên tục từng đoạn và bị chặn đều có đạo hàm (còn gọi là *đạo hàm tổng quát, hay đạo hàm Sobolew*), mặc dù theo ý nghĩa đạo hàm kinh điển, đạo hàm của chúng có thể không có hoặc không xác định tại một hay nhiều điểm.

Ví dụ : Theo định nghĩa 5.5 thì từ

$$\int_{-\infty}^{\infty} \dot{1}(t)\varphi(t)dt = -\int_{0}^{\infty} 1(t)\dot{\varphi}(t)dt = -\varphi(t)\Big|_0^{\infty}$$

$$= \varphi(0) = \int_{-\infty}^{\infty} \delta(t)\varphi(t)dt \ , \ \forall \varphi(t)\in D.$$

ta có ngay được đạo hàm (tổng quát) của hàm Heaviside và nó chính là hàm delta $\delta(t)$. Tiếp theo, đạo hàm $\dot{\delta}(t)$ của hàm delta sẽ có giá trị là

$$\int_{-\infty}^{\infty} \dot{\delta}(t)\varphi(t)dt = -\int_{-\infty}^{\infty} \delta(t)\dot{\varphi}(t)dt = -\dot{\varphi}(0) \ .$$ ❑

5.4.2 Tính chất

Do trong khuôn khổ áp dụng vào kỹ thuật thường chỉ làm việc với hàm mở rộng đều, nên từ nay về sau, *một hàm mở rộng đều sẽ được gọi ngắn gọn là hàm mở rộng*. Từ định nghĩa 5.2 có:

a) Tổng của hai hàm mở rộng $r(t) = r_1(t)+r_2(t)$ là một hàm mở rộng và được xác định như phép tính tổng của một tích phân, tức là

$$\int_{-\infty}^{\infty} r(t)\varphi(t)dt = \int_{-\infty}^{\infty} r_1(t)\varphi(t)dt + \int_{-\infty}^{\infty} r_2(t)\varphi(t)dt$$

với mọi $\varphi(t)\in D$.

b) Phép tịnh tiến một khoảng thời gian τ trên trục thời gian của hàm mở rộng $r(t)$ cho ra một hàm mở rộng $r(t-\tau)$ và được biểu diễn thành

$$\int_{-\infty}^{\infty} r(t-\tau)\varphi(t)dt = \int_{-\infty}^{\infty} r(t)\varphi(t+\tau)dt \ .$$

c) Hàm mở rộng $r(at)$, $a\in\mathbb{R}$ được xác định từ $r(t)$ nhờ phép biến đổi

$$\int_{-\infty}^{\infty} r(at)\varphi(t)dt = \frac{1}{|a|}\int_{-\infty}^{\infty} r(t)\varphi\left(\frac{t}{a}\right)dt \ , \ \forall \varphi(t)\in D.$$

d) Tích $r(t)\gamma(t)$ của hàm mở rộng $r(t)$ với hàm thường $\gamma(t)\in C^\infty(\mathbb{R})$ cũng là một hàm mở rộng và được xác định bởi

$$\int_{-\infty}^{\infty}[r(t)\gamma(t)]\varphi(t)dt = \int_{-\infty}^{\infty} r(t)[\gamma(t)\varphi(t)]dt \tag{5.50a}$$

Vế phải hoàn toàn có nghĩa, vì $\gamma(t)\varphi(t)\in D$.

e) Nếu trong công thức (5.50a) ta thay hàm delta $\delta(t-T_a)$ vào vị trí của hàm mở rộng $r(t)$ và tín hiệu $x(t)$, liên tục tại T_a vào vị trí của $\gamma(t)$ thì sẽ có được công thức lấy giá trị tín hiệu $x(t)$ tại thời điểm T_a như sau:

$$\int_{-\infty}^{\infty}[\delta(t-T_a)x(t)]\varphi(t)dt = x(T_a)\varphi(T_a) = x(T_a)\int_{-\infty}^{\infty}\delta(t-T_a)\varphi(t)dt \tag{5.50b}$$

