HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÀI GIẢNG

MÔ PHỎNG HỆ THỐNG TRUYỀN THÔNG

KHOA VIỄN THÔNG 1

Tác giả: TS. Nguyễn Đức Nhân

HÀ NỘI 07-2014

LỜI MỞ ĐẦU

Trong sự phát triển nhanh chóng các hệ thống viễn thông cùng với sự phát triển mạnh mẽ của các hệ thống máy tính, ngày nay mô phỏng đã trở thành một công cụ không thể thiếu trong quá trình nghiên cứu phát triển hệ thống cũng như trong quá trình thiết kế và triển khai hệ thống. Do vậy cuốn bài giảng "Mô phỏng hệ thống truyền thông" được viết nhằm cung cấp cho sinh viên những kiến thức bổ ích liên quan đến môn học. Tài liệu gồm 6 chương với các nội dung cơ bản như sau:

- Chương 1: Trình bày tổng quan về kỹ thuật mô phỏng bao gồm phương pháp luận, các vấn đề về mô hình hóa và vai trò của mô phỏng trong thiết kế hệ thống.
- Chương 2: Giới thiệu về MATLAB giúp sinh viên nắm được vấn đề cơ bản trong việc sử dụng MATLAB làm công cụ tính toán trong kỹ thuật nói chung và trong mô phỏng hệ thống truyền thông ở các chương sau.
- Chương 3: Giới thiệu về Simulink, một công cụ trong MATLAB được sử dụng để mô phỏng dựa trên mô hình hay sơ đồ khối, thuận tiện cho việc mô phỏng hệ thống.
- Chương 4: Mô phỏng quá trình thu phát tín hiệu bao gồm các vấn đề cơ bản về tín hiệu trong mô phỏng, mô phỏng các quá trình cơ bản thực hiện tại bộ phát và bộ thu trong một hệ thống truyền thông.
- Chương 5: Mô phỏng kênh thông tin bao gồm phương pháp thực hiện để mô phỏng các kênh cơ bản nói chung đến các kênh thông tin cụ thể như kênh hữu tuyến và kênh vô tuyến.
- Chương 6: Ước tính tham số và hiệu năng hệ thống giúp sinh viên nắm được các vấn đề cơ bản về ước tính trong thống kê, đặc biệt là trong đánh giá hiệu năng hệ thống.

Chúng tôi hy vọng rằng cuốn bài giảng sẽ là tài liệu tham khảo hữu ích cho sinh viên chuyên ngành viễn thông và những người quan tâm. Tuy nhiên để giúp sinh viên nắm bắt những vấn đề cơ bản nhất của kỹ thuật mô phỏng hệ thống truyền thông đòi hỏi người học phải có những kiến thức tổng hợp của các môn học cơ sở khác mà không phải thuộc mục tiêu và nội dung chính của môn học này. Đây cũng là phiên bản đầu tiên được viết trong thời gian ngắn do vậy sẽ không tránh khỏi những sai sót. Chúng tôi rất mong nhận được ý kiến đóng góp của các quí thầy cô, các bạn sinh viên và những người quan tâm để hoàn thiện hơn cuốn tài liệu này.

Tác giả

TS. Nguyễn Đức Nhân

MỤC LỤC

DANH SÁCH THUẬT NGỮ VIẾT TẮT

	Thuật ngữ tiếng Anh	Thuật ngữ tiếng Việt
A		
ADC	Analog to Digital Conversion	Chuyển đổi tín hiệu tương tự sang số
AM	Amplitude Modulation	Điều chế biên độ
AMI	Alternate Mark Inversion	Đảo dấu mã
ASK	Amplitude Shift Keying	Khóa dịch biên độ
AWGN	Additive White Gaussian Noise	Nhiễu Gauss trắng cộng
B		
BER	Bit Errors Rate	Tốc độ lỗi
D		
DFT	Discrete Fourier Transform	Khai triển Fourier rời rạc
DPSK	Differential Phase Shift Keying	Khóa dịch pha vi sai
DSB	Double Side Band	Điều chế biên kép
DSP	Digital Signal Processing	Xử lý tín hiệu số
E		
erp	equivalent random process	Quá trình ngẫu nhiên tương đương
F		
FIR	Finite Impulse Response	Đáp ứng xung hữu hạn
FM	Frequency Modulation	Điều chế tần số
FSK	Frequency Shift Keying	Khóa dịch tần
H		
HDL	Hardware Description Language	Ngôn ngữ đặc tả phần cứng
HMM	Hiden Markov Model	Mô hình Markov ẩn
I		
I/O	Input/Output	Vào/Ra
IDFT	Inverse Discrete Fourier Transform	Khai triển Fourier rời rạc đảo
IIR	Infinite Impulse Response	Đáp ứng xung vô hạn
IS	Important Sampling	Lấy mẫu quan trọng
ISI	Inter-Symbol Interference	Giao thoa giữa các ký hiệu
N		
NRZ	Non-Return-to-Zero	Không trở về không
O		
OFDM	Orthogonal Frequency Division	Ghép kênh phân chia theo tần

	Multiplexing	số trực giao
P		
PCM	Pulse Code Modulation	Điều chế xung mã
PET	Performance Evaluation Technique	Kỹ thuật ước tính hiệu năng
PLL	Phase Locked Loop	Mạch vòng khóa pha
PM	Phase Modulation	Điều chế pha
PSD	Power Spectral Density	Mật độ phổ công suất
PSK	Phase Shift Keying	Khóa dịch pha
Q		
QA	Quasi-Analytical	Bán giải tích
QAM	Quadrature Amplitude Modulation	Điều chế biên độ cầu phương
QPSK	Quadrature Phase Shift Keying	Khóa dịch pha cầu phương
R		
RNG	Random Number Generator	Bộ tạo số ngẫu nhiên
RZ	Return-to-Zero	Trở về không
S		
SER	Symbol Error Rate	Tốc độ lỗi ký hiệu
SNR	Signal to Noise Ratio	Tỉ số tín hiệu trên nhiễu
SSB	Single Side Band	Điều chế đơn biên
T		
TDL	Tributary Delay Line	Đường trễ nhánh
V		
VSB	Vestigial Side Band	Điều chế rớt biên

Chương 1 Tổng quan về kỹ thuật mô phỏng

Mô phỏng ngày này có mặt trong nhiều lĩnh vực khoa học kỹ thuật đặc biệt trong thiết kế hệ thống truyền thông. Nội dung chương này sẽ trình bày những vấn đề tổng quan và cơ bản nhất trong kỹ thuật mô phỏng nói chung và trong mô phỏng hệ thống truyền thông nói riêng.

1.1 Giới thiệu chung

Trong những thập kỷ qua các hệ thống truyền thông và kỹ thuật xử lý tín hiệu ngày càng tăng nhanh chóng về mức độ phức tạp. Trong suốt thời gian này sự nổi lên các loại công nghệ mới liên quan đến phần cứng tốc độ cao và chi phí rẻ hơn trong xử lý tín hiệu số, công nghệ quang sợi, các linh kiện mạch tích hợp đã có tác động mạnh mẽ đến việc triển khai các hệ thống truyền thông. Trong khi sự phát triển về mức độ phức tạp của hệ thống truyền thông tăng lên theo thời gian và nỗ lực đòi hỏi cho quá trình phân tích và thiết kế, thì sự cần thiết để đưa các công nghệ mới vào các sản phẩm thương mại nhanh chóng cũng đòi hỏi rằng thiết kế được hoàn thành đúng thời gian, hiệu quả về chi phí và không mất nhiều công sức. Các nhu cầu này có thể được đáp ứng chỉ bằng cách sử dụng các công cụ thiết kế và phân tích mạnh mẽ được trợ giúp bởi máy tính.

Một loạt các kỹ thuật trợ giúp bởi máy tính đã được phát triển trong nhiều năm qua để hỗ trợ trong quá trình mô hình hóa, phân tích và thiết kế các hệ thống truyền thông. Các kỹ thuật được trợ giúp bởi máy tính này nằm ở hai loại cơ bản: tiếp cận dựa trên công thức mà ở đó máy tính được sử dụng để ước tính các công thức phức tạp và tiếp cận dựa trên mô phỏng mà ở đó máy tính được sử dụng để mô phỏng các dạng sóng hoặc tín hiệu truyền qua hệ thống.

Hiệu năng của các hệ thống truyền thông có thể được đánh giá bằng việc sử dụng các tính toán dựa trên công thức, mô phỏng dạng sóng hoặc bằng đo kiểm và chế tạo mẫu thử.

Các kỹ thuật dựa trên công thức dựa vào các mô hình được đơn giản hóa cung cấp cái nhìn sâu về mối quan hệ giữa các tham số thiết kế và hiệu năng hệ

thống và chúng rất hữu ích trong các giai đoạn đầu của quá trình thiết kế cho việc khám phá không gian thiết kế mở rộng. Tuy nhiên ngoại trừ các trường hợp quá đơn giản và lý tưởng hóa thì nó rất khó để đánh giá hiệu năng của các hệ thống truyền thông phức tạp chỉ bằng các kỹ thuật giải tích với độ chính xác cần cho quá trình thiết kế chi tiết hơn.

Đánh giá hiệu năng dựa trên các phép đo kiểm thu được từ các mẫu thiết kế thử phần cứng tất nhiên là một phương pháp chính xác và đáng tin cậy, hữu ích trong các giai đoạn sau của thiết kế khi các lựa chọn thiết kế được giới hạn trong một tập nhỏ. Tiếp cận này nhìn chung là rất tốn kém và mất nhiều thời gian và không linh hoạt. Nó rõ ràng là không khả thi để sử dụng tiếp cận này trong giai đoạn đầu chu trình thiết kế khi số lượng các lựa chọn thiết kế lớn.

Băng tiếp cận dựa trên mô phỏng để đánh giá hiệu năng, các hệ thống có thể được mô hình hóa gần như ở bất kỳ mức chi tiết mong muốn nào và không gian thiết kế có thể được khai thác cụ thể và chi tiết hơn bằng các tiếp cận dựa trên công thức hoặc đo kiểm. Dựa trên mô phỏng ta cũng có thể kết hợp các mô hình thực nghiệm và toán học một cách dễ dàng và kết hợp các đặc tính đo được của các linh kiện và các tín hiệu thực vào trong phân tích và thiết kế. Các dạng sóng được mô phỏng cũng có thể được sử dụng như là tín hiệu đo thử cho việc kiểm tra chức năng hoạt động của phần cứng.

Thực sự, một tiếp cận dựa trên mô phỏng có thể được sử dụng để tạo ra một môi trường chế tạo mẫu thử nhanh chóng cho việc phân tích và thiết kế các hệ thống xử lý tín hiệu và truyền thông, một môi trường trong đó các mô hình phần mềm có thể được kết hợp với dữ liệu phần cứng và các tín hiệu thực để tạo ra các mẫu thiết kế không lỗi tiết kiệm chi phí và thời gian.

Nhược điểm cơ bản của tiếp cận mô phỏng là tải tính toán lớn cái có thể được giảm thiểu bằng sự lựa chọn cẩn thận các kỹ thuật mô hình hóa và mô phỏng.

1.2 Phương pháp luận mô phỏng

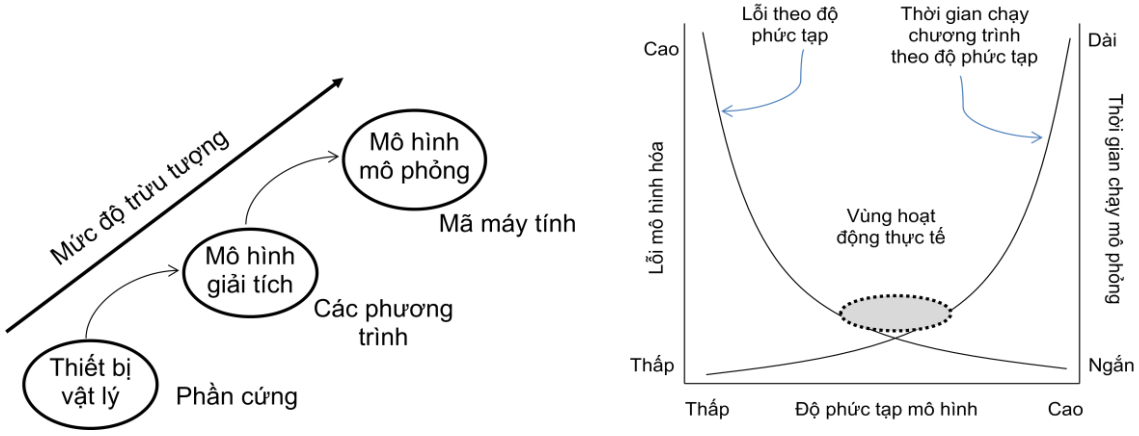
1.2.1 Mô hình hóa bài toán

Mục đích cuối cùng của mô phỏng là phỏng tạo lại các đặc tính của một hệ thống vật lý thực tế thông qua tính toán dựa trên máy tính. Một bài toán mô phỏng đơn giản nhất thường bao gồm bốn bước cơ bản sau:

- Ánh xạ một bài toán đã cho thành một mô hình mô phỏng, đâycũng là bước quan trọng nhất của bài toán mô phỏng. Bước này có thể được xem là bước mô hình hóa để chuyển một mô hình vật lý thực thành mô hình toán học.
- Phân giải bài toán tổng thể thành một tập các bài toán nhỏ hơn. Việc phân chia này cũng là cách để đơn giản hóa bài toán xây dựng đảm bảo tính khả thi trong tính toán.
- Lựa chọn một tập các kỹ thuật phù hợp về mô hình hóa, mô phỏng và ước tính và áp dụng chúng để giải các bài toán con đã được phân chia. Các kỹ thuật giải được sử dụng để tính toán tìm nghiệm thông qua hệ thống máy tính.
- Kết hợp các kết quả của các bài toán con để cung cấp nghiệm cho bài toán tổng thể xác định ban đầu.

Một hệ thống truyền thông thực tế nhìn chung quá phức tạp để có thể mô tả và mô phỏng nó một cách toàn bộ. Do vậy nó rất cần thiết để đơn giản hóa một số mặt của bài toán mô phỏng giúp dễ dàng hơn cho việc tính toán. Ngoài việc phân chia bài toán tổng thể thành các bài toán nhỏ hơn, thì việc chuyển từ bài toán lớn hơn thành dạng đơn giản hơn được xem như là thực hiện thí nghiệm có điều kiện là cũng cần thiết. Xét dạng sóng đầu ra V_t của một hệ thống tại thời điểm rời rạc t theo dạng $V_t = g(\Omega)$, trong đó g là hàm truyền của hệ thống và $\Omega = (z_1, z_2, \ldots, z_K)$ là tập các quá trình đầu vào (rời rạc thời gian). Chức năng của một mô phỏng nói chung là để tạo ra một chuỗi giá trị $\{V_t\}$ đối với $t = kT_s$, $k = \pm1, \pm2,\ldots$, với T_s là chu kỳ lấy mẫu. Chuỗi này sẽ được xử lý theo kiểu để thu được đại lượng hiệu năng hoặc thông tin phù hợp khác. Một thí nghiệm có điều kiện sẽ tạo ra $V_t = g(\Omega')$ trong đó $\Omega' = (z_1, \ldots, z_k, z_{k+1} = \xi_{k+1}, \ldots, z_K = \xi_K)$. Đó là k quá trình đầu tiên được mô phỏng trong khi các quá trình còn lại được duy trì tại các giá trị cố định để tạo ra thí nghiệm đơn giản hơn. Các giá trị này có thể được đặt bằng 0 tương đương với việc bỏ qua các quá trình này. Một dạng điều kiện khác là đơn giản hóa bản thân hệ thống. Sự đơn giản hóa này có thể bao gồm một sự mô tả một hoạt động với độ phức tạp được rút gọn hoặc bỏ quả hoàn toàn một hoặc nhiều hoạt động. Kết hợp đặt điều kiện lên hệ thống và lên các quá trình đầu vào, hệ thống được đơn giản hóa bởi g', thí nghiệm mô phỏng được mô tả bởi $V_t = g'(\Omega')$. Mục đích cuối cùng của quá trình đơn giản hóa là đảm bảo khả năng tính toán trong quá trình mô phỏng.

Tuy nhiên quá trình này được thực hiện bằng các phép gần đúng cũng có nghĩa rằng có sự sai lệch giữa mô hình và hệ thống thực tế.



Hình 1-1 (a) Thiết bị vật lý và các mô hình, (b) Các ảnh hưởng độ phức tạp của mô hình.

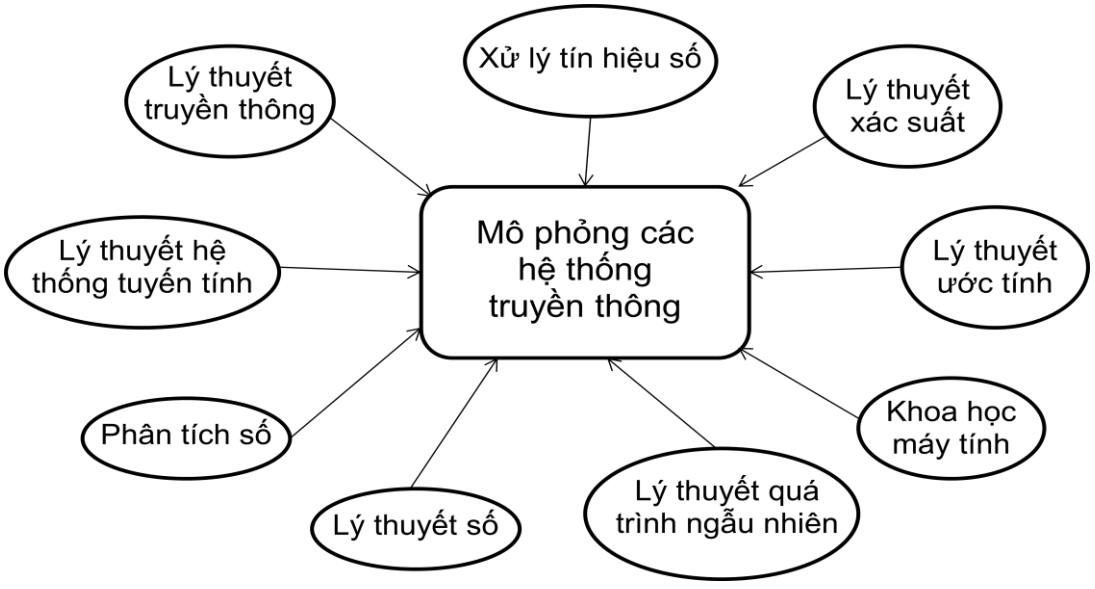
Có hai kiểu mô hình mà ta cần xem xét: mô hình giải tích và mô hình mô phỏng và cả hai đều là sự trừu tượng của một linh kiện hay hệ thống vật lý như cho thấy trong hình 1-1a. Linh kiện vật lý có thể là một phần tử mạch như một điện trở hoặc một phân hệ như một mạch vòng khóa pha, mà nó cũng có thể là một hệ thống truyền thông. Bước đầu tiên và cũng là quan trọng nhất trong quá trình mô hình hóa là nhận ra được các thuộc tính và các đặc tính hoạt động của thiết bị vật lý được mô tả trong mô hình. Các mô hình giải tích điển hình thuộc các dạng phương trình hoặc hệ phương trình xác định quan hệ đầu vào – đầu ra của thiết bị vật lý. Các phương trình này chỉ là sự mô tả một phần của thiết bị được mô hình hóa vì chỉ một số mặt xác định của thiết bị được mô hình hóa. Thêm nữa các phương trình định nghĩa thiết bị thường chỉ chính xác trên một dải giới hạn các tham số (điện áp, dòng, tần số). Mô hình mô phỏng luôn là một tập các giải thuật thực hiện tính toán nghiệm số của các phương trình định nghĩa mô hình giải tích. Các kỹ thuật phân tích số và xử lý tín hiệu số là những công cụ được sử dụng để phát triển các giải thuật này. Các mô hình có các mức trừu tượng khác nhau và tăng lên khi dịch chuyển từ thiết bị vật lý tới mô hình giải tích và cuối cùng là mô hình mô phỏng. Sự tăng mức trừu tượng là kết quả của các giả định và các gần đúng được đưa ra khi đi từ thiết bị vật lý lên mô hình giải tích và mô hình mô phỏng.

Một vấn đề cần lưu ý trong bài toán mô phỏng là sử ảnh hưởng của mức độ phức tạp của mô hình. Các mô hình đơn giản cho phép thực hiện nhanh hơn các mô hình phức tạp hơn. Tuy nhiên các mô hình đơn giản có thể không mô tả một cách đày đủ các thuộc tính quan trọng của thiết bị do đó mô phỏng có thể thu được kết quả có sai số lớn. Như vậy có sự bù trừ giữa độ chính xác của mô hình và thời gian

chạy mô phỏng như cho thấy trong hình 1-1b. Một mô phỏng trong thực tế được thiết kế tốt đảm bảo vùng hoạt động tối ưu cung cấp độ chính xác và tốc độ thực hiện hợp lý. Tuy nhiên tùy thuộc vào mục đích mô phỏng mà trong một số trường hợp đòi hỏi mức độ chính xác cao, nói cách khác mức độ phức tạp mô hình phải đủ để đáp ứng yêu cầu, do vậy thời gian chay mô phỏng lâu là khó tránh khỏi.

1.2.2 Tính đa mặt trong mô phỏng

Trước những năm 1970 bài toán mô phỏng thường được giải quyết theo cách thiên về dạng đặc biệt. Phương pháp luận để phát triển mô phỏng và các nguồn lỗi xuất hiện trong mọi chương trình mô phỏng không được hiểu một cách đầy đủ. Hơn 20 năm qua, cộng đồng nghiên cứu đã tạo ra một khối lượng lớn kiến thức, tạo ra phương pháp luận để phát triển mô phỏng cũng như việc thống nhất về lý thuyết để giải quyết nhiều vấn đề nảy sinh trong quá trình triển khai chương trình mô phỏng. Theo đó, việc dùng mô phỏng như là công cụ phân tích cần thiết để hiểu và hiểu sâu sắc nhằm triển khai mô phỏng có độ tin cậy. Xây dựng lớn khối lượng kiến thức này đòi hỏi phải tích hợp từ nhiều kiến thức trong nhiều lĩnh vực khác nhau. Mặc dù chưa được thấu đáo nhưng 9 lĩnh vực nghiên cứu quan trọng ảnh hưởng đến quá trình nghiên cứu về mô phỏng được mô tả ở hình 1-2. Ta xét ngắn gọn 9 lĩnh vực này nhằm rõ hơn về mối quan hệ của chúng với khoa học mô phỏng.



Hình 1-2 Các lĩnh vực ảnh hưởng lên nghiên cứu mô phỏng các hệ thống truyền thông

Các khái niệm về lý thuyết hệ thống tuyến tính cho ta các kỹ thuật để xác định các quan hệ vào/ ra của hệ thống tuyến tính, cho phép trình bày mô hình hệ

thống trong miền thời gian ở dạng hàm đáp ứng xung kim hệ thống và miền tần số ở dạng hàm truyền đạt hệ thống cũng như việc xây dựng nền tảng cho nhiều vấn đề.

Hiển nhiên, kiến thức lý thuyết truyền thông là rất quan trọng. Cấu trúc hệ thống, đặc tính hoạt động của các phân hệ (bộ giải điều chế, bộ cân bằng, chi tiết hóa các mô hình kênh…) phải được hiểu rõ trước khi triển khai mô phỏng. Khi sử dụng mô phỏng để xác định các giá trị của tham số hệ thống, cần phải lưu ý đến dải giá trị của nó có ý nghĩa thực tế trước khi triển khai mô phỏng. Cần phải có những hiểu biết sâu sắc về đặc tính hệ thống để đảm bảo hoạt động mô phỏng chính xác và kết quả hợp lý.

Các công cụ của xử lý tín hiệu số (DSP) được dùng để triển khai các giải thuật, từ đó xây dựng mô hình mô phỏng hệ thống truyền thông. Mô hình mô phỏng này thường bao gồm một số phép lấy xấp xỉ rời rạc của các phần tử hệ thống liên tục, do vậy cần có kiến thức về xử lý tín hiệu số để hiểu và đánh giá bản chất của các phép lấy xấp xỉ này. Thực tế, mỗi khối chức năng trong mô hình mô phỏng là một hoạt động DSP, vì vậy các công cụ của DSP cho ta các kỹ thuật thực hiện mô phỏng.

Giải tích số có quan hệ chặt chẽ với DSP, nhưng được đề cập tách biệt vì nó là phần kiến thức cũ hơn. Nhiều kỹ thuật kinh điển như phân tích số, nội suy đa thức, kỹ thuật fit đồ thị đều có nguồn gốc trong giải tích số.

Các khái niệm về xác suất cũng là nền tảng căn bản cho mô phỏng. Việc đánh giá hiệu năng hệ thống truyền thông thường được biểu diễn trong các thuật ngữ xác suất. Ví dụ khi đề cập xác suất lỗi bit hay xác suất lỗi ký hiệu trong hệ thống truyền thông số; khi xét bài toán đồng bộ, ta quan tâm xác suất lỗi pha vượt quá một mức cho trước. Lý thuyết xác suất cơ bản cho ta khái niệm về biến ngẫu nhiên và hàm mật độ xác suất. Kiến thức về hàm mật độ xác suất cho phép tính toán các đại lượng như đã đề cập phần trên. Kết quả của mô phỏng thường là một biến ngẫu nhiên và phương sai của biến ngẫu nhiên đó là một đại lượng đo độ chính xác thống kê của mô phỏng.