Tín hiệu $x(t)$ ở đây không cần phải có đạo hàm vô hạn lần như $\gamma(t)$, mà chỉ cần là một hàm thường và liên tục tại T_a là đủ. Sau này, khi mà khái niệm tín hiệu đã được định nghĩa một cách rõ ràng và sự nhầm lẫn giữa một tín hiệu với một hàm mở rộng hoàn toàn có thể bị loại bỏ thì công thức (5.50b) sẽ được viết ngắn gọn thành

$$x(t)\delta(t-T_a) = x(T_a)\delta(t-T_a). \tag{5.50c}$$

Ví dụ 1: Để làm quen với cách viết mới này, ta thử tính giá trị của tích $x(t)\dfrac{d\delta(t)}{dt}$. Trước hết, theo định nghĩa 5.5 về đạo hàm của hàm mở rộng (đều) thì

$$\int_{-\infty}^{\infty} x(t)\frac{d\delta(t)}{dt}\varphi(t)dt = -\int_{-\infty}^{\infty}\delta(t)\frac{dx(t)\varphi(t)}{dt}dt = -\int_{-\infty}^{\infty}\delta(t)\frac{dx(t)}{dt}\varphi(t)dt - \int_{-\infty}^{\infty}\delta(t)x(t)\frac{d\varphi(t)}{dt}dt.$$

Từ đó, với định nghĩa 5.4, có

$$\begin{aligned}\int_{-\infty}^{\infty}\left[x(t)\frac{d\delta(t)}{dt}\right]\varphi(t)dt &= -\frac{dx(0)}{dt}\varphi(0) - x(0)\frac{d\varphi(0)}{dt}\\ &= -\frac{dx(0)}{dt}\int_{-\infty}^{\infty}\delta(t)\varphi(t)dt + x(0)\int_{-\infty}^{\infty}\frac{d\delta(t)}{dt}\varphi(t)dt\end{aligned}$$

và cuối cùng, theo cách viết gọn của (5.50c) thì

$$x(t)\frac{d\delta(t)}{dt} = -\frac{dx(0)}{dt}\delta(t) + x(0)\frac{d\delta(t)}{dt}. \tag{5.50d}$$

Từ công thức (5.5d), với $x(t)=t$, có được phương trình lý thú sau:

$$t\frac{d\delta(t)}{dt} = -\delta(t) \tag{5.50e}$$

Ví dụ 2: Gọi $r(t)$ là một hàm mở rộng, $\dfrac{dr(t)}{dt}$ là đạo hàm của nó. Giả sử tiếp tồn tại ảnh Laplace $R(s)$, $\widetilde{R}(s)$ của chúng, tức là

$$R(s)=\int_{-\infty}^{\infty} r(t)e^{-st}dt \quad \text{và} \qquad \widetilde{R}(s)=\int_{-\infty}^{\infty}\frac{dr(t)}{dt}e^{-st}dt$$

Vậy thì từ định nghĩa 5.4 có

$$\widetilde{R}(s)=\int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty}\frac{dr(t)}{dt}e^{-st}dt\right]\varphi(s)ds=-\int_{-\infty}^{\infty}r(t)\left[\frac{d}{dt}\int_{-\infty}^{\infty}\varphi(s)e^{-st}ds\right]dt$$

$$=\int_{-\infty}^{\infty}\left[s\int_{-\infty}^{\infty}r(t)e^{-st}dt\right]\varphi(s)ds=sR(s).$$ ❑

f) Cho trước hai hàm mở rộng $r_1(t)$ và $r_2(t)$. Tích chập $r_1(t)*r_2(t)$ của chúng được hiểu là

$$\int_{-\infty}^{\infty}[r_1(t)*r_2(t)]\varphi(t)dt=\int_{-\infty}^{\infty}r_1(t)\left[\int_{-\infty}^{\infty}r_2(\tau)\varphi(t+\tau)d\tau\right]dt \tag{5.51}$$