Trong nhiều trường hợp, các dạng sóng tín hiệu và tạp âm được xử lý bởi mô phỏng được coi là các hàm mẫu của một quá trình ngẫu nhiên. Sự phát triển các thuật toán để tạo dạng sóng có các thuộc tính thống kê phù hợp sẽ đòi hỏi kiến thức quá trình ngẫu nhiên cơ bản. Lý thuyết quá trình ngẫu nhiên cho ta các công cụ để mô tả các quá trình này trong miền thời gian (hàm tự tương quan), và trong miền tần

số (mật đổ phổ công suất). Nhiều ứng dụng khác của lý thuyết quá trình ngẫu nhiên cũng sẽ được đề cập trong nội dung bài giảng.

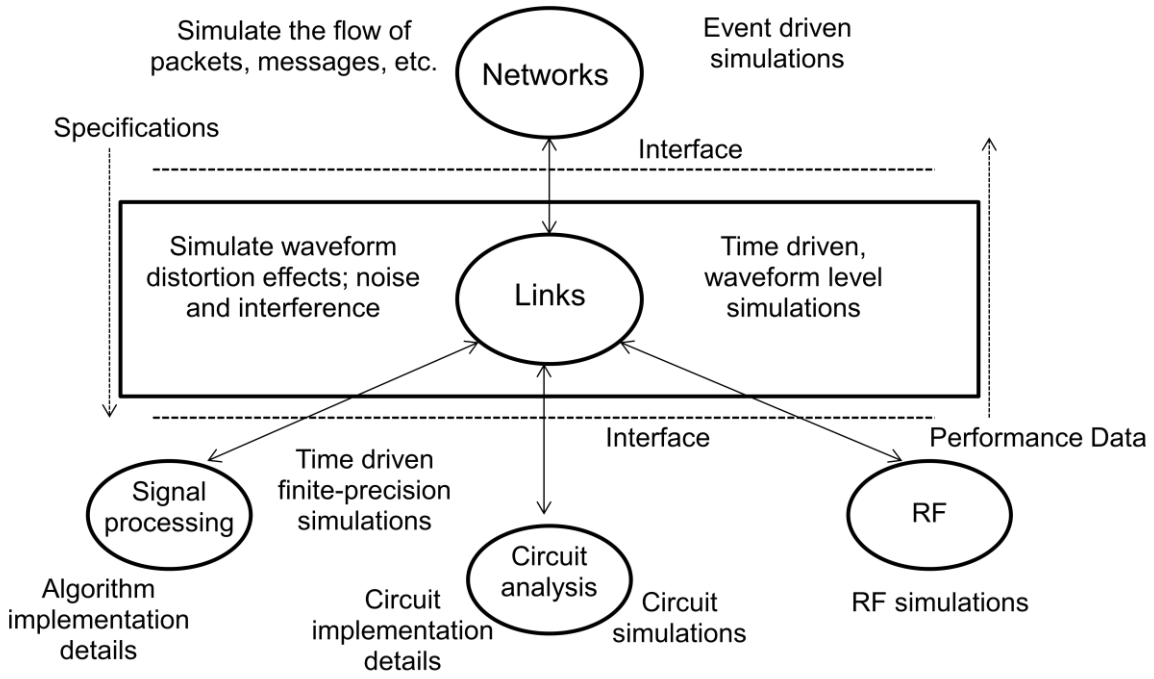
Một vài khái niệm cơ bản về lý thuyết số cung cấp các công cụ để triển khai các bộ tạo số ngẫu nhiên. Các bộ tạo số ngẫu nhiên này là các khối cơ bản của bộ tạo dạng sóng để biểu diễn các chuỗi số, dạng sóng tạp âm, pha đinh tín hiệu, nhiễu ngẫu nhiên.

Khái niệm cơ bản về khoa học máy tính cũng sẽ có ích trong mô phỏng. Ví dụ như độ dài từ mã, định dạng từ mã được dùng để biểu diễn các mẫu tín hiệu sẽ ảnh hưởng đến tính chính xác của mô phỏng. Việc chọn ngôn ngữ lập trình cũng quan trọng khi triển khai các bộ mô phỏng thương mại. Bộ nhớ khả dụng, tổ chức bộ nhớ sẽ ảnh hưởng cách thức dữ liệu và các lệnh được chuyển qua giữa các phần tử của mô phỏng. Các yêu cầu và năng lực về đồ họa sẽ xác định dạng sóng được hiển thị như thế nào và sẽ ảnh hưởng quá trình truyền tải mã chương trình mô phỏng từ máy tính này đến máy tính khác.

Các công cụ và khái niệm về lý thuyết ước tính cho phép định lượng tính hiệu quả của kết quả mô phỏng. Như đã đề cập, kết quả mô phỏng ngẫu nhiên là một biến ngẫu nhiên. Mỗi khi thực hiện mô phỏng sẽ tạo ra một giá trị của biến ngẫu nhiên đó và biến ngẫu nhiên này tạo thành bộ ước tính cho đại lượng cần được ước tính. Một cách tổng quát, lý thuyết ước tính cho ta các công cụ giải tích cần thiết để đánh giá mức độ khả tin của các kết quả mô phỏng.

1.3 Các khái niệm cơ bản về mô hình hóa

Theo nghĩa rộng, thuật ngữ “hệ thống truyền thông” ám chỉ đến một mạng truyền thông toàn cầu, hệ thống vệ tinh địa tĩnh, hệ thống truyền dẫn quang hoặc một modem tích hợp sẵn trong một máy tính cá nhân. Một cách nhìn phân cấp thường được sử dụng để mô tả các hệ thống truyền thông như cho thấy trong hình 1-3. Mức đỉnh trong mô tả này là một mạng truyền thông được tạo bởi các nút mạng kết nối với nhau qua các tuyến thông tin hoặc các hệ thống truyền dẫn như được mô tả trong lớp dưới. Một tuyến truyền dẫn lại được hình thành từ các phần tử như các bộ điều chế, các bộ mã hóa, các bộ lọc, các bộ khuyếch đại và các thành phần khác thực hiện các hoạt động xử lý tín hiệu. Các phần tử này có thể là các mạch điện tương tự, các mạch số hoặc một thuật toán thực thi trên một bộ xử lý tín hiệu số (DSP) khả lập trình. Chi tiết của các phần tử này được mô tả ở lớp dưới cùng của phân cấp.



Hình 1-3 Cấu trúc phân cấp trong mô phỏng

Một loạt các kỹ thuật mô phỏng khác nhau được sử dụng để đánh giá hiệu năng của các lớp khác nhau. Tại mức mạng, luồng các gói và các bản tin trên mạng được mô phỏng bằng việc sử dụng một bộ mô phỏng các sự kiện rời rạc và các đại lượng hiệu năng như thông lượng mạng, thời gian đáp ứng và hiệu suất sử dụng tài nguyên được ước tính như là một hàm của các tham số mạng như tốc độ xử lý, kích cỡ bộ đệm tại nút mạng và dung lượng tuyến. Các mô phỏng mạng được sử dụng để thiết lập các đặc tính cho các bộ xử lý, các giao thức và các tuyến truyền dẫn.

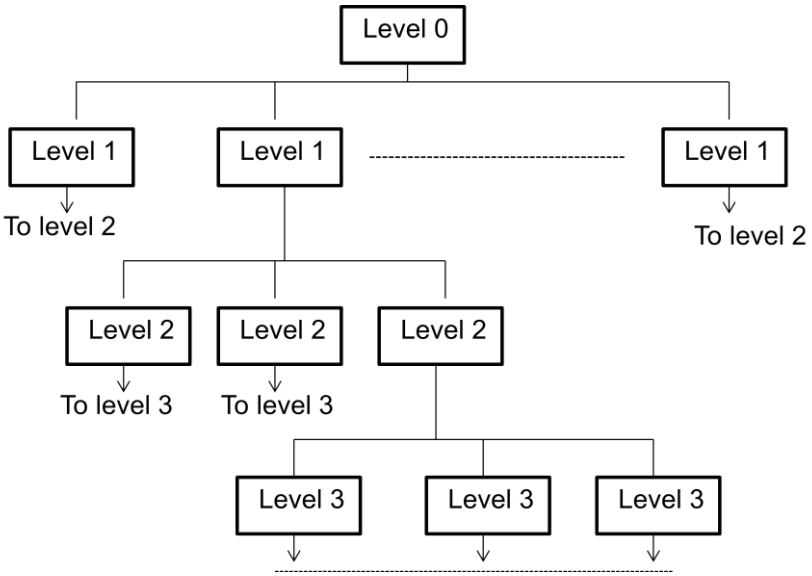
Các hệ thống truyền thông giải quyết việc truyền dẫn các dạng sóng mang thông tin trên các kiểu kênh thông tin khác nhau (không gian tự do, cáp đồng, sợi quang,,..). Đối với các hệ thống truyền dẫn số, hiệu năng của tuyến thông tin được đo theo đặc tính lỗi bít, và hiệu năng tốc độ lỗi thường được ước tính bằng kỹ thuật mô phỏng dạng sóng qua mô hình của các khối chức năng. Khác với mô phỏng mạng được sử dụng để thiết lập các đặc tính của tuyến, thì mô phỏng mức hệ thống được sử dụng để kiểm tra rằng thiết kế tuyến đáp ứng được các đặc tính này. Các tham số thu được từ mô phỏng mức hệ thống được chuyển sang cho bộ mô phỏng mức mạng để kiểm tra hiệu năng mạng.

Lớp dưới cùng trong hình 1-3 liên quan đến hoạt động của các thành phần như các bộ lọc và các bộ cân bằng sử dụng hoặc công nghệ tương tự hoặc công nghệ số. Các bộ mô phỏng mạch như Spice hoặc bộ mô phỏng số như HDL (Hardware Description Language) được sử dụng để mô phỏng, kiểm tra chức năng

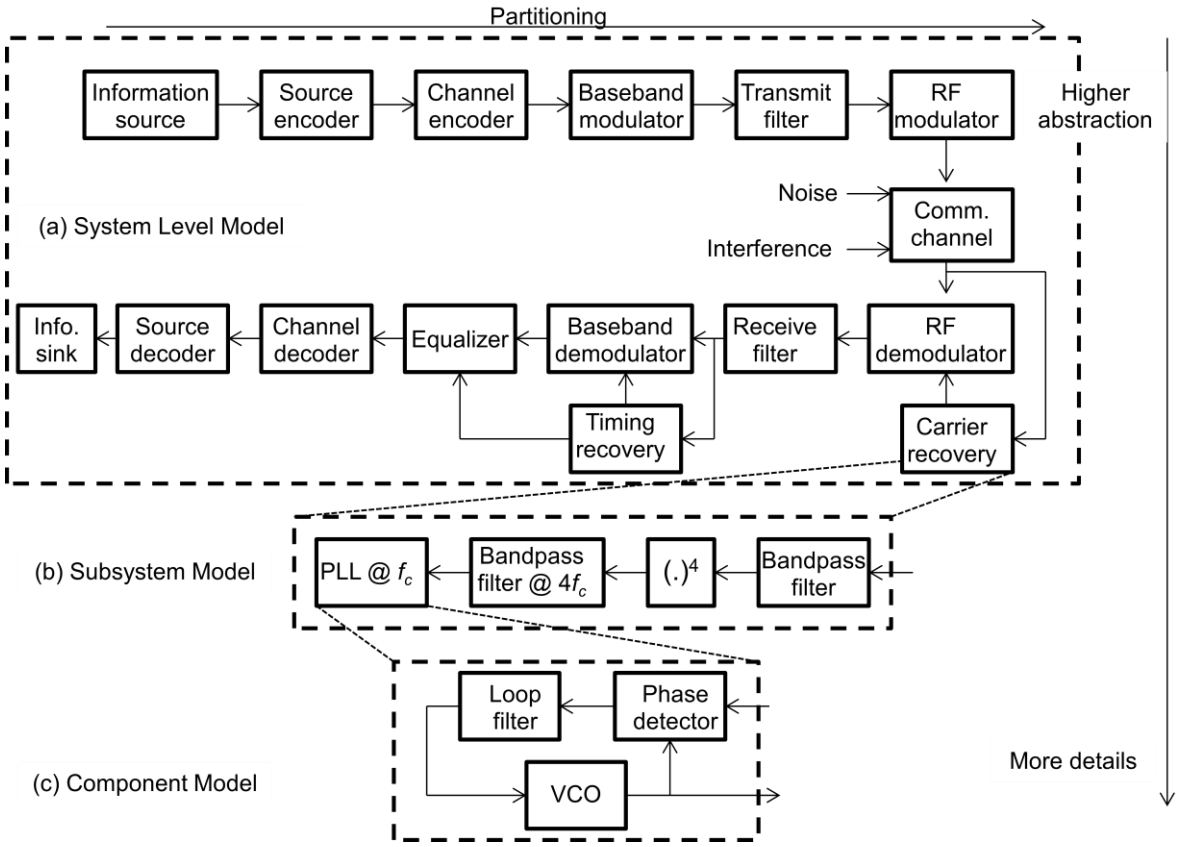
hoạt động và đặc tính của các thành phần linh kiện. Mô phỏng mức hệ thống thiết lập các đặc tính cho việc thực thi hoàn tất và mô phỏng ở mức mạch thực thi được sử dụng để cung cấp các mô hình đặc tính (ví dụ: hàm truyền đạt của một bộ lọc) cho mức hệ thống.

Trong phạm vi giới hạn nội dung, bài giảng này sẽ tập trung vào mô phỏng hệ thống truyền thông sử dụng kỹ thuật mô phỏng dạng sóng.

Nhìn chung về bản chất để mô hình càng chính xác thì mức độ mô tả nó càng chi tiết. Sự mô tả hệ thống được phân chia theo mức độ chi tiết khác nhau. Cách mô tả hệ thống một cách trực quan thường được thực hiện thông qua một sơ đồ khối. Mỗi khối trong sơ đồ có thể được triển khai và được mô tả bởi một sơ đồ các khối con khác kết nối với nhau trong bản thân khối đó. Quá trình này có thể tiếp tục được thực hiện cho đến khi không thể rút gọn xuống thành các khối con được nữa. Kết quả của việc khai triển sơ đồ khối có thể được biểu diễn trực quan qua sơ đồ dạng cây với các nhánh liên tiếp đặc trưng cho các mức độ chi tiết tăng dần lên như cho thấy trong hình 1-4. Cấu trúc này cũng cho thấy tính phân cấp trong việc thực thi phần mềm và cho cả việc quản lý mức độ phức tạp của mô hình hóa. Cây phân cấp tăng dần theo dạng lộn ngược với phía trên là các mô hình mức cao và đi dần xuống là các mô hình mức thấp. Mức độ chi tiết sẽ tăng dần từ trên xuống dưới. Một mô hình mức thấp được xem như là một phần được phân chia ra từ mô hình mức cao hơn bao hàm một sự mô tả sát hơn với mức vật lý.



Hình 1-4 Cấu trúc phân cấp trong mô hình hóa



Hình 1-5 Mô tả mô hình hóa phân cấp hệ thống truyền thông.

Sự phân cấp có thể được minh chứng rõ như mô tả hệ thống truyền thông cho trong hình 1-5. Nó cũng cho thấy rõ rằng mức độ phức tạp tồn tại cả hai chiều: theo mức phân cấp (chiều dọc) và tại mỗi mức (chiều ngang). Mức trên cùng được xem là mức hệ thống gồm các khối chức năng thực hiện một xử lý xác định trong hệ thống. Mỗi khối chức năng ở mức hệ thống cũng có thể được mô tả chi tiết hơn ở mức dưới gọi là mức phân hệ. Quá trình phân chia chi tiết có thể tiếp tục xuống mức các thành phần linh kiện. Tương ứng với cấu trúc phân cấp này bài toán mô hình hóa cũng được phân chia thành ba kiểu mô hình hóa đó là: mô hình hóa hệ thống, mô hình hóa các thành phần linh kiện và mô hình hóa quá trình.

1.3.1 Mô hình hóa hệ thống

Một hệ thống ở đây là một tuyến truyền dẫn được mô tả ở mức cao nhất bằng sơ đồ khối kết nối các phân hệ hay hệ thống con. Vấn đề mô hình hóa hệ thống là một vấn đề về cấu hình theo nghĩa sơ đồ khối mô phỏng và càng sát với thực tế thì mô hình hệ thống càng chính xác. Như đã đề cập, mô hình mức cao nhất có thể cần được sử dụng cho mô phỏng để giảm tải tính toán. Tuy nhiên, ở bất kỳ mức nào trong cây phân cấp đều có thể giảm mức độ phức tạp mô hình hóa bằng cách chỉ sử

dụng một tập con các khối tại mức đó. Đây là dạng rút gọn mức độ phức tạp thường được sử dụng ở mức mô hình hóa hệ thống, điều này muốn nói rằng một số các phân hệ có thể được bỏ qua khỏi mô phỏng hoặc được mô tả theo kiểu đơn giản hóa. Ví dụ trong một số trường hợp mô phỏng hệ thống như mô tả trong hình 1-5 các khối mã hóa nguồn thực hiện chuyển đổi tín hiệu tương tự sang số (ADC) có thể được bỏ qua mà sử dụng luôn nguồn tín hiệu số đầu vào.

Như vậy nói chung điều đáng mong muốn là mô phỏng sơ đồ khối được rút gọn nhiều nhất có thể từ quan điểm tính toán hiệu quả và việc rút gọn như vậy có thể hoàn toàn chấp nhận được trong nhiều trường hợp. Nói cách khác mức độ gần đúng trong một số trường hợp mô phỏng là không thể tránh khỏi.

1.3.2 Mô hình hóa thành phần chức năng

Một thành phần linh kiện ở đây đơn giản là một khối chức năng ở mức phân hệ có những tính chất mà nhà thiết kế hệ thống mong muốn. Từ quan điểm tính toán, mô hình linh kiện lý tưởng có thể diễn tả duy nhất ở mức phân hệ của cây cấu trúc mô hình hóa. Kiểu mô tả mô hình các thành phần linh kiện có thể được thể hiện qua một phương trình, một tập phương trình, một thuật toán hoặc một bảng tra cứu dữ liệu (lookup table). Tuy nhiên trong mô phỏng mức hệ thống, các khối thành phần có thể được mô tả đơn giản bằng một hàm truyền đạt. Có thể ví dụ như mạch vòng khóa pha (PLL) trong khối khôi phục sóng mang có thể được mô tả bằng một phương trình vi phân bậc hai. Một ví dụ khác như nguồn laser trong bộ phát quang có thể được đặc trưng bởi các phương trình tốc độ là một hệ phương trình vi phân.

Một mô hình các thành phần linh kiện tốt cho phép khảo sát chi tiết các đặc tính của các thành phần linh kiện phụ thuộc vào tất cả các tham số ảnh hưởng. Kết quả khảo sát có thể sẽ được sử dụng để xem xét các ảnh hưởng của các thành phần linh kiện trong mô phỏng mức hệ thống.

1.3.3 Mô hình hóa quá trình ngẫu nhiên

Một điều rõ ràng rằng các tín hiệu đầu vào và đầu ra của các hệ thống và các phân hệ là các quá trình ngẫu nhiên mong muốn (thông tin) và không mong muốn (nhiễu và giao thoa) và mục đích cơ bản của mô phỏng là để tính toán mức độ đảm bảo chất lượng tín hiệu mong muốn. Do vậy mức độ trung thực của tính toán này phụ thuộc vào mức độ các quá trình mô phỏng có thể sao chép các tính chất của các quá trình thực. Nhiệm vụ mô hình hóa nhìn chung là để bắt chước một quá trình ngẫu nhiên tại nguồn sinh ra nó, nên nếu ta có các mô hình các khối chức năng tốt

thì các quá trình này sẽ tác động lên tín hiệu đầu vào để sinh ra tín hiệu đầu ra chính xác và hợp lý. Việc sao chép quá trình ngẫu nhiên được thực hiện bởi một bộ tạo số ngẫu nhiên trong quá trình mô phỏng.

Mặc dù các nguồn tin và các nguồn nhiễu cả hai đều là các quá trình ngẫu nhiên trong vận hành, trong thiết kế và đo kiểm hệ thống các nguồn tín hiệu thường được sử dụng hoặc giả định là các tín hiệu đo thử thường có tính xác định. Ví dụ như một tín hiệu đo thử có thể là tín hiệu hình sin hoặc một chuỗi số có cấu trúc sẵn được tạo ra bởi thanh ghi dịch trong một kết nối cụ thể. Các chuỗi số này thường được gọi là các chuỗi giả ngẫu nhiên.

Có một kiểu quá trình ngẫu nhiên khác mà ta cần mô hình hóa nhưng không mô tả quá trình nhiễu hay thông tin, đó là một kênh ngẫu nhiên như kênh pha đinh. Ở đây bài toán mô hình hóa là đáp ứng xung của kênh thường được giả định biến đổi ngẫu nhiên theo thời gian.

Một dạng cấu trúc mô hình hóa khác là mô hình *quá trình ngẫu nhiên tương đương* (erp). Ý tưởng mô hình này như sau: giả sử rằng quá trình ngẫu nhiên đầu vào $x(t)$ đi qua n các khối hay phân hệ nối tiếp nhau và xuất hiện tại đầu ra là quá trình $y(t)$. Nếu bằng một số công cụ ta có thể suy ra một số đặc tính của quá trình ngẫu nhiên $y(t)$ thì toàn bộ quá trình xử lý xảy ra có thể được rút gọn hay đơn giản hóa bằng cách tìm và tạo ra một chuỗi ngẫu nhiên sao chép $y(t)$ mà không cần phải xử lý $x(t)$ qua n khối. Như vậy mô hình erp có thể giúp đơn giản hóa quá trình tính toán. Một ví dụ ứng dụng mô hình này là mô phỏng nhiễu pha trong hệ thống truyền thông.

1.3.4 Mô hình hóa hệ thống giả định

Trong một số trường hợp không phải tất cả các thực thể được mô hình hóa đều đã được xác định rõ ràng trước. Có thể hầu hết các hoạt động mô phỏng được thực hiện để trợ giúp quá trình thiết kế một hệ thống mà các đặc tính riêng biệt của hệ thống ban đầu không được biết và chỉ dần dần được xác định một cách rõ ràng hơn theo quá trình. Một hệ thống mới được thiết kế được xem như là một hệ thống giả định.

Một trong các mặt hữu ích của mô phỏng là khả năng ước tính hiệu năng của một hệ thống một cách chân thực trước khi nó được xây dựng trên thực tế. Do vậy điều mong muốn là có thể định hướng quá trình phát triển phần cứng đảm bảo một hiệu năng xác định. Vấn đề then chốt đối với quá trình này là hiện thực hóa sự ảnh

hưởng của bất kỳ thành phần thiết bị nào có thể được dự đoán chỉ bằng một số tham số quan trọng được lựa chọn tốt. Mặc dù đặc tính của một linh kiện thành phần có thể biến đổi bất kỳ trong cấu trúc chi tiết của nó thậm chí sau khi các tham số đã nói trước được thiết lập, thì những biến đổi này cần gây ra sự thay đổi tương đối nhỏ về hiệu năng vì các tham số chính đã được lựa chọn để cố định các đặc tính cần thiết của linh kiện. Do đó nếu các tham số này được lựa chọn như là các yếu tố điều khiển cho thiết kế phần cứng và nhắm chúng thành các thông số kỹ thuật thì một hệ thống giả định có thể được tổng hợp để gắn với các thông số này, và hiệu năng của một hệ thống được hiện thực hóa được gần đúng hoặc được giới hạn tốt bởi hiệu năng của mô hình phần mềm của hệ thống giả định.

1.4 Kỹ thuật đánh giá hiệu năng

Mục tiêu cuối cùng của mô phỏng là thu được một ước tính một số đại lượng hiệu năng mức hệ thống thông qua một số kỹ thuật đánh giá hiệu năng. Một kỹ thuật đánh giá hiệu năng (PET) là một tập các công cụ giải tích và các giả định được áp dụng với nhau trong một gói phần mềm mô phỏng cho mục đích ước tính hiệu quả một số phép đo hiệu năng. Đối với hệ thống truyền thông tương tự, phép đo hiệu năng cơ bản là tỉ số tín hiệu trên nhiễu đầu ra (SNR). Với hệ thống truyền thông số, phép đo hiệu năng là tỉ số lỗi bit (BER). Tỉ số tín hiệu trên nhiễu cũng là phép đo hiệu năng thứ cấp trong các hệ thống truyền dẫn số.

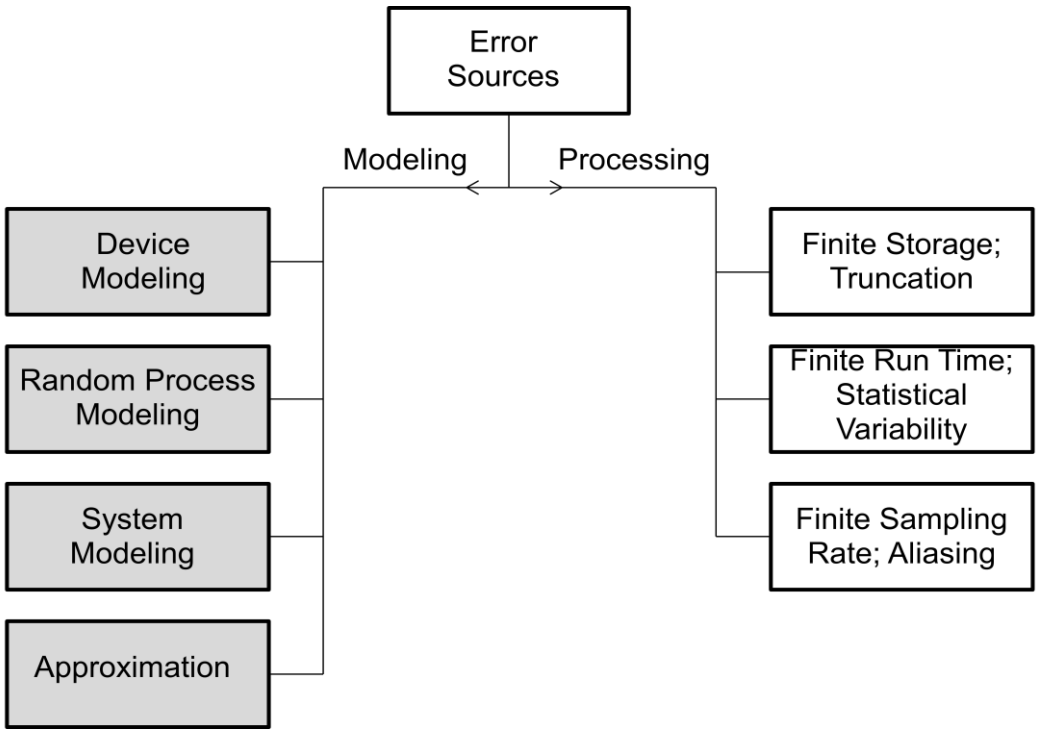
Trong các kỹ thuật PET, phương pháp Monte-Carlo thường được sử dụng để ước tính BER. Tuy nhiên cần chú ý thời gian chạy mô phỏng có thể rất lâu khi sử dụng phương pháp này. Đại lượng đo của một mô phỏng Monte-Carlo (SNR hoặc BER) là một biến ngẫu nhiên. Thời gian chạy mô phỏng càng lâu thì các quan sát càng sát với giá trị thực. Như đã đề cập có một sự bù trừ giữa thời gian chạy và độ chính xác của phép đo. Nếu một hệ thống được dự định để có BER cỡ 10^{-5} thì khi đó một bít lỗi được mong đợi quan sát thấy cứ mỗi 10^5 bits. Để thuyết phục được rằng BER thực sự khoảng 10^{-5} thì sẽ không đủ để chỉ quan sát 1 lỗi trong 10^5 bít. Để đảm bảo độ tin cậy của kết quả ước tính BER, số lượng bít cần truyền đi cần nằm trong dải 10/p đến 100/p với p là BER thực. Đối với $p = 10^{-5}$ thì 10/p tương ứng với 10^6 bít.

Trong trường hợp với p đủ nhỏ, số lượng bít cần truyền lớn làm cho thời gian chạy mô phỏng rất lâu và không khả thi. Do vậy có một số PET khác như sự thay thế hoặc biến đổi của phương pháp Monte-Carlo. Trong hệ thống tuyến tính, nhiễu Gauss được biết để duy trì tính Gauss. Kiến thức này cho phép ta kết hợp các biểu

thức giải tích với mô phỏng không nhiễu để thu được các ước tính BER rất hiệu quả nhanh hơn nhiều phương pháp MC. Sự kết hợp giữa kiến thức giải tich và kỹ thuật mô phỏng được gọi là kỹ thuật bán giải tích (QA). Ngoài kỹ thuật QA, một số kỹ thuật ước tính khác cũng được áp dụng để giảm thời gian chạy mô phỏng như kỹ thuật lấy mẫu quan trọng (IS) và kỹ thuật ngoại suy đuôi phân bố. Việc lựa chọn PET phù hợp cũng sẽ phụ thuộc vào nhiều yếu tố liên quan đến tính chất hệ thống được mô phỏng.

1.5 Sai số trong mô phỏng

Tính hữu ích của chạy mô phỏng liên hệ trực tiếp với độ chính xác của nó tức các kết quả mô phỏng sát với các đặc tính của hệ thống vật lý thực được mô phỏng. Nhìn chung độ chính xác bị giới hạn bởi hai loại sai số đó là sai số mô hình hóa tương ứng với các kiểu mô hình hóa đã đề cập ở trên (mô hình hóa hệ thống, mô hình hóa các thành phần linh kiện và mô hình hóa quá trình ngẫu nhiên) và sai số xử lý vì các giới hạn tính toán và bản chất thực của mô phỏng. Hình 1-6 cho bức tranh tóm tắt các nguồn sai số này.



Hình 1-6 Các nguồn sai số trong mô phỏng

1.5.1 Sai số trong mô hình hóa hệ thống

Nếu sơ đồ khối mô phỏng không phải đúng nguyên trạng tức là được sắp xếp theo cấu hình một – một với hệ thống thực, khi đó sẽ không tránh khỏi kết quả mô

phỏng không chính xác hoàn toàn với đặc tính của hệ thống thực. Lí do sơ đồ khối mô phỏng có thể không phải là bản sao chính xác hệ thống thực là để cắt giảm mức độ phức tạp từ đó giảm thời gian chạy mô phỏng. Tuy nhiên sự rút gọn sơ đồ khối cần lưu ý để không tác động nhiều đến tính chân thực của mô hình.

Một tình huống khác khi kết hợp cấu hình có thể không chính xác hoàn toàn là sự loại bỏ khỏi sơ đồ khối mô phỏng các thành phần đóng góp nhỏ đến méo dạng. Ví dụ như ta không mô hình hóa ăng-ten trong một mô phỏng vì một ăng-ten thường có độ rộng băng tần tương đối rộng so với tín hiệu đi vào. Tương tự, trong các ứng dụng thông tin vệ tinh bầu khí quyển thường bị bỏ qua như một phần tử méo dạng vì đặc tính hàm truyền thường (nhưng không phải luôn) trong suốt với tín hiệu. Do vậy điều quan trọng trong mô hình hóa hệ thống là cần xác định đúng các thành phần quan trọng và loại bỏ những thành phần ảnh hưởng ít đến hiệu năng để giảm mức độ phức tạp, tuy nhiên sẽ có sai số trong kết quả và cần đảm bảo mức sai số là không đáng kể.

1.5.2 Sai số trong mô hình hóa linh kiện

Một số sai số sẽ được sinh ra từ mô hình hóa các thành phần linh kiện. Tại mức cơ bản nhất ta không thể mong đợi một mô hình có độ chân thực hoàn hảo đối với thành phần thực, nhưng có thể có độ chân thực gần hoàn hảo đối với một mô hình rút gọn được lý tưởng hóa của linh kiện vật lý. Trong phân tích mục tiêu thực tế trong mô phỏng là để phát triển các mô hình đủ tốt hay tạo ra lỗi trong kết quả cuối cùng nhỏ có thể chấp nhận được.

Kết quả đầu ra mô phỏng và thực tế quan sát ví dụ như trường hợp bộ lọc nói chung luôn có sự khác nhau. Một lí do là bản thân các phép đo vật lý cũng không hoàn hảo. Như vậy các đặc tính vật lý đo được không giống một cách cần thiết như các đặc tính thật. Một lý do khác là các đặc tính của thành phần linh kiện không thể không đổi theo thời gian vì có thể do già hóa và phản ứng khác nhau đối với các điều kiện môi trường khác nhau.

Nó không cần thiết để sao chép một đặc tính của linh kiện một cách chi tiết nhất để có một mô hình tốt. Nó sẽ chỉ cần thiết để sao chép các đặc tính nổi bật chính về các hiệu ứng gây ra méo dạng của linh kiện. Khó khăn tiềm tàng chính nằm ở mô hình hóa các bộ khuyếch đại có nhớ. Tuy nhiên nó sẽ rất khó để ước tính ảnh hưởng tích lũy của các sai số mô hình hóa trong một chuỗi các khối kết nối với

nhau. Đó chính là lí do tại sao sự thẩm định bản thân mô hình tại các mức mô hình hóa khác nhau phải được thực hiện trong quá trình mô phỏng.

1.5.3 Sai số trong mô hình hóa quá trình ngẫu nhiên

Một nguồn sai số tiềm tàng nữa nằm ở mức độ mà các mô hình các quá trình ngẫu nhiên không mô tả chính xác như các quá trình thực. Các quá trình quan tâm thường là các nguồn tín hiệu và các nguồn nhiễu. Việc phỏng tạo một quá trình ngẫu nhiên được thực hiện bởi một bộ tạo số ngẫu nhiên sinh ra một chuỗi các số như là một tập mẫu. Như vậy đối với các nguồn tín hiệu tương tự như tiếng nói, bộ tạo số ngẫu nhiên giống như tạo ra một tập các mẫu thoại. Tuy vậy các kỹ thuật sử dụng cho bộ tạo số ngẫu nhiên có khả năng bị giới hạn để sao chép tất cả các tính chất của quá trình ngẫu nhiên. Nó thường chỉ có thể sao chép hai tính chất của một quá trình ngẫu nhiên là mật độ xác suất biên độ bậc một và mật độ phổ công suất. Những tính chất này nói chung là có thể đủ để nắm được các tính chất cần thiết của một quá trình cho nhiều mục đích.

Các mô hình nguồn cho hệ thống số là các bảng mẫu tự rời rạc và có thể có nhớ. Có hai trường hợp để mô phỏng hệ thống với đầu vào là tín hiệu đo thử hoặc là một nguồn tín hiệu thực. Trong trường hợp đầu chuỗi đo thử thường là chuỗi giả ngẫu nhiên có độ dài lớn nhất và được sinh ra từ máy tính. Thêm nữa ước tính hiệu năng sẽ không có sai số nếu một chuỗi de Bruijn có độ dài q^m với q là số lượng ký hiệu trong bảng mẫu tự và m là dung lượng nhớ của hệ thống được sử dụng và các ký hiệu được sinh ra một cách độc lập. Nếu nguồn thực không sinh ra các ký hiệu một cách độc lập thì mô hình hóa nguồn đó bằng một bộ tạo ký hiệu độc lập có thể gây ra sai số vì các mẫu giao thoa giữa các ký hiệu sẽ không được phân bố đều. Sai số có thể nhỏ hơn hoặc lớn hơn phụ thuộc vào các mẫu nào thường xuyên xuất hiện hơn. Nếu nguồn được biết có các phụ thuộc thì khi đó nó cần được mô hình hóa để phản ánh các tính chất thực của nó. Các phụ thuộc này thường được mô hình hóa như là các máy trạng thái hữu hạn và nếu mô hình đó được biết thì đơn giản cần thiết kết hợp nó vào trong mô phỏng.

Đối với các nguồn nhiễu trong hệ thống truyền thông có ba kiểu thường được quan tâm là nhiễu nhiệt, nhiễu pha và nhiễu nổ (nhiễu xung kim). Nhiễu nhiệt điển hình được mô hình hóa như nhiễu Gauss trắng và các bộ tạo nhiễu Gauss hiện nay hoàn toàn đủ tốt để xem như mô phỏng nhiễu nhiệt không gây sai số. Tuy nhiên nhiễu pha về tính không ổn định bộ dao động và nhiễu nổ là các quá trình không phải lúc nào cũng được biết rõ hoàn toàn. Do vậy các bộ tạo số ngẫu nhiên được sử

dụng đảm bảo sao chép sát nhất có thể với quá trình thực để sai số sinh ra là nhỏ nhất để đảm bảo độ tin cậy.

1.5.4 Sai số xử lý

Các sai số xử lý là vì bản chất của mô phỏng và giới hạn bộ nhớ, tốc độ và độ chính xác của máy tính.

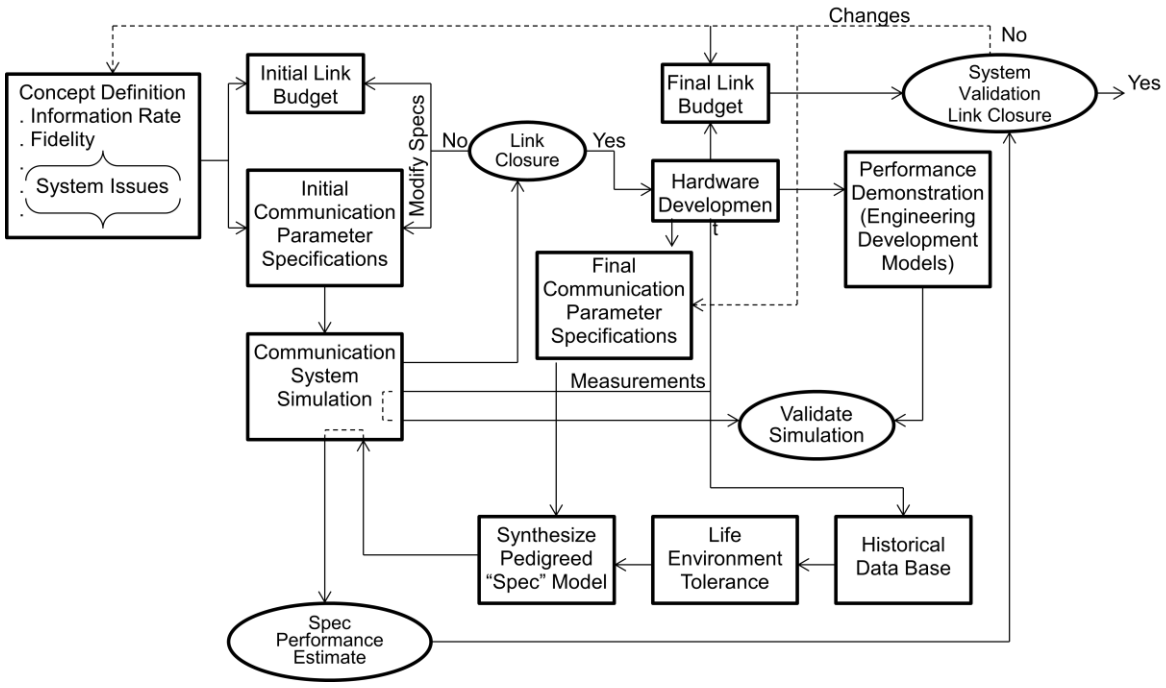
Trước hết ta thấy việc sử dụng đặc trưng rời rạc về thời gian cho các dạng sóng liên tục hoặc mô tả của các bộ lọc sẽ dẫn tới sai số chồng phổ. Tuy nhiên sai số này có thể được tính và được giới hạn ở mức nhỏ theo mong muốn. Về cơ bản trách nhiệm của người dùng mô phỏng là đảm bảo tốc độ lấy mẫu phù hợp để giảm sai số gây ra đến mức nhỏ nhất.

Một vấn đề giới hạn nữa gây ra sai số có thể thấy khi mô phỏng quá trình lọc sử dụng phương pháp bất biến xung kim do phải rút gọn đáp ứng xung của bộ lọc. Điều này sẽ thấy rõ ràng đối với bộ lọc IIR. Có một sự bù trừ giữa độ lớn sai số, độ dài đáp ứng xung được rút gọn và tải tính toán. Do đó cần lưu ý để sai số nằm ở mức nhỏ chấp nhận được bằng việc tính toán phần năng lượng bị bỏ qua trong đáp ứng xung chỉ nên nhỏ ở mức 1-2%.

Một số kiểu sai số xử lý khác cũng cần lưu ý trong mô phỏng như việc sử dụng mô hình tương đương băng gốc cho các quá trình thông dải hay giới hạn tốc độ của máy tính khi chạy mô phỏng Monte-Carlo không đảm bảo đủ lâu. Do vậy ngoài mô hình hóa thì quá trình xử lý tính toán trong mô phỏng cũng cần xem xét để tránh các lỗi nghiêm trọng có thể xảy ra trong kết quả đầu ra.

1.6 Vai trò mô phỏng trong thiết kế hệ thống truyền thông

Trong thế giới thực tế của quá trình thiết kế và xây dựng các hệ thống truyền thông, mô phỏng trở thành công cụ để hỗ trợ trong quá trình này. Mô phỏng không chỉ được sử dụng để ước tính hiệu năng và tối ưu tham số trong thiết kế mà còn được dùng để thiết lập các thủ tục kiểm tra và kiểm chuẩn, các dự đoán tuổi thọ cũng như nghiên cứu tính bất thường sau khi hệ thống được triển khai trong thực tế. Cả phương pháp luận mô phỏng và mô hình mô phỏng đều được sử dụng để biểu diễn hệ thống phụ thuộc vào các giai đoạn khác nhau của quá trình thiết kế, triển khai và vòng đời của hệ thống. Phương pháp luận mô phỏng cũng sẽ được kiểm soát hoặc định hướng bởi quá trình thiết kế tổng thể.



Hình 1-7 Phương pháp luận thiết kế hệ thống truyền thông và vai trò của mô phỏng.

Quá trình thiết kế một hệ thống truyền thông phức tạp được thực hiện từ "trên xuống", ngược lại thực thi phần cứng thường từ "dưới lên". Theo đó, trong thiết kế hệ thống ta bắt đầu tại mức hệ thống (mức trừu tượng cao nhất) và bắt đầu triển khai chi tiết thiết kế từ mức hệ thống, xuống mức hệ thống con và cuối cùng là mức thành phần linh kiện. Tại mức dưới cùng chi tiết của các bộ phận thiết bị được nhận dạng. Trong khi xây dựng một hệ thống các thành phần linh kiện được chế tạo trước, sau đó lắp ráp thành các hệ thống con và cuối cùng toàn bộ hệ thống được xây dựng từ các hệ thống con. Triển khai mô phỏng đi theo cách tiếp cận từ trên xuống với bắt đầu bằng việc mô phỏng mức hệ thống có mức trừu tượng cao tăng dần mức độ chi tiết và các mô phỏng các hệ thống con và các thành phần.

Quá trình thiết kế hệ thống sẽ trải qua các giai đoạn khác nhau và ở mỗi giai đoạn mô phỏng đều đóng một vai trò nhất định.

Quá trình thiết kế một hệ thống truyền thông bắt đầu bằng việc trình bày, phân tích các yêu cầu người dùng và hiệu năng mong muốn gồm: thông lượng, tỷ số lỗi, xác suất ngừng hoạt động, hạn chế băng thông, công suất, trọng lượng, phức tạp/chi phí, kênh truyền trên đó hệ thống hoạt động, và tuổi thọ trung bình của hệ thống. Trên cơ sở những yêu cầu của người dùng, "kỹ sư hệ thống" xuất phát từ khái niệm ban đầu về hệ thống như: sơ đồ điều chế, kỹ thuật mã hóa, cân bằng nếu cần. Tập giá trị tham số gọi là các chỉ tiêu kỹ thuật mức A như các mức công suất, băng thông, chỉ số điều chế cũng được thiết lập trong giai đoạn khởi đầu của quá

trình thiết kế. Mục đích toàn diện tại điểm này là: (i) xác định cấu hình hệ thống; (ii) các giá trị của tham số cần để đáp ứng các mục tiêu hiệu năng và thỏa mãn những ràng buộc thiết kế. Hiệu năng hệ thống là một hàm của SNR (tương đương E_b/N_o) và méo tổng gây ra bởi tất cả các thành phần trong tuyến truyền thông. SNR được thiết lập thông qua một quá trình gọi là dự trữ tuyến là phần tính toán công suất xem xét các yếu tố như: công suất phát, hệ số khuyếch đại ăng ten, suy hao đường truyền, khuyếch đại công suất, hệ số nhiễu của bộ khuyếch đại và bộ lọc. Trong khi dự trữ tuyến không phải là đại lượng quan tâm chính trong mô phỏng, nhưng nó thiết lập một dải các giá trị SNR hoặc E_b/N_o trên đó mô phỏng được thực hiện để ước tính hiệu năng.

Người thiết kế hệ thống bắt đầu bằng cấu hình của hệ thống, các chỉ tiêu kỹ thuật mức A và độ dự trữ tuyến ban đầu. Dự trữ tuyến thường được biểu diễn ở dạng bảng tính và dòng cuối cùng trong dự trữ tuyến là E_b/N_0 thực tại điểm tới hạn trong hệ thống sau khi tất cả các tổn hao thực thi được tính đến. "Điểm tới hạn" này thường là đầu vào bộ thu. Dự trữ tuyến được gọi là "kín" hay "cân bằng" nếu tuyến có E_b/N_0 đủ lớn với hệ số dự phòng an toàn để tạo ra hiệu năng hệ thống chấp nhận được. Tại điểm này trong quá trình thiết kế, số đo hiệu năng được tính toán từ các công thức xấp xỉ và chưa được mô phỏng. Vì tất cả những tổn hao thực thi đã được tính đến trong E_b/N_0 thực nên có thể tính BER theo công thức cho hệ thống lý tưởng. Nếu độ dự trữ tuyến không kín hay không cân bằng thì khi đó các chỉ tiêu kỹ thuật mức A, các tổn hao thực thi và thậm chí cấu hình hệ thống được thay đổi và độ dự trữ tuyến được tính toán lại. Ví dụ băng thông của bộ lọc có thể thay đổi, kích thước ăng ten có thể tăng lên và đặc tính kỹ thuật hệ số tạp âm của bộ khuyếch đại có thể được làm thấp đi. Quá trình này sẽ tiếp tục cho đến khi độ dự trữ tuyến được cân bằng với hệ số dự trữ thích hợp.

Trên cơ sở cấu hình hệ thống ban đầu, các đặc tính kỹ thuật mức A và độ dự trữ tuyến, khi này được coi là kín, cho phép xây dựng một mô hình mô phỏng để kiểm tra độ dự trữ tuyến và tinh chỉnh thiết kế. Có thể ước tính chính xác các phép đo hiệu năng và kiểm tra suy thoái hiệu năng do thực thi không lý tưởng thông qua các mô phỏng chi tiết. Nếu những phân bổ trong độ dự trữ tuyến được kiểm tra thông qua mô phỏng và độ dự trữ tuyến vẫn được kín thì quá trình thiết kế chuyển sang giai đoạn kế tiếp mà ở đó chứa thiết kế chi tiết, thực thi các hệ thống con và các thành phần. Nếu độ dự trữ tuyến không kín thì một số phân bổ méo, cấu hình hệ thống và đặc tính kỹ thuật mức A có thể bị thay đổi. Quá trình lặp cho đến khi độ

dự trữ tuyến cân bằng cung cấp các đặc tính kỹ thuật cho phát triển phần cứng (và phần mềm).

Thiết kế một hệ thống truyền thông mới sẽ luôn chứa một số thuật toán xử lý tín hiệu mới và các công nghệ phần cứng (phần mềm) mới. Bất kỳ công nghệ mới luôn có rủi ro hoặc tính không chắc chắn về hiệu năng. Nếu công nghệ mới được đưa vào một phần tử quan trọng của một hệ thống thì trước hết thành phần đó phải được xây dựng và kiểm tra trong các điều kiện thực tế nhằm kiểm tra hiệu năng và giảm thiểu rủi ro. Vì chỉ vài thành phần then chốt được xây dựng tại giai đoạn đầu của tiến trình thiết kế nên không thể kiểm tra toàn bộ hệ thống phần cứng. Trong tình huống này, mô phỏng tạo ra môi trường kiểm tra tuyệt vời và sử dụng mô phỏng ít tốn kém hơn nhiều so với chế tạo mẫu thử phần cứng. Tất cả các thành phần và tín hiệu được mô phỏng với các đặc tính đo được của thành phần được đưa vào trong mô hình mô phỏng. Ví dụ, nếu thành phần sẽ được kiểm tra là bộ khuyếch đại mới, thì các đặc tính hàm truyền đạt AM-AM và AM-PM của nó được đo và các đặc tính này được đưa vào mô hình phi tuyến cho bộ khuyếch đại đó. Sau đó mô phỏng toàn bộ hệ thống để kiểm tra hiệu năng và độ dự trữ tuyến. Nếu độ dự trữ tuyến kín thì việc phát triển phần cứng tiếp tục cho thành phần quan trọng tiếp theo. Nếu không thì thiết kế lại thành phần, xây dựng lại và kiểm tra lại hoặc độ dự trữ tuyến được sửa đổi để tính đến suy thoái phụ gây ra do thành phần (ngoài những gì đã được phân bổ trong độ dự trữ tuyến cho thành phần đó). Quy trình này được lặp lại đối với các thành phần then chốt khác.

Như thủ tục đã được mô tả, chế tạo nguyên mẫu phần cứng của toàn bộ hệ thống bắt đầu xuất hiện cùng với mô hình mô phỏng kèm theo. Mô hình mô phỏng bây giờ bao gồm các đặc tính đo được cho hầu hết các thành phần trong mô hình. Nhiều số đo hiệu năng cho toàn hệ thống có thể được thực hiện trên nguyên mẫu phần cứng này. Các mô phỏng song song cũng được thực hiện. Các đặc tính hiệu năng đo được có thể so sánh với các kết quả mô phỏng và ngược lại. Các mô phỏng cung cấp những điểm chuẩn để kiểm tra và kết quả kiểm tra sẽ kiểm định mô phỏng. Kết quả cuối cùng của quá trình thiết kế này là một nguyên mẫu hoàn chỉnh của hệ thống, cho ta cơ sở để triển khai phiên bản sản phẩm của hệ thống. Ngoài ra, mô hình mô phỏng được kiểm định có thể được sử dụng để dự đoán tuổi thọ với độ tin cậy cao.

Trong khi quy trình trên dẫn đến thiết kế đảm bảo mức hiệu năng cho trước khi triển khai hệ thống thì một yêu cầu quan trọng nữa phải được đáp ứng cho hầu hết các hệ thống, là hiệu năng tuổi thọ. Nhiều hệ thống truyền thông như vệ tinh, hệ

thống cáp biển được mong đợi phải có tuổi thọ dài (thường khoảng 10 năm hoặc hơn), trong khoảng thời gian này hiệu năng phải được đảm bảo. Tất nhiên, không thể kiểm tra vòng đời thực tế dựa trên nguyên mẫu phần cứng. Trong khi các thủ tục kiểm tra vòng đời tăng tốc đã được phát triển, thực tế thường dùng mô phỏng như giải pháp bổ sung để kiểm tra vòng đời tăng tốc. Các dự đoán hiệu năng tuổi thọ dùng mô phỏng được thực hiện bằng sử dụng các mô hình già hóa cho các thành phần chính của hệ thống. Nếu ta có mô hình mô phỏng đã được kiểm chuẩn cho toàn bộ hệ thống tại lúc bắt đầu vòng đời và cũng có những mô hình đặc tính tốt của các thành phần như hàm thời gian thì các mô hình già hóa có thể được thay thế trong mô hình bắt đầu vòng đời để đạt được các số đo hiệu năng tuổi thọ cho hệ thống. Nếu hiệu năng dự đoán tuổi thọ là thỏa đáng và dự trữ tuyến cuối đời là kín với hệ số dự phòng thỏa đáng thì việc thiết kế và thực thi hệ thống là hoàn thiện. Ngược lại quá trình phải lặp lại cho đến khi đạt được hội tụ.