Tuy nhiên, cần phải xét xem khi nào thì tích phân bên phải của (5.51) có nghĩa (tức là tích phân có tồn tại hay không?, theo nghĩa hàm mở rộng và khi đó tích chập cũng sẽ là một hàm mở rộng). Nói cách khác là phải xét xem, khi nào thì có được

$$\int_{-\infty}^{\infty}r_2(\tau)\varphi(t+\tau)d\tau\in D.$$

Căn cứ theo định nghĩa 5.3, sẽ có được điều kiện này, nếu như tồn tại một miền giới nội sao cho $r_2(t)$ đồng nhất bằng 0 ngoài miền đó, hay $r_2(t)$ có thời gian sống hữu hạn. Ngoài ra, theo định nghĩa 5.2 thì tích chập có tính giao hoán $r_1(t)*r_2(t) = r_2(t)*r_1(t)$, bởi vậy *tích chập hai hàm mở rộng cũng sẽ là một hàm mở rộng, nếu ít nhất một trong hai hàm mở rộng có thời gian sống hữu hạn.*

Cho một hàm mở rộng $r(t)$. Vì hàm delta có thời gian sống hữu hạn nên giá trị của tích chập $r(t)*\delta(t)$ cũng là một hàm mở rộng và

$$\int_{-\infty}^{\infty}r(t)\left[\int_{-\infty}^{\infty}\delta(\tau)\varphi(t+\tau)d\tau\right]dt=\int_{-\infty}^{\infty}r(t)\varphi(t)dt \quad ,\forall\varphi(t)\in D \tag{5.52a}$$

nói cách khác

$$r(t)*\delta(t)=\delta(t)*r(t)=r(t) \tag{5.52b}$$

Như vậy, không gian D' với quan hệ tích chập có $\delta(t)$ là *phần tử đơn vị*. Cũng từ công thức (5.51) dễ dàng chứng minh được rằng

$$r(t)*\delta(t-t_0) = r(t-t_0). \tag{5.52c}$$

g) Khái niệm giới hạn của một dãy hàm mở rộng cũng được xây dựng thông qua dấu tích phân. Giả sử có trong D' một dãy hàm mở rộng $\{r_n(t)\}$ hội tụ tới $r(t)$, tức là $r_n(t) \to r(t)$ khi $n \to \infty$, thì sự hội tụ đó được hiểu theo nghĩa

$$\lim_{n\to\infty} \int_{-\infty}^{\infty} r_n(t)\varphi(t)dt \int_{-\infty}^{\infty} r(t)\varphi(t)dt, \qquad \forall \varphi(t) \in D \tag{5.53}$$

(hội tụ đều). Nhớ đến định lý của Steinhaus: "*Nếu E là một không gian Banach, F là một không gian chuẩn và* $T_n : E \to F$ *là một dãy ánh xạ liên tục tuyến tính thoả mãn* $\lim_{n\to\infty} T_n x = Tx$ *với mọi* $x \in E$ *thì T cũng liên tục, tuyến tính*", có được $r(t) \in D'$, hay $r(t)$ cũng là một hàm mở rộng.

5.4.3 Toán tử Fourier mở rộng

Ở mục 5.4.1 và 5.4.2 ta đã đề cập đến những khái niệm cơ bản về hàm mở rộng từ định nghĩa, tính chất, cho tới các phép biến đổi được xây dựng trên giả thiết rằng chúng đều. Những khái niệm trên hoàn toàn có thể được xây dựng một cách tương tự cho không gian $\mathbb{R}^n$, tức là với các hàm $\varphi \in C^\infty(\mathbb{R}^n)$ và có thời gian sống hữu hạn. Tuy nhiên, để có thể áp dụng được chúng vào việc mô hình hóa và phân tích hệ thống, phục vụ điều khiển kỹ thuật trong miền phức (nhất là các hệ suy biến - *singular systems*), thì vẫn còn thiếu khái niệm về ảnh Fourier của hàm mở rộng mà trong một vài tài liệu khác nhau còn được gọi là *Toán tử Fourier mở rộng*.

Cho trước một hàm mở rộng $r(t)$. Giả sử tồn tại $R(j\omega)$ là ảnh Fourier của $r(t)$. Nếu như $R(j\omega)$ cũng là một hàm mở rộng thì với mọi hàm $\varphi(\omega) \in D$ phải có

$$\int_{-\infty}^{\infty} R(j\omega)\varphi(\omega)d\omega = \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} r(t)e^{-j\omega t}dt \right] \varphi(\omega)d\omega = \int_{-\infty}^{\infty} r(t)\phi(jt)dt, \tag{5.54a}$$

trong đó $\phi(jt)$ là ảnh Fourier của hàm số $\varphi(\omega)$ và cũng phải thuộc D, tức là

$$\phi(jt) = \int_{-\infty}^{\infty} \varphi(\omega)e^{-j\omega t}d\omega \in D. \tag{5.54b}$$

Chú ý rằng dấu tích phân trong công thức (5.54b) và tích phân trong dấu ngoặc vuông của công thức (5.54a) chính là tích phân của toán tử Fourier liên tục và là tích phân toán học thông thường (tích phân *Riemann*), trong khi các tích phân còn lại chỉ là một ký hiệu của hàm mở rộng đều.

Định nghĩa thì rõ ràng như vậy, nhưng nếu có câu hỏi được đặt ra rằng $R(j\omega) \neq 0$ có tồn tại hay không? (theo nghĩa hàm mở rộng), thì ta có thể trả lời ngay được rằng với những giả thiết đã được xây dựng cho đến nay $R(j\omega)$ không thể tồn tại được, vì $\phi(jt)$

không thuộc D. Hàm $\phi(jt)\neq 0$ có đạo hàm vô hạn lần giống như $\varphi(\omega)$, nhưng thời gian sống là vô hạn (ảnh Fourier của một hàm có miền xác định giới nội sẽ xác định trên toàn bộ trục số), bởi vậy $\phi(jt)\notin D$. Nhớ lại phép biến đổi Fourier thông thường đã được trình bày trong chương 2 thì khi hàm $\varphi(\omega)$ có $\text{supp}\varphi(\omega)$ giới nội, ảnh Fourier $\phi(jt)$ của nó sẽ có $\text{supp}\phi(jt)$ gần như là toàn bộ trường số thực, nói cách khác thời gian sống của $\phi(jt)$ hữu hạn khi và chỉ khi $\varphi(\omega)\equiv 0$.

Bởi vậy, để có thể có được $R(j\omega)$ thì cần phải mở rộng D. Ta xét tập E của tất cả các hàm của $C^{\infty}(\mathbb{R})$ nhưng bây giờ không bắt buộc là phải có thời gian sống hữu hạn mà chỉ cần tiến tới 0 khi $t \to \pm\infty$ nhanh hơn bất cứ một đa thức nào khác của t^{-1}.

Định nghĩa 5.6: Một hàm mở rộng (đều), xác định trên E với

a) $D \subset E$,

b) Sự hội tụ của một dãy $\{\varphi_m(t)\}$ trên E được định nghĩa như trên D ,

c) $\lim\limits_{|t|\to\infty} |t|^n \varphi(t) = 0$ với mọi n nguyên,

được gọi là *hàm mở rộng yếu*. Tập hợp tất cả các hàm mở rộng yếu được ký hiệu bằng E'.

Định nghĩa 5.7: Ảnh *Fourier* $R(j\omega)$ của hàm mở rộng yếu $r(t)$ cũng là một hàm mở rộng yếu và được xác định từ $r(t)$ như sau:

$$\int_{-\infty}^{\infty} R(j\omega)\varphi(\omega)d\omega = \int_{-\infty}^{\infty} r(t)\phi(jt)dt \ , \ \forall \varphi(t)\in D, \tag{5.55}$$

trong đó $\phi(jt)$ là ảnh Fourier của hàm số $\varphi(\omega)$.

Từ nay về sau ký hiệu $\mathcal{F}$ sẽ được sử dụng chỉ toán tử Fourier, $\phi(j\omega)=\mathcal{F}\{\varphi(t)\}$ chỉ ảnh Fourier của $\varphi(t)$ và *mọi hàm mở rộng (đều) được nói đến là hàm mở rộng yếu.*