Tóm tắt các bước then chốt trong trình tự thiết kế và vai trò của mô phỏng trong thiết kế các hệ thống truyền thông được minh họa ở hình 1-7.

1.7 Tổng kết chương

Chương 1 đã cung cấp những vấn đề cơ bản nhất liên quan đến mô phỏng nói chung và mô phỏng hệ thống truyền thông nói riêng. Các bước cơ bản thực hiện trong bài toán mô phỏng và những vấn đề đơn giản hóa bài toán đã được trình bày. Cấu trúc phân cấp trong mô phỏng và những vấn đề về mô hình hóa cũng được bàn luận đến. Cuối cùng vai trò của mô phỏng trong thiết kế hệ thống truyền thông đã được phân tích giúp người đọc hiểu rõ hơn vai trò của môn học này.

Câu hỏi/bài tập chương 1

1/ Một bài toán mô phỏng được thực hiện như thế nào ?

2/ Trong xây dựng bài toán mô phỏng cần chú ý những vấn đề gì ?

3/ Vai trò của mô hình phân cấp trong mô phỏng ?

4/ Có các kiểu mô hình hóa nào? Tại sao mô hình hóa quá trình ngẫu nhiên có ý nghĩa quan trọng trong mô phỏng hệ thống truyền thông ?

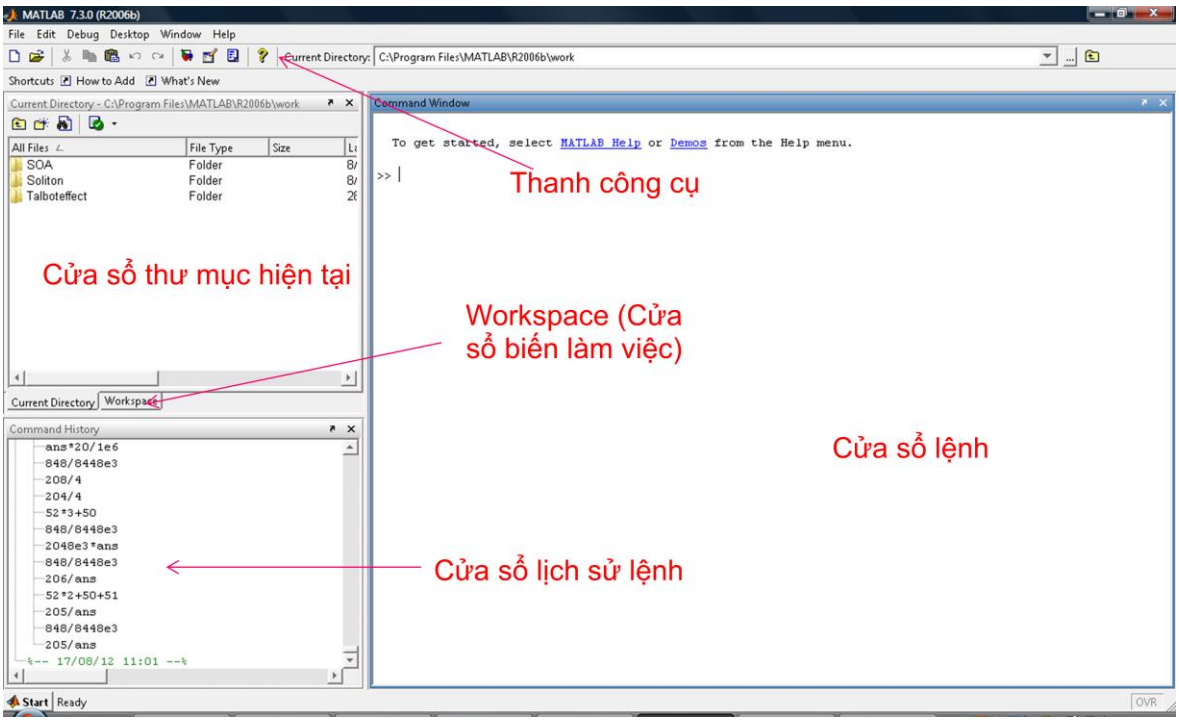
5/ Mô phỏng có vai trò gì trong thiết kế hệ thống truyền thông?

Chương 2 Giới thiệu về MATLAB

Chương này giới thiệu một công cụ tính toán trong kỹ thuật đó là MATLAB. Ngoài những nội dung về cấu trúc cơ bản của MATLAB giúp sinh viên biết cách sử dụng cho tính toán, chương này cũng còn cung cấp một số phương pháp số và vận dụng MATLAB để giải một số bài toán hay gặp trong kỹ thuật.

2.1 Giới thiệu chung

MATLAB là một công cụ mô phỏng và tính toán số được phát triển như một công thương mại với giao diện thân thiện với người dùng và các thư viện các hàm số. Khác với các chương trình tính toán máy tính khác cái thường đòi hỏi các cấu trúc dữ liệu phức tạp dẫn tới cú pháp phức tạp theo đối với người dùng, thì MATLAB chỉ liên quan đến một cấu trúc dữ liệu đơn mà tất cả các hoạt động dựa trên đó. Cấu trúc này là một trường số hay còn gọi là ma trận, đó cũng là lý do mà chương trình có tên MATLAB viết tắt từ MATrix LABoratory.



Hình 2-1 Cửa sổ giao diện MATLAB

MATLAB cũng là một ngôn ngữ dịch cho phép các lệnh được thực hiện trực tiếp và cho ra kết quả ngay, đây cũng là đặc điểm thuận tiện của MATLAB so với các ngôn ngữ lập trình khác. Để hỗ trợ các phép tính toán dạng biểu tượng giống như các chương trình MAPLE hay MATHEMATICA, một bộ công cụ symbolic cũng đã được bổ sung trong MATLAB kể từ phiên bản 7.0.

2.2 Các cấu trúc cơ bản trong MATLAB

2.2.1 Các biến MATLAB

Một biến MATLAB là một đối tượng thuộc về một kiểu dữ liệu cụ thể. Kiểu dữ liệu cơ bản nhất trong MATLAB là ma trận. Do vậy một biến MATLAB về cơ bản là một ma trận. Các ma trận có thể được hình thành từ các số thực hoặc phức cũng như các ký tự (ký tự ASCII) liên quan đến xử lý các xâu ký tự (string).

Định nghĩa biến MATLAB: Biến là một dãy ký tự thường bắt đầu bằng chữ cái có độ dài tối đa 31 kí tự và có phân biệt chữ hoa và chữ thường được gán giá trị theo cú pháp sau:

```
Tên_biến = biểu thức
```

Ví dụ:

```
>> x = 2.5
x =
    2.5000
```

Định nghĩa một vector và một ma trận được thực hiện với các giá trị của các phần tử cho trong ngoặc vuông [ ]. Các giá trị của các phần tử ở trên cùng một hàng được cách bởi dấu cách (space) hoặc bởi dấu phẩy (,), còn giữa các hàng sẽ được phân biệt bằng dấu chấm phẩy (;).

Ví dụ: Tạo một vector hàng và kết quả

```
>> v = [1 5 -3]
v =
     1     5    -3
```

Ví dụ: Tạo một ma trận và kết quả

```
>> m = [3 1+2*i 2; 4 0 -5]
m =
```

```
   3.0000    1.0000 + 2.0000i   2.0000
   4.0000         0            -5.0000
```

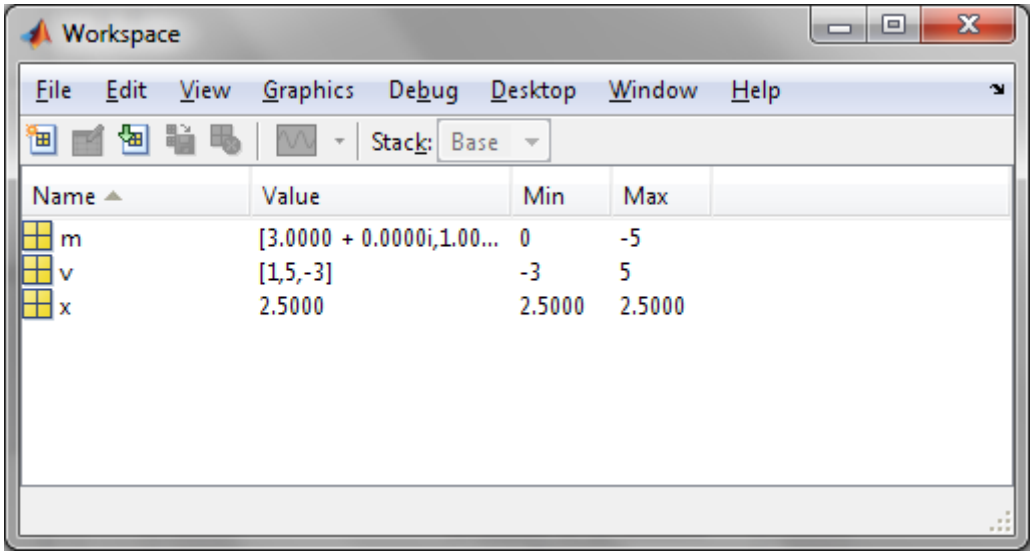
Chú ý ma trận m chứa một phần tử là số phức. Các số phức có thể được định nghĩa theo cách đặc trưng đại số sử dụng kí hiệu i và j làm số ảo. Nếu không có tên biến được đặt thì MATLAB tự gán tên `ans` (answer) cho kết quả như ví dụ sau:

```
>> [2 3 4; 3 -1 0]
ans =
     2.0000    3.0000    4.0000
     4.0000   -1.0000   -5.0000
```

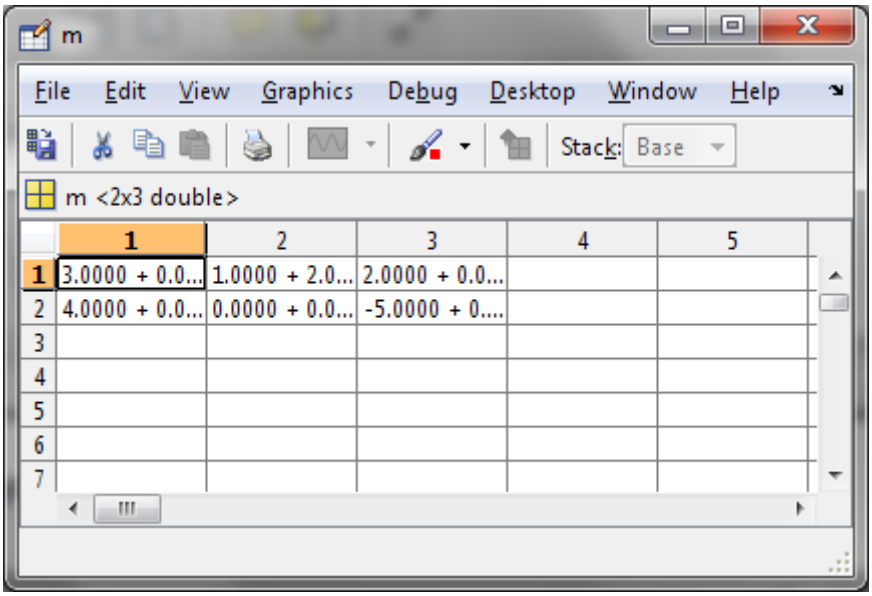
Quản lý biến: Tất cả các biến được định nghĩa sẽ được lưu giữ trong workspace của MATLAB để kiểm tra trạng thái của biến ở bất cứ lúc nào. Lệnh `who` sẽ trả về các tên biến được lưu giữ và lệnh `whos` cung cấp các thông tin bổ sung về các biến.

```
>> who
Your variables are:
m  v  x
>> whos
  Name      Size          Bytes  Class     Attributes

  m         2x3              96  double    complex
  v         1x3              24  double
  x         1x1               8  double
```

Cửa sổ workspace cũng cho đầy đủ các thông tin về các biến được lưu giữ như trong hình 2-2. Bằng việc kích đúp vào một biến trong cửa sổ này sẽ mở ra một cửa sổ variable editor và hiển thị các nội dung của các biến theo kiểu bảng tính Excel như cho thấy trong hình 2-3. Nhiều biến có thể được mở đồng thời cùng lúc, khi đó một thanh công cụ được mở ra để cho sự lựa chọn biến. Kích thước của ma trận, định dạng và các gí trị của các phần tử trong ma trận có thể được thay đổi trong variable editor. Điều này sẽ rất hữu ích đối với các ma trận kích thước lớn.



Hình 2-2 Giao diện cửa sổ workspace



Hình 2-3 Giao diện và mô tả của một ma trận trong variable editor.

Một số các biến không cần sử dụng nữa, cách dễ dàng nhất để xóa chúng là sử dụng lệnh clear như ví dụ sau:

```
>> clear m x
```

Để xóa toàn bổ các biến có thể sử dụng lệnh clear hoặc clear all. Các hoạt động này cũng có thể thực hiện qua cửa sổ workspace.

Xử lý các biến: Với cấu trúc cơ bản là một ma trận, các phần tử trong một biến MATLAB có thể được thay đổi bằng việc trỏ tới vị trí phần tử đó trong ma trận qua ngoặc đơn (n,m) với n là chỉ số hàng và m là chỉ số cột như ví dụ sau:

```
>> m(1,3) = 1
```

```
m =
      3.0000       1.0000 + 2.0000i      1.0000
      4.0000            0               -5.0000
```

Việc truy xuất giá trị các phần tử trong ma trận cũng được thực hiện theo cách tương tự.

```
>> a = m(1,:)
a =
      3.0000     1.0000 + 2.0000i    1.0000
```

Một ma trận hay một vector có thể được mở rộng bằng việc bổ sung thêm các phần tử hoặc thu hẹp bằng việc loại bỏ các phần tử. Quá trình mở rộng bằng cách thêm hàng như ví dụ sau:

```
>> m = [m; 1 2 3]
m =
  3.0000          1.0000 + 2.0000i     1.0000
  4.0000               0              -5.0000
  1.0000          2.0000               3.0000
```

Việc bổ sung thêm cột cũng được thực hiện theo cách tương tự:

```
>> m = [m, [1; 2; 3]]
m =
  3.0000    1.0000 + 2.0000i   1.0000    1.0000
  4.0000         0            -5.0000    2.0000
  1.0000    2.0000             3.0000    3.0000
```

hoặc bằng cách:

```
>> v = [1; 2; 3]
v =
     1
     2
     3
>> m = [m, v]
m =
  3.0000    1.0000 + 2.0000i    1.0000    1.0000
```

```
  4.0000         0        -5.0000    2.0000
  1.0000    2.0000         3.0000    3.0000
```

Các hoạt động xóa bớt hàng hoặc cột trong một ma trận được thực hiện bằng việc gán hàng hoặc cột cần xóa bằng một vector trống [ ]. Hàng hoặc cột cần xóa được lựa chọn qua chỉ số của ma trận.

Ví dụ xóa cột 2 trong ma trận m:

```
>> m(:,2)= []
m =
      3     1     1
      4    -5     2
      1     3     3
```

Ví dụ xóa hàng đầu tiên trong ma trận:

```
>> m(1,:)= []
m =
      4    -5     2
      1     3     3
```

Ở đây dấu hai chấm (:) có nghĩa là lấy giá trị của tất cả các hàng hoặc cột tùy thuộc vị trí của dấu trong chỉ số ma trận.

Đối với các ma trận hoặc vector có kích cỡ lớn, kết quả đầu ra tại cửa sổ lệnh thường gây sự bất tiện. Để chặn việc hiển thị kết quả đầu ra trên cửa sổ lệnh ta có thể dùng dấu chấm phẩy (;) tại cuối câu lệnh trước khi thực hiện. Ví dụ định nghĩa một vector kích cỡ lớn gồm các số từ 1 đến 5000 với khoảng cách 2 đơn vị.

```
>> vt = 0:2:5000;
```

Lệnh trên thực hiện tạo ra một dải giá trị có thể được thực hiện theo cú pháp tổng quát sau: `tên_biến = a:del:b` với a, b là các giá trị đầu và cuối của dải giá trị và del là giá trị cỡ bước thay đổi. Dấu ; cuối câu lệnh sẽ chặn việc hiển thị kết quả trên cửa sổ lệnh, nếu không sử dụng dấu này kết quả có thể được hiển thị lần lượt trên cửa sổ bằng lệnh `more on` hoặc tắt chức năng này bằng lệnh `more off`.

2.2.2 Các phép tính số học

Các hoạt động số học (+, -, *, v.v.) trong MATLAB được thực hiện đơn giản và thuận tiện, giúp người mới bắt đầu dễ dàng thực hiện các phép tính.

Các phép tính ma trận: Vì cấu trúc dữ liệu cơ bản của MATLAB là ma trận nên các hoạt động số học được xem như là các phép tính ma trận. Do đó mà các quy tắc tính toán trong ma trận cần phải được tuân thủ. Ví dụ phép nhân 2 ma trận phải đảm bảo số cột ma trận trước bằng số hàng của ma trận sau. Nếu điều kiện không được thỏa mãn phép tính không thực hiện được và sẽ báo lỗi. Một ngoại lệ với quy tắc xảy ra chỉ nếu một trong các biến là vô hướng hoặc một ma trận 1x1.

```
>> M = [1 2 3; 4 -1 2]           % Tạo ma trận M 2x3
M =
       1     2     3
       4    -1     2
>> N = [1 2 -1; 4 -1 1; 2 0 1]   % Tạo ma trận N 3x3
N =
      1     2    -1
      4    -1     1
      2     0     1
>> V = M*N                  % Thử tính phép nhân M*N
V =
      15     0     4
       4     9    -3
>> W = N*M                  % Thử phép nhân N*M
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

Các phép tính theo từng phần tử: Bên cạnh các phép tính ma trận, trong nhiều trường hợp có nhu cầu thực hiện các phép tính đại số tương ứng cho từng phần tử. Các phép tính theo từng phần tử được xem như các phép tính trường hay mảng trong MATLAB được thực hiện bằng việc su dụng dấu chấm (.) trước toán tử thực hiện. Tuy nhiên hoạt động toán học đối với hai ma trận theo từng phần tử đòi hỏi hai ma trận có kích thước giồng nhau.

```
>> M = [1 2 3; 4 -1 2]           % Tạo ma trận M 2x3
M =
       1     2     3
       4    -1     2
```

```
>> N = [1 -1 0; 2 1 -1]          % Tạo ma trận N 2x3
N =
      1    -1     0
      2     1    -1
>> M*N                        % Nhân ma trận MxN
??? Error using ==> mtimes
Inner matrix dimensions must agree.
 >> M.*N                     % Nhân từng phần tử
ans =
         1    -2     0
         8    -1    -2
```

Các phép tính chia: Dấu chia có ý nghĩa đặc biệt. Trong MATLAB có sự phân biệt giữa chia phải / và chia trái \. Để hiểu rõ sự khác biệt xét trường hợp các ma trận vuông, phép chia hai ma trận X = A/B có thể xem như là $X = A.B^{-1}$ nếu ma trận đảo B^{-1} tồn tại. Đối với phép chia trái X = A\B có thể xem như là $X = A^{-1}.B$ nếu A^{-1} tồn tại.

```
>> A = [2 1; 1 1]                % Tạo ma trận A 2x2
A =
      2     1
      1     1
>> B = [-1 1; 1 1]               % Tạo ma trận B 2x2
B =
     -1     1
      1     1
>> Ainv = inv(A)              % Tính ma trận A^-1
Ainv =
           1    -1
          -1     2
>> Binv = inv(B)              % Tính ma trận B^-1
Binv =
          -0.5000    0.5000
           0.5000    0.5000
```

```
>> X1 = A/B                    % Phép chia phải
X1 =
       -0.5000    1.5000
         0    1.0000
>> X2 = A*Binv                 % Đối chiếu
X2 =
       -0.5000    1.5000
         0    1.0000
>> Y1 = A\B                    % Phép chia trái
Y1 =
      -2.0000   -0.0000
       3.0000    1.0000
>> Y2 = Ainv*B                 % Đối chiếu
Y2 =
       -2     0
        3     1
```

Một trong những phép chia có thể áp dụng để tìm nghiệm cho bài toán hệ phương trình đại số tuyến tính trong kỹ thuật $A\vec{x} = \vec{b}$, nghiệm được xác định qua phép chia trái $\vec{x} = A \backslash \vec{b}$

```
>> A = [2 1; 1 1]              % Tạo ma trận hệ số A
A =
      2     1
      1     1
>> b = [2; 1]                  % Vector b
b =
     2
     1
>> x = A\b                     % Tìm nghiệm
x =
     1.0000
     0.0000
```

```
>> A*x                          % Kiểm chứng lại
ans =
         2.0000
         1.0000
```

2.2.3 Các phép tính logic và quan hệ

Các hoạt động logic dựa trên các toán tử trường và thu được các giá trị logic đúng (true) hoặc sai (false) được thể hiện bằng giá trị 1 hoặc 0 tương ứng. Ví dụ kiểm tra xem liệu các thành phần của hai ma trận có cùng giá trị ≠ 0 (= giá trị logic true) hay không bằng sử dụng toán tử logic & (AND):

```
>> A = [1 -3; 0 0]                 % Tạo ma trận A
A =
      1    -3
      0     0
>> B = [0 5; 0 1]                  % Tạo ma trận B
B =
      0     5
      0     1
>> res = A&B                       % Hoạt động AND
res =
        0     1
        0     0
```

Ma trận kết quả `res` chỉ chứa các giá trị 1 và 0, giá trị 1 tương ứng với các thành phần của hai ma trận cùng ≠ 0, nếu không sẽ là 0. Ngoài toán tử & còn có các toán tử logic khác: OR (|), NOT (~) và XOR (xor).

Các toán tử quan hệ hay so sánh cũng làm việc theo kiểu tương tự toán tử logic. Ví dụ về kiểm tra liệu các thành phần ma trận A lớn hơn các thành phần ma trận B tương ứng:

```
>> comp = A>B                      % Hoạt động so sánh lớn hơn
comp =
          1     0
          0     0
```

Như vậy chỉ những phần tử nào thỏa mãn điều kiện sẽ trả về kết quả logic 1 (true), nếu không thỏa mãn sẽ trả về kết quả logic 0 (false). Các toán tử quan hệ khác bao gồm ≥ và ≤ (theo cú pháp MATLAB sẽ là >= và <=), nhỏ hơn (<), bằng nhau (==) và không bằng nhau (~=).

Để biết được hết tất cả các toán tử cơ bản trong MATLAB có thể sử dụng lệnh trợ giúp trong cửa sổ lệnh `help ops`. Sau khi thực hiện một danh sách các toán tử MATLAB sẽ được liệt kê.

Việc sử dụng các toán tử quan hệ và logic có nhiều ý nghĩa trong mô phỏng. Ví dụ bài toán lựa chọn các thành phần từ một vector có giá trị lớn hơn 2:

```
>> vect = [-2, 3, 0, 4, 5, 19, 22, 17, 1]   % Tạo vector
vect =
        -2     3     0     4     5    19    22    17     1
>> compvect = 2*ones(1, 9)  % Hàm ones tạo ma trận giá trị 1
compvect =
         2     2     2     2     2     2     2     2     2
>> comp = vect>compvect           % Tạo biến logic so sánh
comp =
           0     1     0     1     1     1     1     1     0
>> res = vect(comp) % Chọn lọc các giá trị thỏa mãn điều kiện
res =
      3     4     5    19    22    17
```

Nguyên tắc chung các vector và ma trận kết nối bởi các toán tử quan hệ phải có cùng kích thước. Trong trường hợp so sánh chỉ với một giá trị, lệnh thực hiện có thể đơn giản hơn:

```
>> comp = vect>2
comp =
       0     1     0     1     1     1     1     1     0
```

Các trường logic không chỉ sinh ra từ kết quả hoạt động so sánh mà có thể cũng được định nghĩa trực tiếp theo cách sau:

```
% Tạo biến trực tiếp
>> logiField1 = [true, true, false, true, false, true]
logiField1 =
```

```
           1     1     0     1     0     1
>> numField = [1, 1, 0, 1, 0, 1]       % Tạo mảng số
numField =
          1     1     0     1     0     1
% Chuyển đổi sang trường logic
>> logiField2 = logical(numField)
logiField2 =
                  1     1     0     1     0     1
```

2.2.4 Các hàm toán học

MATLAB có rất nhiều các hàm toán học sẵn có trong các thư viện hàm (toolboxes). Để biết được các hàm tính toán cơ bản có thể sử dụng trợ giúp trong cửa sổ lệnh qua lệnh sau: `help elfun`. Kết quả thực hiện sẽ cho một danh sách tên các hàm toán cơ bản và mô tả ngắn gọn các hàm. Hoạt động của một hàm toán lên một vector hay ma trận được thực hiện theo từng phần tử.

Ví dụ tính hàm sine của một vector:

```
>> t = (0:1:5)
t =
     0     1     2     3     4     5
>> s = sin(t)
s =
   0    0.8415    0.9093    0.1411   -0.7568   -0.9589
```

Kết quả trả về của hàm cũng cùng kích thước với tham số biến đầu vào. Ý nghĩa của việc tính toán các hàm toán học dạng vector hoặc ma trận trong MATLAB thấy rõ trong các mô phỏng lớn khi cho phép thay thế dạng vòng lặp trong lập trình ở các ngôn ngữ khác. Ví dụ trong C++, thực hiện chuỗi tính toán trên đòi hỏi phải thực hiện bởi vòng lặp sau:

```
double s[6];
for(i=0; i<6; i++)
     s[i] = sin(i);
```

Để biết thông tin về cách sử dụng các hàm một cách trực tiếp, có thể sử dụng lệnh trợ giúp: `help <tên_hàm>`. Ví du:

```
>> help sin
```

```
 SIN    Sine of argument in radians.
    SIN(X) is the sine of the elements of X.