Bây giờ, giả sử rằng $r(t)$ là một hàm thường có ảnh Fourier, tức là tồn tại $\mathcal{F}[r(t)]$. Với ký hiệu d^m để chỉ phép lấy đạo hàm bậc thứ m của một hàm mở rộng và do $r(t)$ cũng là một hàm mở rộng nên theo định nghĩa 5.5 và 5.7 có được

$$\int_{-\infty}^{\infty} d^m r(t)\phi(jt)dt = (-1)^m \int_{-\infty}^{\infty}\left[r(t)\frac{d^m}{dt^m}\left(\int_{-\infty}^{\infty} e^{-j\omega t}\varphi(\omega)d\omega \right)\right]dt$$

Suy ra

$$\begin{aligned}\mathcal{F}[d^m r(t)] &= \int_{-\infty}^{\infty}\left[r(t)\int_{-\infty}^{\infty}(j\omega)^m e^{-j\omega t}\varphi(\omega)d\omega\right]dt \\ &= \int_{-\infty}^{\infty}\left((j\omega)^m\varphi(\omega)\int_{-\infty}^{\infty} r(t)e^{-j\omega t}dt\right)d\omega ,\end{aligned}$$

hay nói cách khác

$$\mathcal{F}[d^m r(t)] = (j\omega)^m \mathcal{F}[r(t)].$$

Cũng tương tự như vậy cho $d^m\{\mathcal{F}[r(t)]\}$ có

$$\int_{-\infty}^{\infty} d^m \mathcal{F}[r(t)]\varphi(\omega)d\omega = \int_{-\infty}^{\infty} \frac{d^m}{d\omega^m}\left[\int_{-\infty}^{\infty} r(t)e^{j\omega t}dt\right]\varphi(\omega)d\omega$$

$$= \int_{-\infty}^{\infty} (-jt)^m r(t)\left[\int_{-\infty}^{\infty} \varphi(\omega)e^{j\omega t}dt\right]d\omega$$

Suy ra

$$d^m\{\mathcal{F}[r(t)]\} = \mathcal{F}[(jt)^m r(t)] .$$

Tổng quát lên cho E' ta đi đến:

Định lý 5.5: Với một hàm mở rộng $r(t) \in E'$ có

a) $\mathcal{F}[d^m r(t)] = (j\omega)^m \mathcal{F}[r(t)]$,

b) $d^m\{\mathcal{F}[r(t)]\} = \mathcal{F}[(jt)^m r(t)]$.

Bên cạnh định lý về ảnh của đạo hàm cũng như đạo hàm của ảnh, ta còn có kết luận nữa về ảnh của tích chập cũng rất cần thiết trong nhận dạng được phát biểu như sau:

Định lý 5.6: Giả sử $r(t), r_1(t), r_2(t) \in E'$, thì

a) $\mathcal{F}[r(t-T)] = \mathcal{F}[r(t)]\, e^{-j\omega T}$ và

b) $\mathcal{F}[r_1(t)*r_2(t)] = \mathcal{F}[r_1(t)]\mathcal{F}[r_2(t)]$ nếu tồn tại tích chập $[r_1(t)*r_2(t)]$.

Chứng minh:

a) Vì

$$\int_{-\infty}^{\infty} r(t-T)\phi(jt)dt = \int_{-\infty}^{\infty} r(t)\phi[j(t+T)]dt = \int_{-\infty}^{\infty} r(t)\left[\int_{-\infty}^{\infty} e^{-j(t+T)\omega}\varphi(\omega)d\omega\right]dt$$

nên

$$\mathcal{F}[r(t-T)] = e^{-j\omega T}\int_{-\infty}^{\infty} r(t)\left[\int_{-\infty}^{\infty} e^{-jt\omega}\varphi(\omega)d\omega\right]dt = e^{-j\omega T}\int_{-\infty}^{\infty} r(t)\phi(jt)dt$$

và đó là đ.p.c.m. thứ nhất.

b) Do phép tính tích chập có tính giao hoán, nên sẽ không mất tính tổng quát nếu được giả thiết thêm rằng $r_2(t)$ có thời gian sống hữu hạn. Khi đó, cùng với (5.51) có

$$\mathcal{F}[r_1(t)*r_2(t)] = \int_{-\infty}^{\infty}[r_1(t) * r_2(t)]\phi(jt)dt = \int_{-\infty}^{\infty} r_1(t)\left[\int_{-\infty}^{\infty} r_2(\tau)\phi(j(t+\tau)d\tau\right]dt$$