    See also asin, sind.

    Overloaded methods:
       codistributed/sin

    Reference page in Help browser
       doc sin
```

Bên cạnh các hàm toán cơ bản, các hàm toán đặc biệt cần thiết cho việc tính toán trong kỹ thuật cũng có thể được liệt kê bằng việc sử dụng lệnh help specfun và help matfun.

Một số ví dụ sau cho thấy việc sử dụng một số hàm cơ bản trong MATLAB. Ví dụ xác định biên độ và pha theo radian và theo độ của một vector số phức, đây là bài toán hay gặp trong việc xác định hàm truyền đạt của các hệ thống tuyến tính.

```
>> cnum = [1+j, j, 2*j, 3+j, 2-2*j, -j]% Tạo vector số phức
cnum =
  Columns 1 through 4
   1.0000 + 1.0000i          0 + 1.0000i          0 + 2.0000i
3.0000 + 1.0000i
  Columns 5 through 6
   2.0000 - 2.0000i        0 - 1.0000i
>> magn=abs(cnum)     % Xác định thành phần biên độ
magn =
    1.4142    1.0000    2.0000    3.1623    2.8284    1.0000
>> phase=angle(cnum)  % Xác định thành phần pha theo rad
phase =
    0.7854    1.5708    1.5708    0.3218   -0.7854   -1.5708
>> deg = angle(cnum)*360/(2*pi)  % Chuyển đổi sang độ
deg =
   45.0000   90.0000   90.0000   18.4349  -45.0000  -90.0000
```

Một ví dụ khác chuyển đổi các giá trị tín hiệu đo ở dạng điện áp sang dạng dB bởi hàm $20.\log_{10}(U)$ với U là tín hiệu đo.

```
>> meas = [25.5 16.3 18.0;...% Tạo ma trận giá trị đo điện áp
2.0 6.9 3.0; ...
0.05 4.9 1.1]
meas =
   25.5000   16.3000   18.0000
    2.0000    6.9000    3.0000
    0.0500    4.9000    1.1000
>> dBmeas=20*log10(meas)          % Chuyển đổi sang hệ dB
dBmeas =
   28.1308   24.2438   25.1055
    6.0206   16.7770    9.5424
  -26.0206   13.8039    0.8279
```

Trong ví dụ trên khi một lệnh MATLAB quá dài mà muốn viết tiếp ở dòng tiếp theo thì có thể sử dụng dấu ba chấm (…) tại cuối mỗi dòng.

2.2.5 Các hàm đồ họa

Một trong những sức mạnh nội bật của MATLAB là khả năng biểu diện đồ họa trực quan các kết quả tính toán. MATLAB có sẵn các hàm đồ họa đơn giản những mạnh mẽ để biểu diễn các đồ thị hai chiều (xy) và các đồ thị ba chiều (xyz).

Một sự khảo sát tất cả các hàm đồ họa có thể thu được bằng lệnh `help graph2d`, `help graph3d` và `help graphics` trong cửa sổ lệnh.

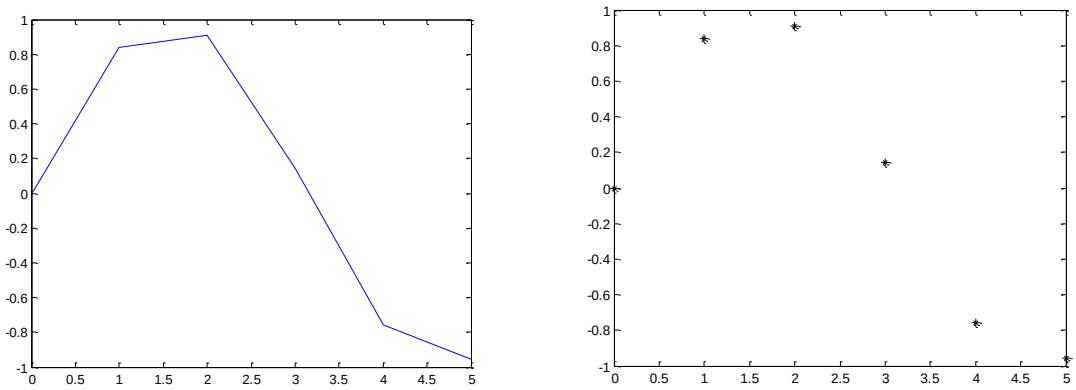
Đồ thị hai chiều: Hàm đồ họa sử dụng nhiều nhất và quan trọng nhất là hàm `plot`. Để biết cách sử dụng hàm và cung cấp thêm thông tin, sử dụng lệnh trợ giúp `help plot`. Đồ thị của một hàm được vẽ dựa trên nguyên tắc nối các điểm được đặc trưng bởi cặp giá trị (x,y) lại với nhau. Do vậy hai vector được yêu cầu để vẽ một đồ thị. Vector đầu tiên đặc trưng cho vector giá trị trục x và vector thứ hai đặc trưng cho vector giá trị trục y. Các vector do đó phải có cùng kích thước.

Ví dụ vẽ đồ thị hàm sine:

```
>> t = (0:1:5);
>> s = sin(t);
>> plot(t,s)
```

Kết quả được cho thấy trong hình 2-4 với các thuộc tính ở chế độ mặc định như đường mầu xanh và liền nét. Các thuộc tính đồ thị có thể thay đổi theo ý muốn như ví dụ:

```
>> plot(t,s,'k*')      % Vẽ đồ thị dạng điểm sao mầu đen
```



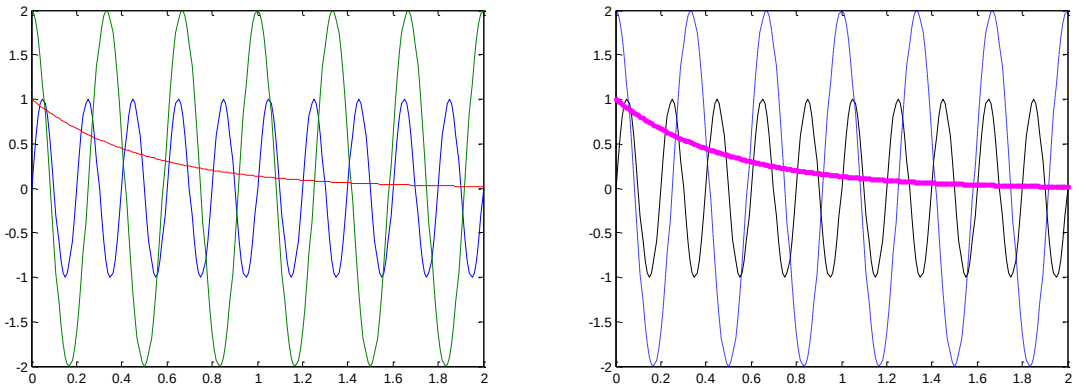
Hình 2-4 Ví dụ vẽ đồ thị sử dụng hàm `plot`

Nhiều hàm đồ thị cũng có thể được vẽ đồng thời hoặc bằng cách viết mỗi cặp vector x,y sau cặp khác trong danh sách tham số, hoặc khi cùng vector x được sử dụng thì có thể kết hợp các vector y thành một ma trận tương ứng. Ví dụ vẽ ba hàm đồ thị khác nhau trên cùng một hình:

```
>> t = (0:0.01:2);
>> sinfct = sin(2*pi*5*t);
>> cosfct = 2*cos(2*pi*3*t);
>> expfct = exp(-2*t);
>> plot(t,[sinfct; cosfct; expfct])
```

Kết quả được cho thấy trong hình 2-5. Các thuộc tính của từng đồ thị có thể thay đổi theo ý muốn theo lệnh sau và kết quả thu được như hình 2-5:

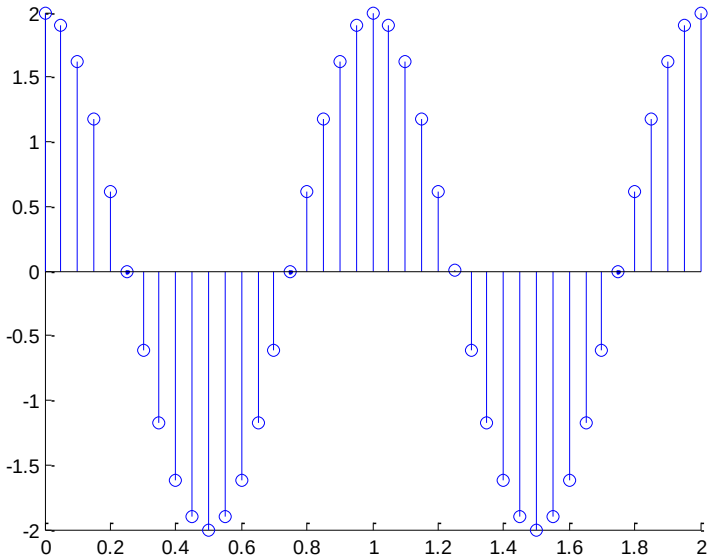
```
>> plot(t,sinfct,'k-', t, cosfct, 'b--', t, expfct, 'm.')
```



Hình 2-5 Ví dụ vẽ nhiều đồ thị sử dụng hàm `plot`.

Bên cạnh hàm plot được sử dụng thường xuyên, MATLAB còn có một số hàm vẽ đồ thị 2D khác. Trong xử lý tín hiệu hàm stem thường được sử dụng để đặc trưng cho tính rời rạc của tín hiệu. Hình 2-6 cho thấy kết quả sử dụng hàm `stem` qua ví dụ dưới đây:

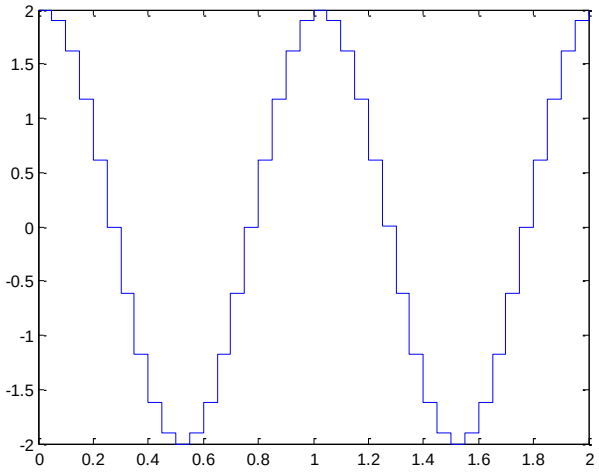
```
>> t=(0:0.05:2);
>> cosfct=2*cos(2*pi*t);
>> stem(t,cosfct)
>> box
```



Hình 2-6 Ví dụ vẽ đồ thị sử dụng hàm `stem`.

Một hàm phù hợp để hiển thị các tín hiệu rời rác là hàm stairs vẽ tín hiệu ở dạng bậc thang. Hình 2-7 cho thấy kết quả sử dụng hàm stairs qua lệnh sau:

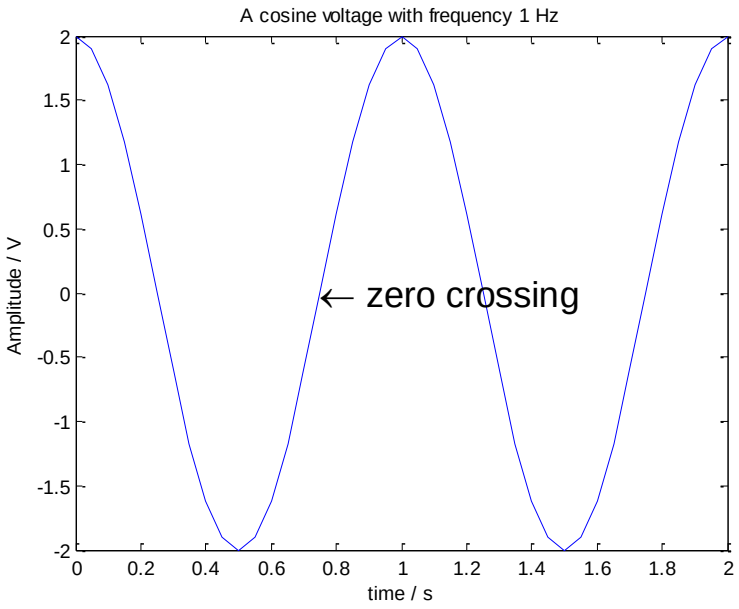
```
>> stairs(t,cosfct)
```



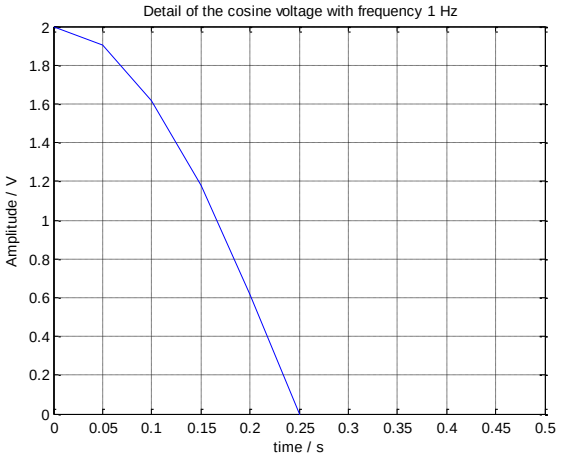
Hình 2-7 Ví dụ vẽ đồ thị sử dụng hàm `stairs`

MATLAB cũng cung cấp một số hàm để bổ sung thêm các chức năng biểu diễn đồ thị của hình như đánh nhãn các trục bằng các hàm `xlabel`, `ylabel` và `title`, kẻ lưới hình bằng hàm `grid`, thay đổi định cỡ truc bằng hàm `axis`. Chuỗi lệnh MATLAB sau cho ví dụ về sử dụng các hàm bổ sung này và kết quả cho thấy trong hình 2-8 và 2-9.

```
>> t=(0:0.05:2);
>> cosfct=2*cos(2*pi*t);
>> plot(t,cosfct)
>> xlabel('time / s')
>> ylabel('Amplitude / V')
>> text (0.75,0,'\leftarrow zero crossing','FontSize',18)
>> title('A cosine voltage with frequency 1 Hz')
>> figure % mở một cửa sổ hình mới !
>> plot(t,cosfct)
>> xlabel('time / s')
>> ylabel('Amplitude / V')
>> grid % thiết lập khung lưới đồ thị
>> axis([0, 0.5, 0, 2]) % biểu diễn trong khoảng [0, 0.5],
% còn biên độ trong khoảng [0, 2]
>> title('Detail of the cosine voltage with frequency 1 Hz')
```



Hình 2-8 Ví dụ bổ sung thêm các thuộc tính cho đồ thị

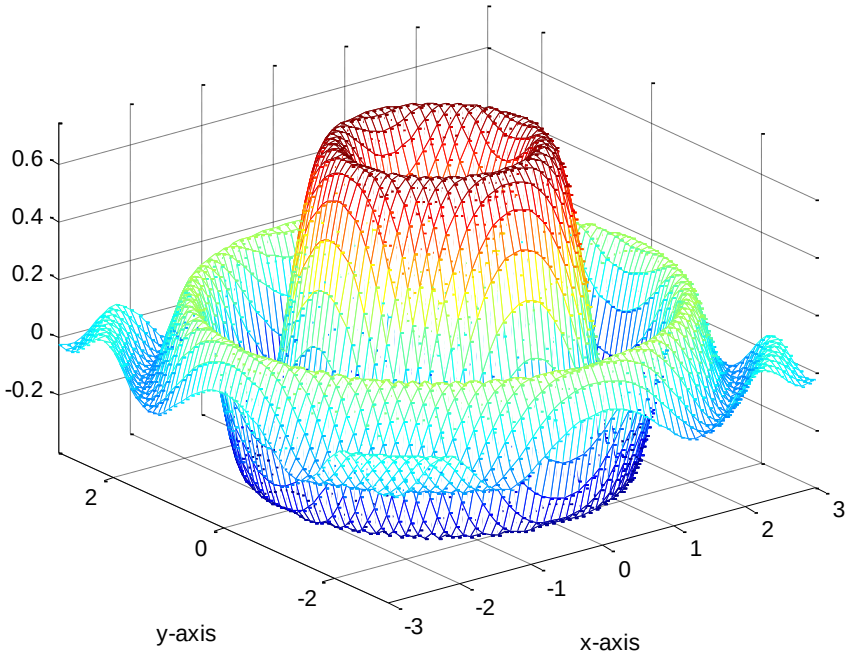


Hình 2-9 Ví dụ sử dụng hàm `axis` để thay đổi định cỡ đồ thị

Bên cạnh xử lý đồ thị bằng các hàm kể trên, một cách khác thuận tiện để xử lý và thay đổi các thuộc tính của đồ thị là vào menu `Edit – Figure Properties`, khi đó một cửa sổ `Property Editor` mở ra như một thanh công cụ cho phép thay đổi các thuộc tính tùy theo đối tượng (trục, đường, hình) được lựa chọn. Một cách mở cửa sổ `Property Editor` khác là lựa chọn đối tượng và kích đúp bằng chuột trái.

Đồ thị ba chiều: Có một số hàm vẽ đồ thị ba chiều hay được sử dụng là hàm `mesh`, `surf` và `contour`. Trong vẽ ba chiều mỗi điểm trên đồ thị được đặc trưng bởi 3 giá trị (x,y,z), thông thường giá trị z được tính dựa trên cặp giá trị (x,y) bởi

một hàm xác định. Do vậy để vẽ được đầy đủ mặt ba chiều của một hàm cần phải tạo ra tổ hợp các điểm lưới trên mặt phẳng (x,y) trước khi tính z.

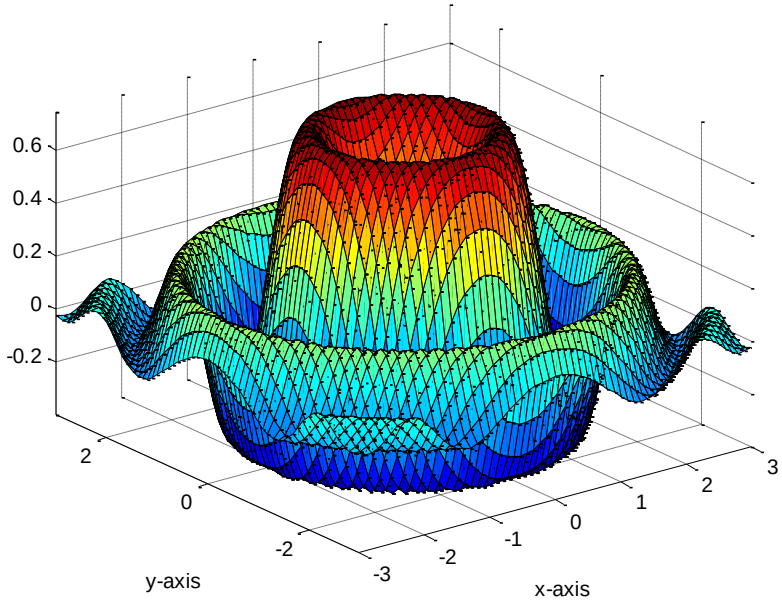


Hình 2-10 Ví dụ vẽ đồ thị 3 chiều sử dụng hàm `mesh`

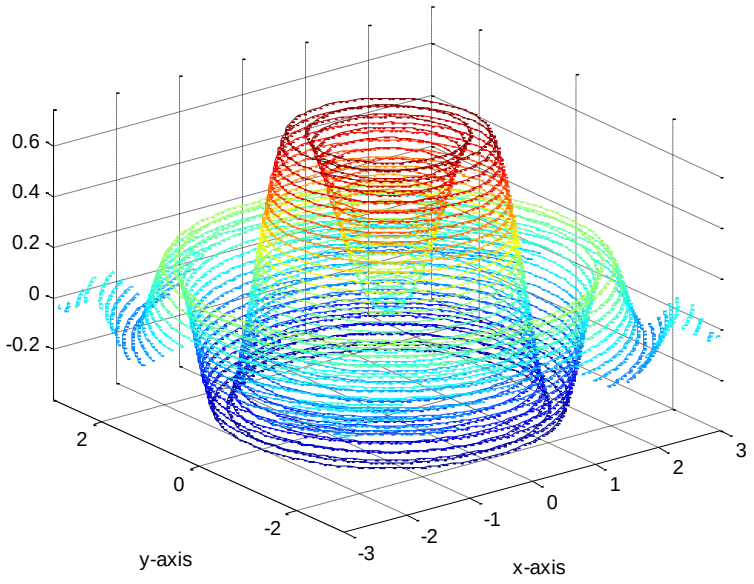
Ví dụ vẽ đồ thị hàm $f(x,y)=\sin\left(x^2+y^2\right)e^{-0.2\left(x^2+y^2\right)}$ trong dải giá trị mặt (x,y) [-3, 3]x[-3, 3] sử dụng hàm `mesh` hoặc `surf`. Chuỗi lệnh thực hiện như sau:

```
>> x=(-3:0.1:3); % Tạo dải giá trị theo chiều x
>> y=(-3:0.1:3)'; % Tạo dải giá trị theo chiều y
>> v=ones(length(x),1); % auxiliary vector
>> X=v*x; % Tạo lưới tổ hợp các điểm ma trận X
>> Y=y*v'; % Tạo lưới tổ hợp các điểm ma trận Y
>> % Tính giá trị hàm
>> f=sin(X.^2+Y.^2).*exp(-0.2*(X.^2+Y.^2));
>> mesh(x,y,f) % vẽ bằng hàm mesh
>> mxf = max(max(f)); % xác định giá trị cực đại của hàm
>> mif = min(min(f)); % xác định giá trị cực tiểu của hàm
>> axis([-3,3,-3,3,mif,mxf])% hiệu chỉnh các trục
>> xlabel('x-axis'); % label axes
>> ylabel('y-axis');
```

```
>> figure % mở hình mới
>> surf(x,y,f) % vẽ bằng hàm surf
>> axis([-3,3,-3,3,mif,mxf])% hiệu chỉnh các trục
>> xlabel('x-axis'); % label axes
>> ylabel('y-axis');
```



Hình 2-11 Ví dụ vẽ đồ thị 3D sử dụng hàm `surf`



Hình 2-12 Ví dụ vẽ đồ thị 3D sử dụng hàm contour3

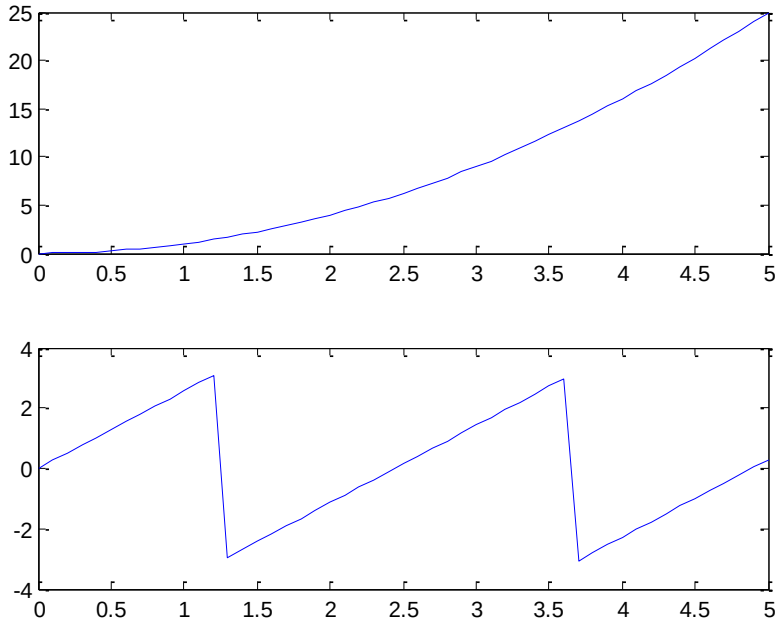
Một hàm hay sử dụng để vẽ các đường đẳng mức trong không gian ba chiều là hàm `contour3`. Hình 2-12 cho thấy đồ thị contour sau khi thực hiện các lệnh sau:

```
>> contour3(x,y,f,30) % vẽ bằng hàm contour3
% với 30 đường đẳng mức.
>> axis([-3,3,-3,3,mif,mxf])% adjust axes
>> xlabel('x-axis'); % label axes
>> ylabel('y-axis');
```

Các đồ thị con (Subplots): Bên cạnh khả năng vẽ các đồ thị trong các cửa sổ khác nhau thì MATLAB cũng có thể vẽ nhiều đồ thị trong cùng một cửa sổ đơn. Chức năng này có thể được thực hiện bằng sử dụng lệnh `subplot`.

Ví dụ vẽ biên độ và pha của hàm phức $f(t) = t^2 e^{jt} j^t$ trên hai đồ thị trong cùng một hình:

```
>> t=(0:0.1:5);
>> f=(t.^2).*exp(j*t).*(j.^t); % Tính hàm.
>> subplot(211) % Vẽ đồ thị con trên
>> plot(t,abs(f))
>> subplot(212) % Vẽ đồ thị con dưới
>> plot(t,angle(f))
```



Hình 2-13 Ví dụ vẽ nhiều đồ thị phân chia trên cùng một hình bằng hàm `subplot`

Ở đây lệnh `subplot` thiết lập các trục đồ thị ở vị trí thích hợp, các tham số của lệnh subplot xác định bao nhiêu đồ thị được vẽ theo chiều dọc (số đầu tiên) và chiều ngang (số thứ hai). Số thứ ba xác định đồ thị thứ mấy sẽ được thực hiện theo chiều từ trên xuống dưới và từ trái qua phải.

Ngoài vẽ nhiều đồ thị cùng lúc như cho thấy trong sử dụng hàm `plot`, chức năng vẽ nhiều đồ thị đè lên nhau trong hình có thể được bật tắt qua sử dụng lệnh `hold on` và `hold off`.

2.2.6 Các hoạt động I/O

Các hoạt động Input/Output liên quan đến việc truyền dữ liệu giữa MATLAB và các tệp bên ngoài. Hoạt động này là cần thiết khi ta muốn nhập dữ liệu thu được từ bên ngoài vào MATLAB hoặc truyền các kết quả từ MATLAB tới các ứng dụng khác.

Các lệnh load và save:

Lệnh save thực hiện lưu nội dung các biến trong workspace vào một tệp nhị phân có đuôi mở rộng .mat với một số cú pháp cơ bản sau:

```
% Lưu tất cả các biến trong workspace vào tệp
>> save <tên_file>
>> save <tên_file>  x y  % Chỉ lưu các biến x và y vào tệp
% Lưu các biến x và y vào tệp với đường dẫn xác định theo
% định dạng của phiên bản MATLAB 6
>> save 'đường_dẫn\tên_file' x y -V6
```

Lệnh `load` đọc dữ liệu lưu ở tệp để đưa vào workspace theo cú pháp cơ bản sau:

```
>> load <tên_file>   % Đọc tất cả các biến lưu trong tệp
>> load <đường_dẫn\tên_file>  x y  % Chỉ đọc các biến x và y lưu ở tệp
```

Bên cạnh định dạng tệp .mat, các định dạng khác ví dụ như ASCII cũng có thể được lưu sử dụng lệnh `save` hoặc được đọc sử dụng lệnh `load`. Để biết thêm thông tin sử dụng các lệnh này sử dụng lệnh trợ giúp `help save` và `help load`.

Các hoạt động tương tự cũng có thể được thực hiện qua lựa chọn menu `File – Save Workspace As` và `File – Import Data`.

2.3 Thao tác ma trận và vectơ

Có một số lệnh MATLAB phổ biến được sử dụng để điều khiển ma trận. Lệnh help elmat sẽ cho biết toàn bộ danh sách các lệnh này.

Một số lệnh để tạo các ma trận đặc biệt:

```
>> M = zeros(2,2)                % Tạo ma trận giá trị 0
M =
     0     0
     0     0
>> N = ones(3,2)              % Tạo ma trận giá trị toàn 1
N =
     1     1
     1     1
     1     1
>> x1 = [1,2,3,4,5,6];       % Tạo vector x1
>> v=zeros(length(x1),1)     % Khởi tạo vector cột cùng kích thước với x1
v =
     0
     0
     0
     0
     0
     0
```

Lệnh `eye` để tạo một ma trận đơn vị:

```
>> E5 = eye(5)
E5 =
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
```

Chiều dài của một vector và kích thước của một ma trận có thể được xác định qua sử dụng các lệnh `length` và `size`.