Áp dụng công thức toán tử Fourier cho cho $\phi(j(t+\tau))$ được

$$\mathcal{F}[r_1(t)*r_2(t)] = \int_{-\infty}^{\infty} r_1(t)\left[\int_{-\infty}^{\infty} r_2(\tau)\left(\int_{-\infty}^{\infty} e^{-j(t+\tau)\omega}\varphi(\omega)d\omega\right)d\tau\right]dt$$

và cuối cùng là điều phải chứng minh thứ hai

$$\mathcal{F}[r_1(t)*r_2(t)] = \int_{-\infty}^{\infty}\left[\left(\int_{-\infty}^{\infty} r_1(t)e^{-jt\omega}dt\right)\left(\int_{-\infty}^{\infty} r_2(\tau)e^{-j\tau\omega}d\tau\right)\right]\varphi(\omega)d\omega \qquad \square$$

Từ định lý trên với $r_1(t)=\delta(t)$ dễ dàng thu được

$\mathcal{F}[r_2(t)] = \mathcal{F}[\delta(t)]\mathcal{F}[r_2(t)]$ hay $\mathcal{F}[\delta(t)] = 1$.

Ở đây 1 phải được hiểu là một hàm mở rộng. Bằng lời, định lý 5.2 còn được diễn tả thành: "*ảnh của tích chập, nếu chúng tồn tại, chính là tích của hai ảnh*".

Do $\mathcal{F}$ là một isomorphism trên E và ảnh Fourier $R(j\omega)$ của một hàm mở rộng $r(t)$ được định nghĩa nhờ ảnh của hàm $\varphi(\omega)\in E$ nên điều ngược lại cũng đúng: "*ảnh của một tích bằng tích chập của hai ảnh, nếu tồn tại tích chập đó*", tức là

$$\mathcal{F}[r_1(t)r_2(t)] = \frac{1}{2\pi}\mathcal{F}[r_1(t)]*\mathcal{F}[r_2(t)] \tag{5.56}$$

Khác với định lý 5.6, ở công thức (5.56) còn có thêm hệ số $\frac{1}{2\pi}$ do định nghĩa về toán tử Fourier ngược sinh ra. Có thể dễ dàng kiểm chứng lại tính đúng đắn của công thức (5.56) tương tự như đã làm với định lý 5.6.

Nếu $\varphi(t)\in E$, thì tồn tại ảnh Fourier $\phi(j\omega)$ của nó. Xuất phát từ công thức của toán tử Fourier ngược đối với hàm $\varphi(t)$ có được

$$\begin{aligned}\varphi(0) &= \frac{1}{2\pi}\int_{-\infty}^{\infty}\phi(j\omega)d\omega \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty}\varphi(t)e^{-j\omega t}dt\right]d\omega = \int_{-\infty}^{\infty}\left[\frac{1}{2\pi}\int_{-\infty}^{\infty}e^{-j\omega t}d\omega\right]\varphi(t)dt\end{aligned}$$

Do đẳng thức trên đúng với mọi hàm $\varphi(t)$ thuộc E (tức là cũng đúng với mọi hàm $\varphi(t)\in D$) nên theo định nghĩa 5.4 về hàm $\delta(t)$ sẽ thu được

$$\delta(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty}e^{-j\omega t}d\omega = \frac{1}{2\pi}\int_{-\infty}^{\infty}\cos(\omega t)d\omega = \frac{1}{2\pi}\int_{-\infty}^{\infty}e^{j\omega t}d\omega \tag{5.57a}$$

để ý rằng ở đây đã sử dụng công thức $e^{-j\omega t} = \cos(\omega t) - j\sin(\omega t)$ (công thức Euler) và $\sin(\omega t)$ là hàm chẵn. Từ đây suy ra

$$\delta(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\cos(\omega t)d\omega = \lim_{a\to\infty}\frac{1}{2\pi}\int_{-a}^{a}\cos(\omega t)d\omega = \lim_{a\to\infty}\frac{\sin(at)}{\pi t} \tag{5.57b}$$

Các công thức (5.57a) và (5.57b) là hai trong những công thức rất cơ bản, được sử dụng nhiều trong các công việc nghiên cứu và ứng dụng khác nhau của hàm $\delta(t)$.