```
>> length(x1)          % Xác định độ dài vector x1
ans =
       6
>> B = E5;             % Lưu ma trận E5 vào B
>> B(:,2) = [ ]  % Xóa cột thứ 2
B =
     1     0     0     0
     0     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
>> [rows, columns] = size(B) % Xác định kích thước ma trận B
rows =
         5
columns =
         4
```

Một số kiểu ma trận, biến đặc biệt cũng được liệt kê bởi help elmat. Khi sử dụng MATLAB tránh đặt tên biến trùng với các biến đặc biệt này có thể làm thay đổi giá trị khi sử dụng.

Hoạt động chuyển vị của một ma trận hoặc vector có thể sử dụng toán tử ' và được đặt ngay sau ma trận cần được chuyển vị.

```
>> M = [1 2; 3 -2; -1 4]    % Tạo ma trận M 3x2
M =
     1     2
     3    -2
    -1     4
>> N = M'              % Chuyển vị ma trận M
N =
     1     3    -1
```

```
     2    -2     4
```

Tuy nhiên cần chú ý khi đối với ma trận số phức khi thực hiện toán tử ' sẽ có liên hợp phức.

```
>> M = [i 2; 3 -j]    % Tạo ma trận M có số phức
M =
   0 + 1.0000i   2.0000
   3.0000        0 - 1.0000i
>> N = M'        % Ma trận chuyển vị
N =
   0 - 1.0000i   3.0000
   2.0000        0 + 1.0000i
```

Muốn tránh liên hợp phức trong trường hợp này cần sử dụng dấu (.) trước toán tử:

```
>> K = M.'
K =
   0 + 1.0000i   3.0000
   2.0000        0 - 1.0000i
```

Chuyển đổi một ma trận thành một vector cột qua sử dụng dấu hai chấm (:):

```
>> M = [1 2; 3 -2; -1 4]          % Tạo ma trận M
M =
     1     2
     3    -2
    -1     4
>> mVec = M(:)
mVec =
     1
     3
    -1
     2
    -2
     4
```

Hàm `repmat` có thể được sử dụng để sao chép lặp lại ma trận:

```
>> N = repmat(M,2,2)  % Sao chép ma trận M 4 lần
% theo sắp xếp 2x2
N =
     1     2     1     2
     3    -2     3    -2
    -1     4    -1     4
     1     2     1     2
     3    -2     3    -2
    -1     4    -1     4
```

Một toán tử hay được sử dụng trong lập trình là toán tử end, xác định chỉ số cuối cùng của một vector mà không cần giá trị cụ thể.

```
>> yVec = [0, 3, -1, 0, 1, 99]
yVec =
     0     3    -1     0     1    99
>> yVec(end+1:end+3) = [-1 -2 -3]
yVec =
     0     3    -1     0     1    99    -1    -2    -3
```

2.4 Lập trình trong MATLAB

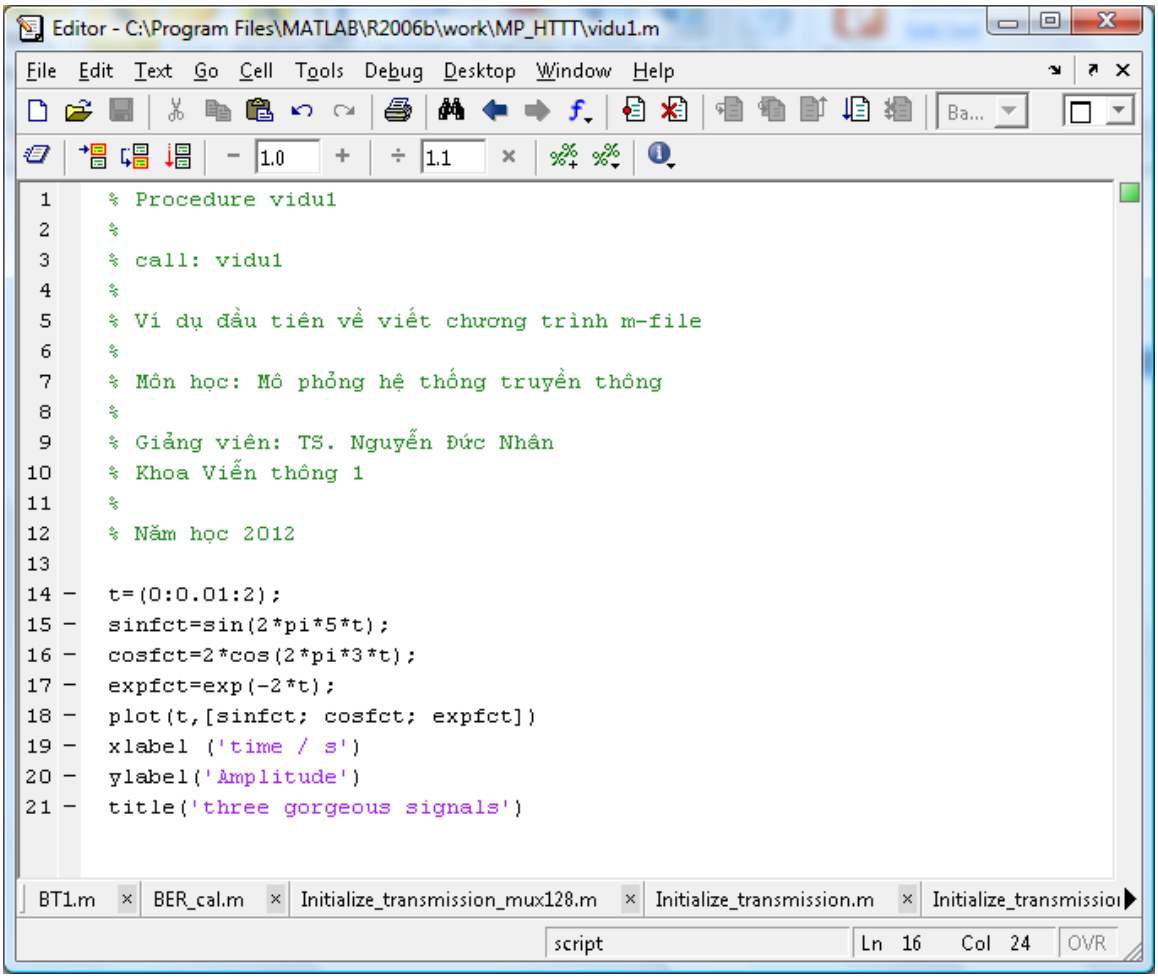
MATLAB bản thân nó cũng là một ngôn ngữ lập trình, điều này có nghĩa rằng MATLAB gồm các cấu trúc ngôn ngữ chương trình như các vòng lặp và phân nhánh và cung cấp khả năng viết các hàm và các thủ tục giống như những ngôn ngữ lập trình khác.

2.4.1 Các thủ tục MATLAB

Lập trình trong MATLAB cung cấp các tệp các tập lệnh trong đó các chuỗi lệnh MATLAB được tập hợp lại thành các thủ tục đơn giản. Việc viết các chuỗi lệnh MATLAB được thực hiện qua cửa sổ lập trình `Editor` có thể được mở ra khi vào mục `File – New – m-file` và được lưu vào trong một tệp dưới một cái tên có đuôi mở rộng .m (m-file) trong một đường dẫn có thể truy nhập tới MATLAB.

Chuỗi các lệnh khi đó có thể được thực hiện trong cửa sổ lệnh chi bằng một lệnh tương ứng với tên m-file. Để tránh các vấn đề xung đột xảy ra, bất kỳ tên lệnh được sử dụng trước đó không nên được dùng. Để kiểm tra xem liệu tên lệnh đã tồn tại hay chưa có thể sử dụng lệnh trợ giúp `help <tên>` ở cửa sổ lệnh hoặc sử dụng hàm `exist <tên>`.

Ví dụ thủ tục viết chương trình MATLAB được cho thấy trong hình 2-14 cùng với giao diện bộ `Editor`. Một chuỗi các lệnh có thể thực hiện lần lượt ở ngoài cửa sổ lệnh được tập hợp vào một tệp và lưu lại với tên tệp là `vidu1.m`.



Hình 2-14 Ví dụ viết chương trình MATLAB trong cửa sổ lập trình Editor

Tại cửa sổ lệnh bằng việc thực hiện:

```
>> vidu1
```

chuỗi lệnh trong tệp sẽ được thực hiện và cho kết quả giống như hình 2-5. Tại đầu của tệp chương trình, dấu % được sử dụng đặt trước các nội dung chú giải của chương trình. Các chú giải là cần thiết để làm cho chương trình rõ ràng và luôn có trong lập trình chuyên nghiệp. Các nội dung chú giải ở đầu tệp có thể được đọc ở

cửa sổ lệnh bằng lệnh trợ giúp help, ví dụ: `help vidu1`. Chú ý rằng, khi chương trình thực hiện, tất cả các biến được định nghĩa trong tệp đều được biết trong workspace.

2.4.2 Các hàm con MATLAB

Trong một số trường hợp để linh hoạt trong tính toán người sử dụng định nghĩa các hàm MATLAB (functions) hơn là kết hợp một cách đơn giản các lệnh trong tệp chương trình vì các hàm có thể truyền qua các tham số.

Cấu trúc cơ bản của một hàm trong m-file như sau:

```
function [các tham số đầu ra] = tên_hàm(các tham số đầu vào)
% Ghi chú giải hàm
```

ở đây các tham số đầu vào và các tham số đầu ra cách nhau bằng dấu phẩy (,). Những điều cần lưu ý khi xây dựng hàm đó là tên hàm và tên tệp m-file phải giống nhau, trong hàm có thể chứa các hàm khác nhưng các hàm con trong hàm đó chỉ được gọi trong chính nó. Mỗi hàm có không gian làm việc riêng nên khác với thủ tục ở trên các biến được tạo ra trong hàm là các biến cục bộ và chỉ có ý nghĩa trong không gian làm việc của hàm. Nếu hàm muốn sử dụng chung các biến thì biến đó phải được khai báo là biến toàn cục: `global tên_biến`.

Việc gọi hàm ở cửa sổ lệnh hay trong m-file có thể thực hiện theo cách sau:

```
>> [y1,y2,…] = tên_hàm(x1,x2,…) hoặc
>> tên_hàm(x1,x2,…)
```

với x1, x2, … là các tham số chứa giá trị tương ứng với các tham số trong hàm được xây dựng. Lưu ý các tham số đầu vào và đầu ra khi hàm được gọi chỉ có tác dụng bên trong hàm đó. Để quản lý số lượng tham số đầu vào và đầu ra của hàm có thể sử dụng các biến đặc biệt `nargin` và `nargout` tương ứng.

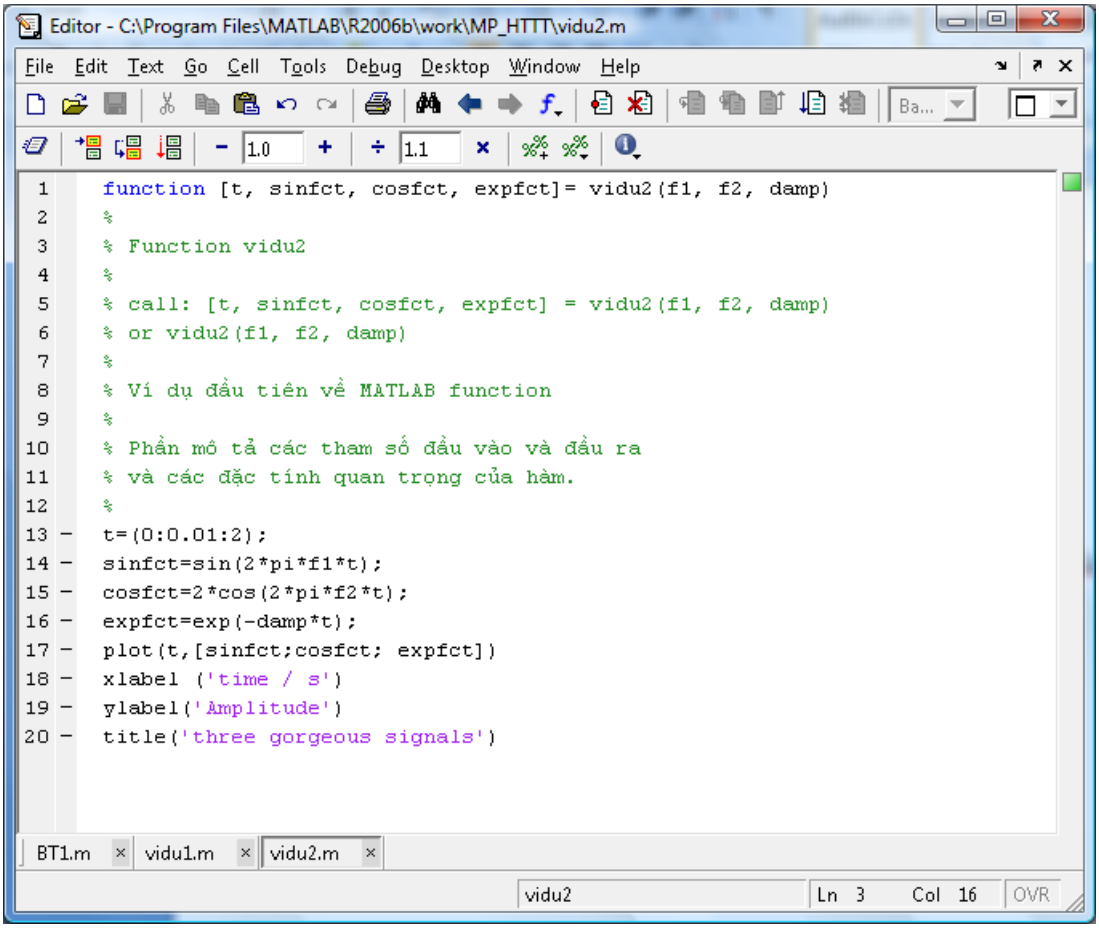
Ví dụ xây dựng một hàm đơn giản với cấu trúc:

```
[t,sinfct,cosfct,expfct] = vidu2(f1,f2,damp)
```

được cho thấy trong hình 2-15. Hàm được lưu cùng với tên tệp là `vidu2.m`. Ngoài cửa sổ lệnh việc gọi hàm có thể thực hiện như:

```
>> [t,s1,c1,e1] = vidu2(3,5,2);
```

kết quả tại workspace các biến `t, s1, c1, e1` được tạo ra với các giá trị được tính bởi hàm từ các tham số đầu vào là các giá trị nhập trực tiếp và đồ thị tương tự hình 2-5.



Hình 2-15 Ví dụ viết một hàm trong MATLAB

2.4.3 Cấu trúc ngôn ngữ MATLAB

MATLAB cũng là một ngôn ngữ lập trình và như vậy nó cũng có các cấu trúc ngôn ngữ chương trình. Để biết được các cấu trúc ngôn ngữ trong MATLAB có thể sử dụng lệnh trợ giúp `help lang`. Còn để biết được thông tin chi tiết của mỗi cấu trúc có thể sử dụng lệnh `help <tên_cấu_trúc>`. Một số cấu trúc cơ bản sẽ được trình bày trong phần này.

Câu lệnh `if`: có cấu trúc cơ bản như sau

```
if biểu_thức1
      các_câu_lệnh1
elseif biểu_thức2
      các_câu_lệnh2
```

```
else
      các_câu_lệnh3
end
```

Ví dụ sử dụng câu lệnh `if` viết một hàm so sánh hai số A và B

```
function sosanh(A,B)
if A > B
    disp('A  lon hon B');
elseif A == B
    disp('A bang B');
else
    disp('A nho hon B');
end
```

Hàm được lưu lại vào tệp với tên `sosanh.m` và có thể gọi ngoài cửa sổ lệnh

```
>> sosanh(5,2)
A  lon hon B
```

Câu lệnh `for`: dạng cấu trúc lặp được sử dụng khi đã biết số vòng lặp và có cấu trúc cơ bản như sau:

```
for biến = biểu_thức
       câu_lệnh
      ...
       câu_lệnh
end
```

Ví dụ sử dụng câu lệnh `for` tính giá trị các phần tử trong ma trận A kích thước 5x5 với $a_{ij} = 1/(i + j - 1)$

```
k = 5;
a = zeros(k,k)   % Khởi tạo kích thước ma trận A
for m = 1:k
    for n = 1:k
        a(m,n) = 1/(m+n -1);     % Tính giá trị các phần tử
    end
end
```

```
disp('Ma tran A:');
disp(a);
```

Chương trình được lưu vào tệp với tên `tinhmatran.m`. Ngoài cửa sổ lệnh có thể gọi chương trình để kiểm tra kết quả:

```
>> tinhmatran
Ma tran A:
    1.0000    0.5000    0.3333    0.2500    0.2000
    0.5000    0.3333    0.2500    0.2000    0.1667
    0.3333    0.2500    0.2000    0.1667    0.1429
    0.2500    0.2000    0.1667    0.1429    0.1250
    0.2000    0.1667    0.1429    0.1250    0.1111
```

Câu lệnh `while`: dạng cấu trúc lặp được sử dụng khi chưa biết số vòng lặp nhưng có điều kiện ràng buộc và có cấu trúc cơ bản như sau:

```
while biểu_thức
      các_câu_lệnh
end
```

Ví dụ sử dụng câu lệnh `while` tìm N để $S = \sum_{k=1}^{N} k^2 < 1000$

```
n = 1
s = 0;
while s < 1000
    s = s + n^2;
    n = n + 1;
end
disp('Gia tri tong S:');
disp(s);
disp('Gia tri n:');
disp(n-1);
```

Chương trình được lưu vào tệp với tên `timn.m`. Ngoài cửa sổ lệnh có thể gọi chương trình để kiểm tra kết quả:

```
>> timn
Gia tri tong S:
        1015
```

```
Gia tri N:
     14
```

Câu lệnh `switch`: có cấu trúc cơ bản như sau

```
switch switch_biểu_thức
case case_biểu_thức
     câu_lệnh, ..., câu_lệnh
case {case_biểu_thức1, case_biểu_thức2, ...}
     câu_lệnh, ..., câu_lệnh
otherwise
     câu_lệnh, ..., câu_lệnh
end
```

Ví dụ sử dụng câu lệnh `switch` để xem thời khóa biểu như sau

```
t = input('Nhap vao ngay thu trong tuan: ');
switch t
    case 2
        'Toan, Van, Anh'
    case 3
        'Van, S?, Dia'
    case 4
        'Toan, Sinh, Ly'
    case 5
        'Hoa, Anh, Ly'
    case 6
        'Tin, Su, Toan'
    case 7
        'Anh, Van, Hoa'
    otherwise
        'Nghi hoc'
end
```

Chương trình được lưu vào tệp với tên `thoikb.m`. Ngoài cửa sổ lệnh có thể gọi chương trình để kiểm tra kết quả:

```
>> thoikb
Nhap vao ngay thu trong tuan: 4
ans =
```

```
Toan, Sinh, Ly
```

2.4.4 Hàm *eval*

Hàm `eval` mở ra một số khả năng thú vị trong MATLAB trong việc đánh giá các xâu ký tự (string). Các ví dụ sau cho thấy rõ hoạt động của hàm `eval`.

```
>> clear
>> theCommands = ['x = 2.0;','y = 3.0;','z = x*y;','whos']
theCommands =
x = 2.0; y = 3.0; z = x*y; whos
>> eval(theCommands)
  Name            Size            Bytes  Class     Attributes
  theCommands      1x31              62  char
  x                1x1                8  double
  y                1x1                8  double
  z                1x1                8  double
>> z
z =
     6
```

Trong ví dụ trên biến `theCommands` chứa một chuỗi lệnh MATLAB được định nghĩa như một xâu ký tự. Xâu ký tự này được dịch thành chuỗi lệnh MATLAB qua hàm `eval` sau đó.

Thêm một ví dụ về cách sử dụng hàm `eval`:

```
>> x = 1/6;
>> digs = num2str(2);
>> post = num2str(7);
>> printcommand = …
['fprintf(''%',digs,'.',post,'f\n'',',num2str(x),')'];
>> eval(printcommand)
0.1666700
```

Trong ví dụ trên hàm `fprintf` được sử dụng để định dạng dữ liệu vào một tệp hoặc ghi lên màn hình giá trị một số thập phân với 7 số sau dấu phẩy. Để thực hiện hoạt động này có thể sử dụng lệnh đơn giản sau:

```
>> fprintf('%2.7'f\n', x)
```

Tuy nhiên trong nhiều trường hợp lệnh thực hiện ở dạng xâu ký tự do vậy cần sử dụng hàm `eval` để thực hiện. Ví dụ trên cũng cho thấy hoạt động của một số hàm chuyển đổi từ định dạng giá trị số sang dạng xâu ký tự `num2str`. Biến `printcommand` là một biến ký tự.

2.4.5 Điều khiển hàm

Trong một số trường hợp xác định nó phù hợp hoặc thậm chí bắt buộc các hàm được truyền tới các hàm khác qua danh sách các tham số đầu vào. Kỹ thuật này có ý nghĩa trong các ứng dụng tính toán số sử dụng MATLAB cho các bài toán kỹ thuật.

Trong MATLAB, các hàm có thể được truyền tới các hàm MATLAB khác trong danh sách các tham số như là các điều khiển (hay trỏ) hàm. Một điều khiển hàm (function handle) tương ứng với một con trỏ tới hàm và các kết quả khi dấu @ được đặt trước hàm liên quan. Các điều khiển hàm mô tả một kiểu dữ liệu cụ thể trong MATLAB. Việc xử lý của các điều khiển hàm cũng đã trải qua sự thay đổi nhiều từ phiên bản MATLAB 7 so với các phiên bản trước. Các điều khiển hàm giờ đây có thể được sử dụng theo cùng một cách giống như hàm.

Một số ví dụ đơn giản dưới đây cho thấy hoạt động của điều khiển hàm:

```
>> FH_Sin = @sin;           % Định nghĩa điều khiển hàm
>> value = sin(2)
value =
    0.9093
>> value = FH_Sin(2)  % Tính giá trị hàm qua điều khiển hàm
value =
    0.9093
```

Trong ví dụ trên điều khiển hàm `FH_Sin` là một con trỏ chỉ ra mã hàm *sin*. Như vậy hàm này sẽ được thực hiện khi `FH_Sin` được gọi.

Đối với các phiên bản MATLAB trước, việc gọi hàm được thực hiện qua sự trợ giúp của hàm `feval` như sau:

```
>> value = feval(FH_Sin, 2)
value =
```

```
    0.9093
```

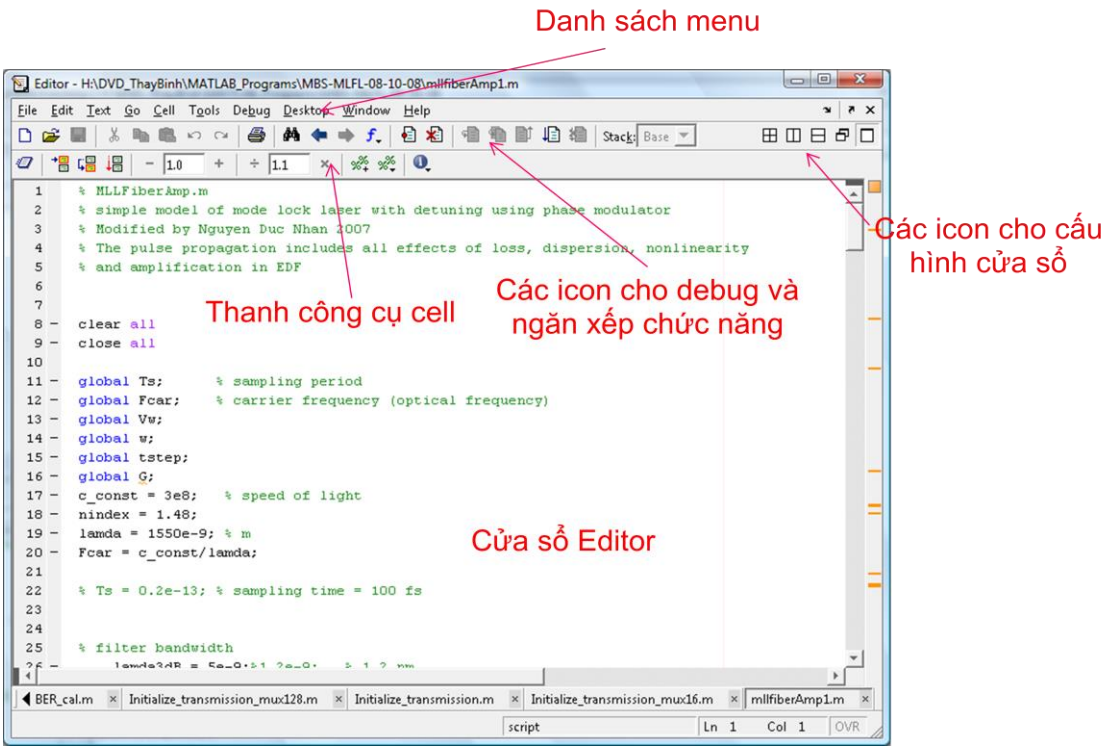
Các ví dụ về sử dụng điều khiển hàm sẽ còn được cho thấy trong các phần tiếp theo khi giải các bài toán trong kỹ thuật.

2.5 MATLAB Editor và Debugger

2.5.1 Các chức năng Editor

Như đã đề cập việc lập trình trong MATLAB được thực hiện trên cửa sổ giao diện `Editor` được mở qua mục `File - New` hoặc `File - Open`. Cửa sổ này cũng được mở ra khi kích đúp vào tệp m-file trong cửa sổ quản lý thư mục `Current directory`. Giao diện của Editor gồm các thành phần cơ bản được cho trong hình 2-16.

Các tham số của cửa sổ `Editor` có thể được thiết lập bằng việc sử dụng `Preferences`. Các cell mô tả một sự cải tiến quan trọng, đó là các phần được phân cách bởi một dòng chú thích bắt đầu bằng hai dấu %%. Con trỏ khi đặt ở trong một cell thì cell đó sẽ được làm nổi bật lên. Do đó các cell rất hữu ích trong việc xây dựng và phát triển một chương trình. Bằng việc sử dụng menu Cell hoặc các thanh công cụ cell, các cell có thể được đánh giá. Cửa sổ bộ Editor cũng có thể được phân chia trong trường hợp làm việc với nhiều tệp.

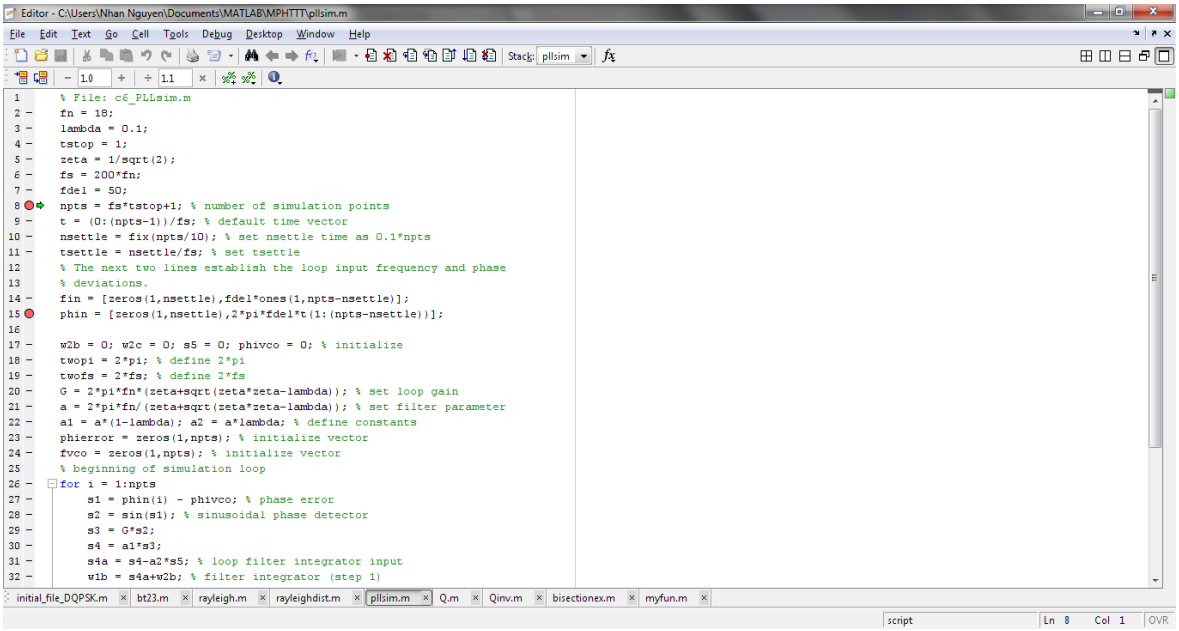


Hình 2-16 Giao diện cửa sổ `Editor`

2.5.2 Các chức năng Debugger

Về cơ bản có hai lỗi thường gặp phải trong quá trình phát triển chương trình đó là: các lỗi cú pháp và các lỗi thời gian chạy. Các lỗi cú pháp thường dễ khắc phục vì người lập trình sẽ được thông báo bằng một bản tin lỗi thích hợp. Các lỗi thời gian chạy trong một chương trình chính xác về cú pháp trong quá trình thực hiện và biểu hiện hoặc như một sự kết thúc chương trình đột ngột hoặc cho kết quả sai. Trong những trường hợp này nó cần thiết kiểm tra cẩn thận quá trình chạy của chương trình khi thực hiện. Nhiệm vụ này được thực hiện bởi công cụ gỡ rối (debugger) một chương trình chứa các điểm dừng (breakpoints) tại các vị trí cụ thể mà khi thực hiện chương trình đang được phát triển có thể được tạm dừng để kiểm tra.

Việc đặt các điểm breakpoint có thể được thực hiện thông qua thanh công cụ cho thấy trong hình 2-17 hoặc vào menu Debug. Các điểm breakpoint sẽ hiển thị như những hình tròn mầu đỏ tại vị trí được thiết lập như cho thấy trong hình 2-17. Các điểm này dễ dàng được loại bỏ bằng cách bấm trỏ chuột trực tiếp vào vị trí điểm đó.



Hình 2-17 Giao diện bộ gỡ rối debugger.

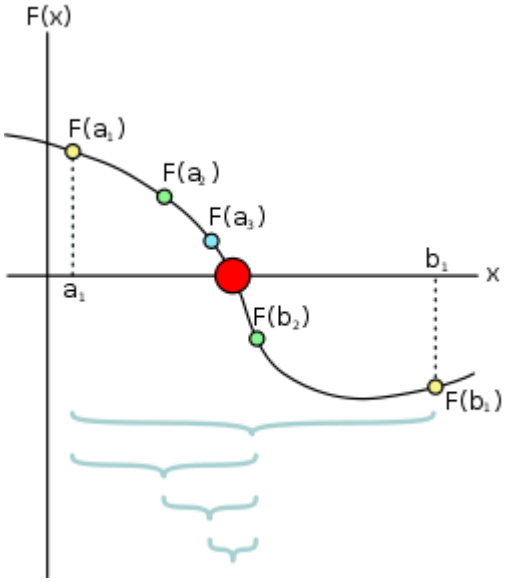
Bên cạnh công cụ debugger, cũng có các công cụ khác để hỗ trợ phát triển chương trình: như kiểm tra mã bằng M-Lint và profiler. Các công cụ này có thể được thực hiện qua menu Tools.

2.6 Một số phương pháp số sử dụng MATLAB

Các ứng dụng quan trọng nhất của MATLAB là giải các bài toán kỹ thuật bằng phương pháp số. Trong phần này một số phương pháp số cơ bản sử dụng MATLAB sẽ được giới thiệu.

2.6.1 Phương pháp tìm nghiệm

Tìm nghiệm của phương trình là một trong những bài toán cơ bản trong kỹ thuật. Có một số phương pháp để tìm nghiệm gần đúng của một phương trình như phương pháp chia nửa (bisection), phương pháp Newton, phương pháp nội suy,v.v. Trong giới hạn nội dung, phương pháp tìm nghiệm đơn giản chia nửa được trình bày như một ví dụ về phương pháp tìm nghiệm phương trình gần đúng.



Hình 2-18 Mô tả phương pháp chia nửa để tìm nghiệm phương trình

Phương pháp chia nửa được áp dụng để giải phương trình $f(x) = 0$ đối với biến thực x, trong đó f là một hàm liên tục trên một khoảng [a, b]. Trong trường hợp này khoảng [a, b] phải thỏa mãn chứa một nghiệm cần tìm hay nói cách khác f(a) và f(b) trái dấu nhau. Giải thuật của phương pháp được thực hiện như sau:

- Tính điểm giữa bằng việc chia nửa khoảng tìm nghiệm: c = (a + b)/2, và tính giá trị hàm tại điểm giữa f(c).
- Kiểm tra xem nếu f(c) đủ nhỏ trong phạm vi cho phép thi dừng vòng lặp và trả về nghiệm tìm được x_0 = c. Nếu không tiếp tục thực hiện bước tiếp theo.

- Kiểm tra dấu của *f*(c) và thay thế a hoặc b bằng c đảm bảo rằng khoảng [a, b] mới phải chứa nghiệm.
- Lặp lại các bước cho đến khi khoảng tìm nghiệm đủ nhỏ hay sai số tương đối (b - a)/b < ε với ε là sai số cho phép.

Chương trình MATLAB thực hiện giải thuật chia nửa để tìm nghiệm phương trình được cho trong ví dụ dưới đây.

```
% Ví dụ về giải thuật tìm nghiệm dùng pp chia nửa.
% myfun(x) là hàm chứa phương trình f(x) cần tìm nghiệm
a = 0; fa = myfun(a);
b = 3; fb = myfun(b);

while (b-a)/b > eps
    x = (a+b)/2;
    fx = myfun(x);
    if abs(fx) < eps
        break;
    else
        if sign(fx) == sign(fa)
            a = x; fa = fx;
        else
            b = x; fb = fx;
        end
    end
end
disp('Nghiem can tim:');
disp(x);

% Hàm myfun
function y = myfun(x)
% Hàm cần tìm nghiệm
y = x.^3-2*x-5;
```

Trong MATLAB có một số hàm sử dụng cho mục đích tìm nghiệm như fzero với cách sử dụng đơn giản như sau:

`x = fzero(@myfun,x0)`, trong đó `x0` là giá trị dự đoán ban đầu.

Nếu *f*(*x*) là một đa thức thì có thể sử dụng hàm `roots` để tìm các nghiệm của đa thức đó theo cách sau:

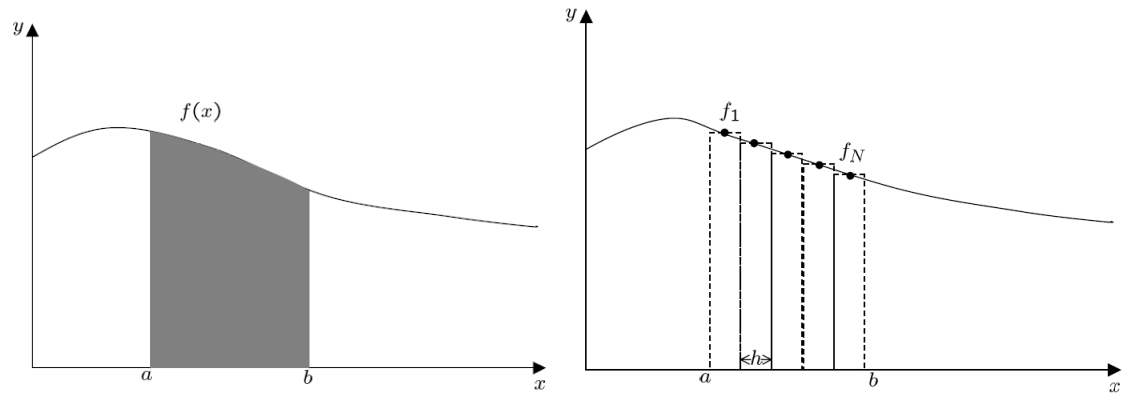
`x = roots(p)` với `p` là vector chứa các hệ số của đa thức.

2.6.2 Phương pháp tích phân

Tích phân số là phép tính tích phân giới hạn sau bằng phương pháp số:

$$I = \int_a^b f(x)dx \qquad (2.1)$$

trong đó $f(x)$ là hàm lấy tích phân và a và b được gọi là giới hạn dưới và trên của tích phân. Ý nghĩa toán học của phép tính tích phân cho biết diện tích bị giới hạn bởi đường cong $f(x)$ trong khoảng [a, b] như cho thấy trong hình 2-19. Có một số phương pháp số có thể được sử dụng để tính gần đúng tích phân này như phương pháp điểm giữa (midpoint), phương pháp hình thang và phương pháp Simpsons, v.v.



Hình 2-19 Tích phân giới hạn (a) và phương pháp điểm giữa để tính gần đúng (b).

Trong giới hạn nội dung, phương pháp điểm giữa được trình bày như một ví dụ để tính gần đúng tích phân. Để xác định được công thức tính gần đúng của phương pháp này, khoảng lấy tích phân $a \le x \le b$ được chia thành N đoạn bằng nhau có độ rộng là h với $h = (b - a)/N$. Đặt $x_j = a + (j - 1/2)h$ với $j = 1, \ldots, N$, và đặt $f_j = f(x_j)$ do đó f_j là các giá trị hàm của $f(x)$ tại điểm giữa của các đoạn. Khi đó diện tích dưới đường cong $f(x)$ giữa a và b được tính gần đúng bởi diện tích của N hình chữ nhật

$$\int_a^b f(x)dx \cong \sum_{j=1}^{N} hf_j = h\sum_{j=1}^{N} f_j \qquad (2.2)$$

Như vậy công thức (2.2) được sử dụng để tính gần đúng tích phân và độ chính xác của phép tích phân phụ thuộc vào độ rộng mỗi đoạn h hay số lượng đoạn N.

Một hàm tính tích tích phân được xây dựng trong MATLAB dựa trên phương pháp điểm giữa được cho trong đoạn mã dưới đây trong đó hàm số để lấy tích phân được trỏ tới bằng sử dụng điều khiển hàm:

```
% Ví dụ về tính tích phân giới hạn dùng pp điểm giữa.
function [integral] = midpoint(a, b, F, N)
% Function midpoint
% ví dụ cách dùng: integ = midpoint(0, 2, @myfun, 10)

h=(b-a)/N;                      % Độ rộng mỗi đoạn (cỡ bước)
integral = F(a+h/2);            % F tại đoạn đầu tiên

for k=2:N
    xi = a + (k-1/2)*h;         % Tính điểm giữa mỗi đoạn
    integral = integral+F(xi);
end;

integral = integral*h;          % Chuẩn hóa theo h
```

Trong thư viện của MATLAB, có một số hàm để sử dụng tính tích phân ví dụ hàm `quad` và `trapz`:

```
I = quad(f,a,b) % thực hiện tích phân hàm f trong [a, b]

Z = trapz(X,Y)   % tính gần đúng tích phân của Y theo X
```

2.6.3 Phương pháp giải phương trình vi phân

Trong bài toán mô hình hóa, một hệ thống thường được mô tả bởi một phương trình vi phân. Các tham số của mô hình sẽ là hàm của thời gian $y(t)$, gồm cả vận tốc dy/dt và gia tốc d^2y/dt^2. Trong trường hợp hầu hết các mô hình động biểu diễn bằng các phương trình vi phân bậc hai có thể được rút gọn về thành các phương trình vi phân bậc một.

Trong các ứng dụng kỹ thuật, một phương trình vi phân thường (ODE) bậc một thường thấy trong bài toán giá trị ban đầu, tức tham số y có giá trị cụ thể y_0 tại vị trí ban đầu x_0 (hoặc tại thời điểm ban đầu t_0). Các giá trị của y tại các vị trí khác (hoặc tại các thời điểm khác sau đó) được xác định từ điều kiện ban đầu qua phương trình ODE bậc một. Một cách tổng quát, bài toán ODE bậc một được mô tả đầy đủ bởi

$$y(x_0) = y_0 \tag{2.3}$$

$$\frac{d}{dx} y(x) = f\big(x, y(x)\big) \tag{2.4}$$

Hàm f có thể là một hàm tuyến tính hoặc phi tuyến của biến độc lập x và tham số phụ thuộc y. Có một số phương pháp số để giải các ODE bậc một.

Phương pháp Euler: Cách tiếp cận đơn giản nhất cho giải bài toán ODE bằng phương pháp số được dựa trên phép tính gần đúng đạo hàm

$$y(x+h) \approx y(x) + hy^{(1)}(x) \tag{2.5}$$

Nếu $x_1 = x_0 + h$ thì đạo hàm từ phương trình vi phân được thay thế trong công thức sai phân hữu hạn để có

$$y_1 = y_0 + hf(x_0, y_0) \tag{2.6}$$

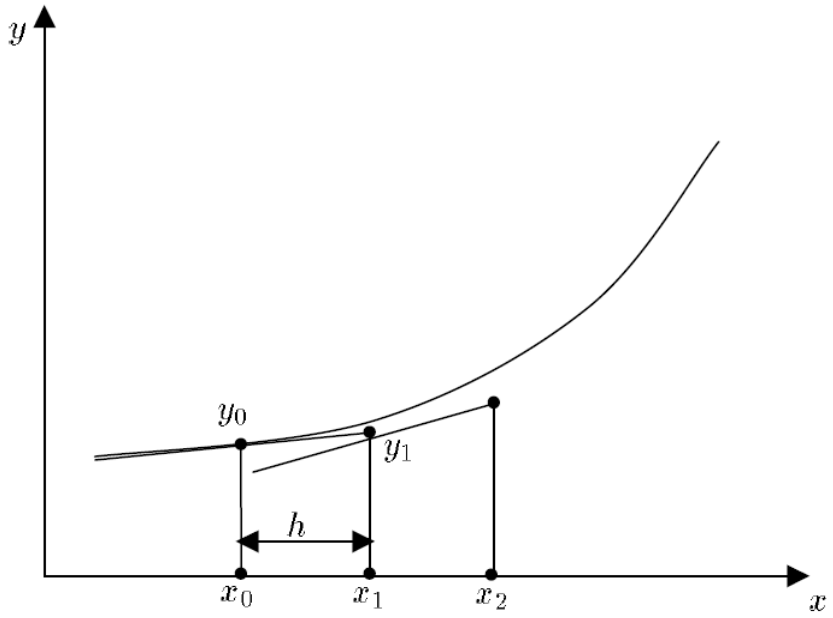
tính gần đúng $y(x_1) = y(x_0 + h)$. Tương tự, $y(x_2) = y(x_1 + h) = y(x_0 + 2h)$ có thể tính gần đúng bằng

$$y_2 = y_1 + hf(x_1, y_1) \tag{2.7}$$

Một cách tổng quát:

$$y_{n+1} = y_n + hf(x_n, y_n) \tag{2.8}$$

với $n = 0, 1, 2, \ldots$ trong đó y_{n+1} là phép tính gần đúng cho $y(x_{n+1})$ và $x_{n+1} = x_0 + (n+1)h$. Phương pháp Euler có thể xem là phương pháp tính gần đúng đường cong $y(x)$ bằng một đa giác mà cạnh đầu tiên chính là tiếp tuyến của đường cong tại x_0.



Hình 2-20 Mô tả phương pháp Euler

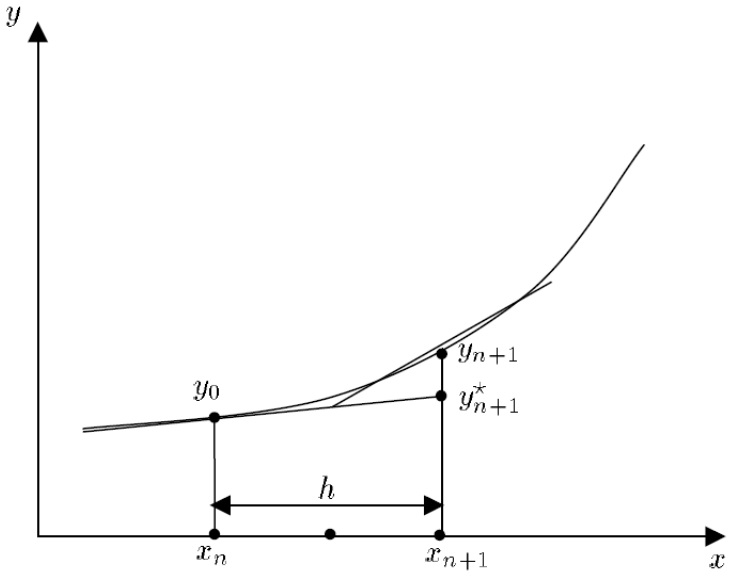
```
% Ví dụ về giải thuật giải ODE dùng pp Euler.
% Fdot - hàm vi phân cần giải
x(1) = x0;              % Giá trị ban đầu
y(1) = y0;              % Nghiệm ban đầu

for k = 2:N             % N - Số lượng bước tính
    x(k) = x(k-1) + h;
    y(k) = y(k-1)+Fdot(x(k-1),y(k-1))*h; % Tính điểm kế tiếp
end;
```

Phương pháp Euler biến đổi: Do phương pháp Euler tính nghiệm của điểm tiếp theo dựa vào độ dốc của điểm trước đó nên hạn chế về độ chính xác. Để cải thiện độ chính xác, một phương pháp Euler biến đổi được sử dụng để ước tính nghiệm kế tiếp dựa trên độ dốc tại điểm giữa mỗi khoảng tính bằng cách tính độ dốc trung bình giữa hai điểm x_n và x_{n+1} như mô tả trong hình 2-21. Tuy nhiên do nghiệm tại điểm kế tiếp chưa biết nên cần phải tính giá trị dự đoán hay tạm thời bằng phương pháp Euler dựa trên giá trị điểm hiện tại

$$y_{n+1}^{*} = y_n + hf(x_n, y_n) \tag{2.9}$$

Sử dụng giá trị dự đoán này để ước tính độ dốc trung bình giữa hai điểm để tính nghiệm cuối cùng y_{n+1} theo công thức sau

$$y_{n+1} = y_n + \frac{h}{2}\left(f(x_n, y_n) + f(x_{n+1}, y_{n+1}^{*})\right) \tag{2.10}$$



Hình 2-21 Mô tả phương pháp Euler biến đổi

Như vậy giải thuật để tính nghiệm ODE bằng phương pháp Euler biến đổi được thực hiện theo hai bước: bước 1 là tính nghiệm dự đoán bằng phương pháp Euler, bước 2 là tính nghiệm kế tiếp dựa trên công thức Euler biến đổi (2.10). Đoạn mã giải thuật của phương pháp Euler biến đổi thực hiện trong MATLAB được cho dưới đây.