Câu hỏi ôn tập và bài tập

1. Hãy chỉ rằng hàm đặc tính tần $G(j\omega)$ của khối D/A theo kỹ thuật B–spline bậc 0, 1,3 là những hàm bị chặn, tức là $|G(j\omega)|<\infty$ với mọi ω. Từ đó có thể rút ra được một kết luận chung nào không về các khối D/A theo kỹ thuật B–spline bậc lẻ?.
2. Nếu áp dụng phương pháp nội suy dãy tín hiệu đo được $\{x_k\}$ thì khoảng tần số nhận dạng $G(j\omega)$ sẽ được mở rộng. Ngược lại phương pháp ngoại suy $\{x_k\}$ lại có tác dụng tăng độ phân giải, tức là tăng số các giá trị $G(k\Omega_N)$ tính được trong khoảng tần số cố định (giảm Ω_N). Ở những trường hợp nào, khi áp dụng cả hai phương pháp nội và ngoại suy dãy $\{x_k\}$ mà lại không làm tăng độ phân giải của kết quả?.
3. Chứng minh rằng ảnh Fourier $X(j\omega)$ của một hàm chẵn $x(t)$ là một hàm thực.
4. Chứng minh rằng ảnh Fourier $X(j\omega)$ của một hàm lẻ $x(t)$ là một hàm thuần ảo.

Tài liệu tham khảo

[1] Allen, J. B.
Short–Term spectral Analysis, Synthesis and Modification by Discrete Fourier Transform. IEEE Transaction on ASSP, 25(1977)3, pp. 235–238.

[2] Bergland, G.D.
A guided Tour of the Fast Fourier Transform. IEEE Spectrum, 6(1969)7, pp. 41–52.

[3] Brigham, E.O.
Fast Fourier Transform. Verlag R.Oldenburg, München, Wien,1987.

[4] Eykhoff,P.
System Identification. London–Wilay, 1974.

[5] Geckinli,N.C. ; Yavus,D.
Some novel Windows and Concise Tutorial Comparion of Window Families. IEEE Transaction on ASSP, 26(1978)6, pp. 501–507.

[6] Gantmacher, F.R.
Matrizentheorie. Springer Verlag, 1966.

[7] Horowitz,L.L.
The Effects of Spline Interpolation on Power spectral Estimation. IEEE Transaction on ASSP, 22(1974)1, pp. 22–27.

[8] Isermann, R.
Identifikation dynamischer Systeme. Springer Verlag, 1994.

[9] Lutz,H. ; Wendt, W.
Taschenbuch der Regelungstechnik. Harri Deutsch Verlag, 1998.

[10] Marple, S.L.
Digital Spectral Analysis with Application. Prentice Hall, 1993.

[11] Morf, M. ; Dickimson, B. ; Kalailath, T. ; Viera, A.
Efficient Solution of Covariance Equation for linear Prediction. IEEE Transaction on ASSP, 25(1977)3, pp. 429–433.

[12] Phước, N.D.
Identifikation dynamischer Systeme mittels Spectralanalysis. Dissertation, 1994.

[13] Phước, N.D. & Minh, P.X.
Điều khiển tối ưu và bền vững. Nhà xuất bản Khoa học và Kỹ thuật, 1999.

[14] Phước, N.D. & Minh, P.X.
Hệ phi tuyến. Nhà xuất bản Khoa học và Kỹ thuật, 1999.

[15] Papoulis, A.
The Fourier intergral and its Application. McGraw Hill book company, 1962.

[16] Rabiner, L.R. ; Allen, J.B.
Short Times Fourrier Analysis Techniques for FIR System Identification and Power Spectrum Estimation. IEEE Transaction on ASSP, 27(1979)2, pp. 182–192.

[17] Wernstedt, J.
Experimentelle Prozeßanalyse. VEB Verlag Technik, 1989.

[18] Wu, N.
An explicit solution and data extention in the maximum entropie method. IEEE Trans. on ASSP, 35(1987)9, pp.486–491.