```
% Ví dụ về giải thuật giải ODE dùng pp Euler biến đổi.
% fdot là hàm phương trình vi phân cần tìm nghiệm
x(1) = x0;
y(1) = y0;
for k=2:N
  x(k) = x(k-1) + h;
  ytemp = y(k-1)+fdot(x(k-1),y(k-1))*h; % Tính nghiệm dự đoán
  % Tính nghiệm điểm tiếp theo
  y(k) = y(k-1)+h*(fdot(x(k-1),y(k-1))+fdot(x(k),ytemp))/2;
end;
```

Phương pháp Runge-Kutta: Bao gồm một họ phương pháp Runge-Kutta để tính nghiệm gần đúng. Cách tiếp cận của phương pháp R-K cũng dựa trên ước tính độ dốc của nhiều điểm để cải thiện độ chính xác khi ước tính nghiệm kế tiếp. Một cách tổng quát phương pháp R-K nghiệm kế tiếp được xác định bởi

$$y_{n+1} = y_n + \sum_{i=1}^{S} b_i k_i \tag{2.11}$$

trong đó

$$\begin{aligned} k_1 &= hf(x_n, y_n), \\ k_2 &= hf(x_n + c_2 h, y_n + a_{21} k_1), \ \ldots \\ k_S &= hf(x_n + c_S h, y_n + a_{S1} k_1 + a_{S2} k_2 + \ldots + a_{S,S-1} k_{S-1}), \end{aligned} \tag{2.12}$$

ở đây S xác định số tầng tính toán hay bậc của phương pháp, ma trận các hệ số a_{ij} (cho $1 \le j < i \le S$) được gọi là ma trận Runge-Kutta, b_i (cho $i = 1, 2, \ldots, S$) và c_i (cho $i = 2, 3, \ldots, S$) là các trọng số và các điểm nút. Các hệ số thường có quan hệ $\sum_{i=1}^{S} b_i = 1$ và $\sum_{j=1}^{i-1} a_{ij} = c_i$. Tùy thuộc vào giá trị cụ thể của S và các hệ số tương ứng ta có các phương pháp Runge-Kutta khác nhau.

- Phương pháp Runge-Kutta bậc 2: Khi S = 2 ta có phương pháp R-K bậc 2. Với $b_1 = b_2 = ½$ và $c_2 = a_{21} = 1$ ta có công thức tính nghiệm giống với phương pháp Euler biến đổi. Với $b_1 = 0$, $b_2 = 1$ và $c_2 = a_{21} = ½$ ta có công thức tính nghiệm gần đúng theo phương pháp điểm giữa. Các hệ số khác cũng hay được sử dụng như $b_1 = ¼$, $b_2 = ¾$ và $c_2 = a_{21} = 2/3$.
- Phương pháp Runge-Kutta bậc 3: Khi S = 3 ta có phương pháp R-K bậc 3. Một ví dụ cụ thể phương pháp R-K bậc 3 có các hệ số $b_1 = 2/8$, $b_2 = b_3 = 3/8$ và $c_2 = c_3 = a_{21} = a_{32} = 2/3$, $a_{31} = 0$.

- Phương pháp Runge-Kutta bậc 4: Khi S = 4 ta có phương pháp R-K bậc 4. Các hệ số cụ thể cho phương pháp R-K bậc 4 là $b_1 = b_4 = 1/6$, $b_2 = b_3 = 1/3$ và $c_2 = c_3 = ½$, $c_4 = 1$, $a_{21} = a_{32} = ½$, $a_{43} = 1$ và $a_{31} = a_{41} = a_{42} = 0$. Cụ thể

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{2.13}$$

với

$$\begin{aligned} k_1 &= hf(x_n, y_n), \\ k_2 &= hf(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_1), \\ k_3 &= hf(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_2), \\ k_4 &= hf(x_n + h, y_n + k_3), \end{aligned} \tag{2.14}$$

Như vậy dựa trên các công thức tính nghiệm gần đúng trong các phương pháp Runge-Kutta, các giải thuật dễ dàng được thực hiện trong MATLAB.

Trong MATLAB cũng sẵn có các hàm để sử dụng giải các phương trình vi phân ODE được cho trong bảng dưới đây:

Solver	Solves These Kinds of Problems	Method
ode45	Nonstiff differential equations	Runge-Kutta
ode23	Nonstiff differential equations	Runge-Kutta
ode113	Nonstiff differential equations	Adams
ode15s	Stiff differential equations and DAEs	NDFs (BDFs)
ode23s	Stiff differential equations	Rosenbrock
ode23t	Moderately stiff differential equations and DAEs	Trapezoidal rule
ode23tb	Stiff differential equations	TR-BDF2
ode15i	Fully implicit differential equations	BDFs

Cách sử dụng cơ bản các hàm này như sau:

```
[T,Y] = solver(odefun,tspan,y0)
```

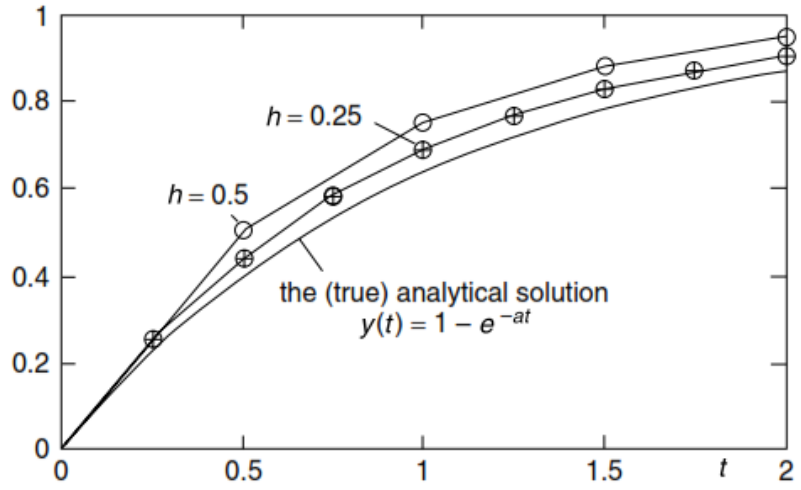
ở đây `solver` là tên hàm sử dụng để giải, `odefun` là tên hàm ODE cần giải, `tspan` là khoảng giá trị để giải ODE, `y0` là giá trị ban đầu.

Ví dụ : `[t,y] = ode45(@myfun,[0 3],0);`

Ví dụ 2.1: Ví dụ giải phương trình vi phân ODE bậc 1 $y'(t) + ay(t) = r$ với điều kiện ban đầu $y(0) = y_0$ bằng phương pháp Euler tại các kích cỡ bước tính khác

nhau. Biết nghiệm giải tích của phương trình này là $y(t) = \left(y_0 - \frac{r}{a}\right)e^{-at} + \frac{r}{a}$ sử dụng để đánh giá độ chính xác của phương pháp. Cho $r = 1$, $a = 1$ và $y_0 = 0$, phương trình trên được giải bằng phương pháp Euler được cho trong đoạn mã dưới đây và kết quả cho thấy trong hình 2-22:

```
% Ví dụ về giải ODE bậc 1 dùng pp Euler.
clear, clf  % Xóa các biến và hình vẽ cũ
a=1;r=1;y0=0; tf=2;
t = [0:0.01:tf]; yt=1-exp(-a*t);     % tính nghiệm giải tích
plot(t,yt,'k'), hold on              % vẽ đường cong giải tích
N = [8 4 2]; hs = tf./N;       % tính cỡ bước
y(1) = y0;
for itr = 1:3 % lặp cho các cỡ bước khác nhau h = 1/8,1/4,1/2
    klast = N(itr); h = hs(itr); y(1)=y0;
    for k = 1:klast
        y(k + 1) = (1 - a*h)*y(k) +h*r; % công thức Euler
        plot([k - 1 k]*h,[y(k) y(k+1)],'b',k*h,y(k+1),'ro') % vẽ kquả
        if k<4,
           pause;
        end
    end
end
```



Hình 2-22 Kết quả giải phương trình vi phân cho trong ví dụ 2.1.

Ví dụ 2.2: Sử dụng hàm `ode23` để giải bài toán dao động con lắc được mô tả bởi phương trình vi phân bậc hai sau:

$$\alpha''(t) = -\frac{g}{l}.\sin(\alpha(t)), \text{ với g = 9,81 m/s}^2 \qquad (2.15)$$

Ở đây $\alpha(t)$ là góc của con lắc có độ dài l tạo thành tại thời điểm t so với vị trí cân bằng của nó (xem hình 2-23). Để giải phương trình (2.15) đòi hỏi cần hai điều kiện ban đầu

$$\vec{\alpha}(0) = \begin{pmatrix} \alpha(0) \\ \alpha'(0) \end{pmatrix},$$

xác định vị trí bắt đầu (độ lệch) của con lắc và vận tốc ban đầu của nó. Phương trình (2.15) cần được chuyển đổi thành hệ phương trình vi phân bậc 1 trước khi áp dụng phương pháp số để giải. Việc chuyển đổi được thực hiện bằng cách đặt:

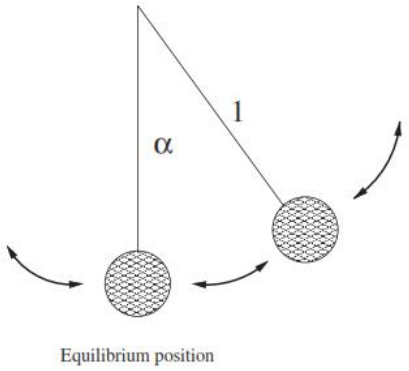
$$\alpha_1(t) = \alpha(t)$$
$$\alpha_2(t) = \alpha'(t)$$

Khi đó phương trình (2.15) được viết lại theo dạng:

$$\alpha_1'(t) = \alpha_2(t)$$
$$\alpha_2'(t) = -\frac{g}{l}.\sin(\alpha_1(t)) \qquad (2.16)$$

với điều kiện ban đầu

$$\vec{\alpha}(0) = \begin{pmatrix} \alpha(0) \\ \alpha'(0) \end{pmatrix} = \begin{pmatrix} \alpha_1(0) \\ \alpha_2(0) \end{pmatrix},$$



Hình 2-23 Bài toán dao động con lắc.

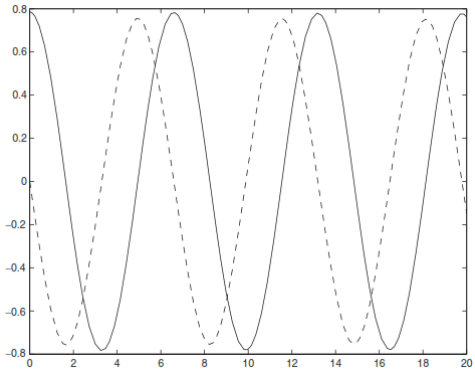
Trước khi sử dụng hàm giải phương trình vi phân, một hàm được xây dựng với tên tệp `pendde.m` để mô tả hệ phương trình vi phân ở dạng vector như sau:

```
% Định nghĩa hàm mô tả hệ phương trình vi phân bậc 1 dạng vector.
function [alphadot] = pendde(t,alpha)
% Định nghĩa các hằng số
l = 10;            % độ dài con lắc
g = 9.81;          % gia tốc trọng trường m/s
% Khởi tạo
alphadot = [0;0];
% Mô tả hệ phương trình vi phân
alphadot(1) = alpha(2);
alphadot(2) = -(g/l)*sin(alpha(1));
```

Sau đó có thể gọi hàm `ode23` để giải hệ phương trình với điều kiện ban đầu cho là $\alpha(0) = \pi/4$ và $\alpha'(0) = 0$ trong khoảng [0, 20] như sau:

```
>> [t,solution]=ode23(@pendde,[0 20],[pi/4, 0]);
>> plot(t,solution(:,1),'r',t,solution(:,2),'b:');
```

Nghiệm tìm được trả về trong biến `solution` là một ma trận gồm 2 cột trong đó cột 1 chứa nghiệm của phương trình cần giải ($\alpha(t)$), còn cột 2 chứa đạo hàm của nó ($\alpha'(t)$) theo đúng cấu trúc được định nghĩa trong hàm `pendde.m`. Kết quả bài toán được cho thấy trong hình 2-24.



Hình 2-24 Kết quả giải phương trình vi phân bậc 2 trong ví dụ 2.2.

2.7 Tổng kết chương

Chương này đã giới thiệu các cấu trúc cơ bản của MATLAB bao gồm các kiểu biến, các phép tính dựa trên vector và ma trận. Việc xây dựng chương trình MATLAB có cấu trúc cơ bản như các ngôn ngữ lập trình khác, nhưng đơn giản và dễ dàng thực hiện hơn. Người sử dụng cũng dễ dàng xây dựng thư viện các hàm tính toán theo mục đích riêng. Với mục đích sử dụng MATLAB như một công cụ tính toán trong kỹ thuật nên một số phương pháp số để giải các bài toán hay gặp trong kỹ thuật như tìm nghiệm, tính tích phân và giải phương trình vi phân thường. Những phương pháp số này không chỉ giúp sinh viên có tư duy về phương pháp tính dựa trên máy tính ma cũng rất quan trọng trong mô phỏng hệ thống truyền thông.

Câu hỏi/bài tập chương 2

1/ Tạo các vectơ và ma trận trong MATLAB với các biến:

$$M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & j & 1 \\ j & j+1 & -3 \end{pmatrix},$$
$$k = 2.75,$$
$$\vec{v} = \begin{pmatrix} 1 \\ 3 \\ -7 \\ -0{,}5 \end{pmatrix},$$
$$\vec{w} = \begin{pmatrix} 1 & -5.5 & -1.7 & -1.5 & 3 & -10.7 \end{pmatrix},$$
$$\vec{y} = \begin{pmatrix} 1 & 1.5 & 2 & 2.5 & \cdots & 100.5 \end{pmatrix}.$$

2/ Khai triển ma trận M thành ma trận V 6x6:

$$V = \begin{pmatrix} M & M \\ M & M \end{pmatrix}.$$

Xóa hàng 2 và cột 3 từ ma trận *V*

Tạo vectơ z từ hàng 4 của ma trận *V*

Biến đổi giá trị tại *V*(4,2) thành j+5

3/ Dùng hoạt động ma trận để biến đổi từ ma trận A thành ma trận C

$$A = \begin{pmatrix} -1 & 3.5 & 2 \\ 0 & 1 & -1.3 \\ 1.1 & 2 & 1.9 \end{pmatrix}. \qquad C = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1.9 \end{pmatrix}$$

4/ Cho ma trận:

$$C = \begin{pmatrix} 1 & 2 & 3 & 4 & 10 \\ -22 & 1 & 11 & -12 & 4 \\ 8 & 1 & 6 & -11 & 5 \\ 18 & 1 & 11 & 6 & 4 \end{pmatrix}.$$

Sử dụng các toán tử quan hệ để chuyển đổi các số hạng trong ma trận có giá trị > 10 và < -10 bằng 0.

5/ Tính giá trị của tín hiệu:

$$s(t) = \sin(2\pi 5t)\cos(2\pi 3t) + e^{-0.1t}$$

với vectơ thời gian từ 0 đến 10 có cỡ bước 0,1.

6/ Cho vectơ tần số: $\omega = (0.01,\ 0.02, 0.03,\ 0.04, \ldots,\ 5)\ \ \mathrm{rad/s},$ và các hàm truyền của một bộ tích phân và của một phần tử trễ thời gian bậc 1 tương ứng:

$$H(j\omega) = \frac{1}{j\omega} \qquad H(j\omega) = \frac{1}{1 + j\omega},$$

thường gặp trong xử lý tín hiệu và kỹ thuật điều khiển. Hãy vẽ đồ thị biên độ của các hàm truyền này trên 2 hình riêng biệt. Sử dụng các hàm semilogx, semilogy và loglog để thay đổi kết quả biểu diễn đồ thị theo các kiểu trục khác nhau. Xác định kiểu biểu diễn nào là tốt nhất.

7/ Tính và vẽ hàm $x^2 + y^2$ trong dải [-2,2]x[-1,1] sử dụng lưới có cỡ bước 0,2 theo chiều x và 0,1 theo chiều y.

8/ Vẽ hình cầu có bán kính R = 3.

9/ Viết một chương trình MATLAB có tên *circle_prog.m* để thực hiện các hoạt động sau: vẽ đường tròn có bán kính r = 3, trả về các kết quả tính chu vi và diện tích hình tròn.(Hint: sử dụng lệnh *axis equal* để hiển thị đồ thị tốt hơn).

10/ Thay đổi chương trình trên để hiển thị kết quả với 5 số sau dấu phẩy. (Hint: có thể dùng lệnh *sprintf*).

11/ Cho một hàm $f(x) = x^3/3+4x^2+x-6$ trong dải $-1<x<3$. Viết chương trình tìm nghiệm phương trình trên bằng phương pháp bisection với sử dụng 2 dự đoán ban đầu tại x = 0 và x = 3.

(Sử dụng lệnh *input* để cho phép nhập giá trị các tham số đầu vào từ bàn phím khi chạy chương trình)

12/ Viết mã chương trình sử dụng vòng lặp để tính tích phân:

$$h(x) = \int_{-1.5}^{1.5} 4x^3 2e^x Cos(x)dx$$

bằng phương pháp midpoint với số lượng điểm N = 100.

13/ Viết mã chương trình sử dụng vòng lặp while để tính gần đúng $\sqrt{2}$ dựa trên phương pháp Newton dùng hệ thức đệ quy:

$$x_{n+1} = \frac{x_n^2 + 2}{2 \cdot x_n}, \qquad x_0 = 2\,.$$

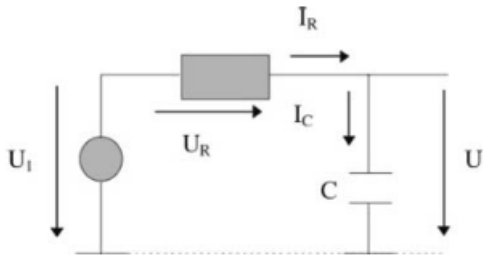
Quá trình lặp thực hiện cho đến khi x_n thay đổi chỉ 0.0001.

14/ Viết chương trình tìm nghiệm ptr vi phân: $y'(t) + ay(t) = r$ với a = 1, r = 1 và y(0) = 0 bằng phương pháp Euler biến đổi với cỡ bước h = 0.25. Xác định sai số so với nghiệm giải tích tại 2 thời điểm t = 1 và t = 2.

15/ Tương tự bài tập 14 nhưng sử dụng phương pháp RK bậc 3.

16/ Tương tự bài tập 14 nhưng sử dụng phương pháp RK bậc 4.

17/ Cho sơ đồ mạch RC hình dưới:



Điện áp đầu ra của hệ thống tuân theo ptr vi phân tuyến tính:

$$\frac{d}{dt}u(t) = -\frac{1}{RC}u(t) + \frac{1}{RC}u_1(t)\,.$$

Hãy viết chương trình tìm nghiệm của ptr này trong khoảng [0, 1] s bằng phương pháp RK bậc 4, biết C = 4.7μF và R = 10 kΩ. Hàm $u_1(t)$ là hàm bậc đơn vị. Sau đó so sánh kết quả với nghiệm thu được bằng việc sử dụng lệnh *ode45*.

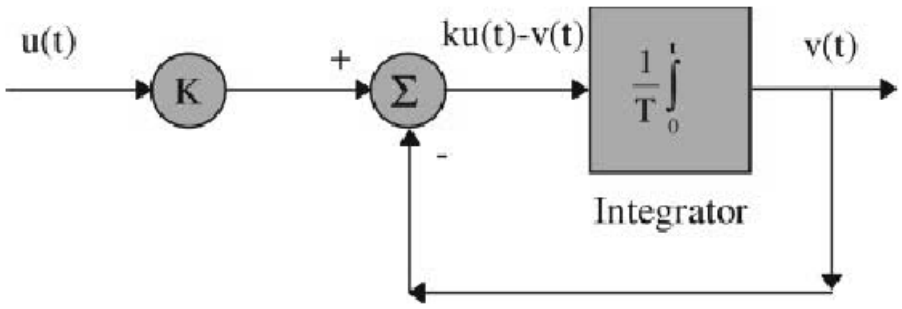
Chương 3 Giới thiệu về Simulink

Với sự phát triển mạnh trong lĩnh vực mô phỏng hệ thống, MATLAB đã đưa ra một bộ công cụ mô phỏng Simulink thuận tiện cho việc mô phỏng các hệ thống động. Chương này sẽ giới thiệu những đặc điểm cơ bản nhất của Simulink giúp sinh viên hiểu và áp dụng vào mô phỏng các hệ thống đơn giản từ đó có thể phát triển mô phỏng cho các hệ thống phức tạp hơn.

3.1 Giới thiệu chung

Simulink là một công cụ để mô phỏng các hệ thống động với giao diện đồ họa trực quan được phát triển đặc biệt cho mục đích này. Trong môi trường MATLAB, Simulink là một bộ công cụ MATLAB nhưng khác với các bộ công cụ khác cả về giao diện và kỹ thuật lập trình liên quan đến nó.

Trong các hệ thống động, các quá trình phụ thuộc thời gian tuyến tính hoặc phi tuyến đều có thể được mô tả bằng các phương trình vi phân. Bên cạnh đó các hệ thống này cũng được mô tả qua sơ đồ khối như ví dụ cho trong hình 3-1. Simulink dựa trên kiểu mô tả này và một giao diện đồ họa được sử dụng để chuyển đổi một sơ đồ khối thành mô hình Simulink và mô phỏng hoạt động của hệ thống. Do vậy để sử dụng hiệu quả Simulink đòi hỏi người dùng có kiến thức tốt về công nghệ điều khiển và lý thuyết hệ thống.

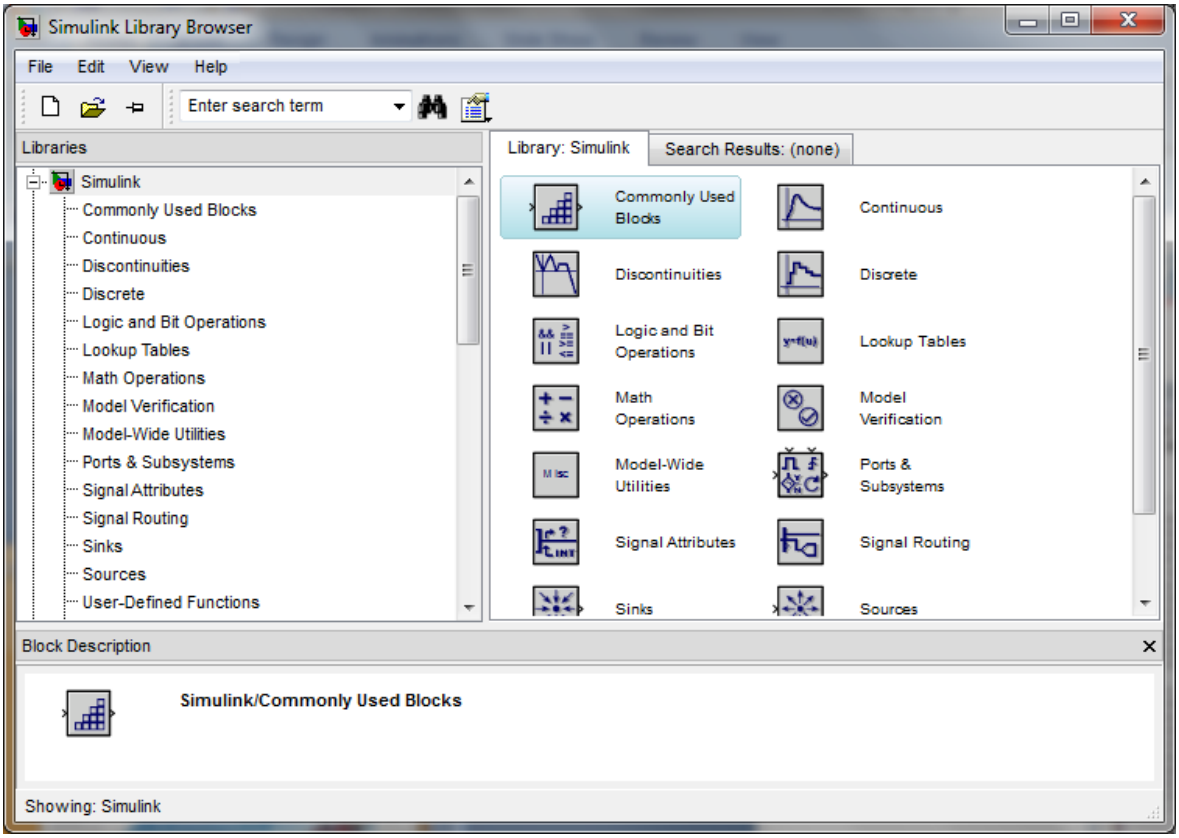


Hình 3-1 Mô tả một hệ thống động bằng sơ đồ khối

Simulink cho phép mô phỏng hệ thống dựa trên sơ đồ khối để tìm nghiệm phương trình vi phân mô tả nó. Nói cách khác một phương trình vi phân có thể được chuyển đổi thành một sơ đồ khối để giải bằng Simulink.

3.2 Nguyên lý hoạt động của Simulink

Chương trình Simulink có thể được khởi tạo bằng lệnh `simulink` hoặc lệnh `open_system('simulink.mld')` trong cửa sổ lệnh MATLAB. Sau khi thực hiện một cửa sổ trình duyệt thư viện các khối chức năng sẵn có trong Simulink cái thường được hiển thị ở dạng biểu tượng.



Hình 3-2 Cửa sổ giao diện trình duyệt thư viện các khối chức năng của Simulink

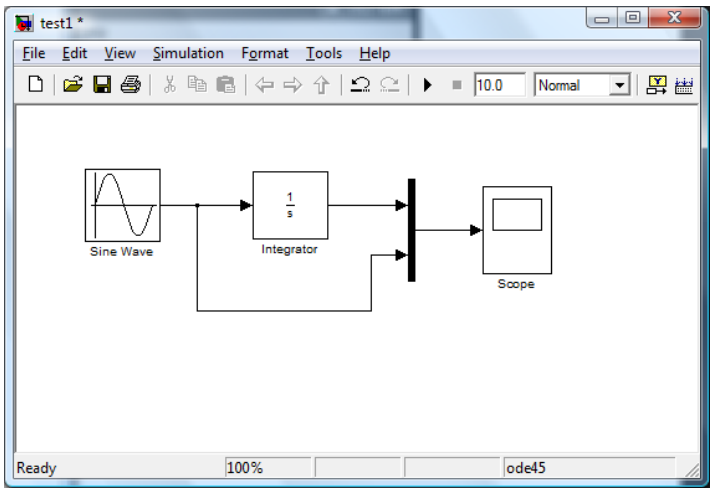
Thư viện các khối chức năng được tổ chức thành các nhóm chức năng cho mục đích mô phỏng khác nhau. Trong lĩnh vực mô phỏng hệ thống truyền thông, ngoài thư viện các khối thư viện chung Simulink các thư viện hay được sử dụng đó là `Communications Blockset`, `Signal Processing Blockset` và `RF Blockset`.

3.2.1 Xây dựng sơ đồ khối Simulink

Để xây dựng hệ thống mô phỏng bằng việc sử dụng thư viện các khối chức năng, trước hết cần mở một cửa sổ mới bằng việc lựa chọn `File - New Model` trong cửa sổ trình duyệt Simulink. Cũng có thể mở mô hình có sẵn bằng việc lựa

chọn File – Open. Mô hình mới được lưu lại dưới một tên phù hợp với đuôi mở rộng mdl (mdl=model) bằng việc lựa chọn File – Save As.

Xây dựng một hệ thống lấy tích phân làm ví dụ lưu trong tên tệp test1.mdl. Trước hết lấy khối nguồn sóng hình sine Sine Wave làm nguồn tín hiệu từ thư viện Sources bằng cách kéo khối đó vào cửa sổ mô hình được mở. Tiếp theo để tích phân tín hiệu, lấy khối Integrator từ thư viện Continuous, sau đó kết nối đầu ra của khối nguồn với đầu vào khối tích phân bằng sử dụng chuột. Khối tích phân có biểu tượng $\frac{1}{s}$ vì liên quan đến khai triển Laplace của quá trình tích phân. Hầu hết các khối chức năng tuyến tính được mô tả bởi khai triển Laplace hoặc khai triển z (đối với tín hiệu rời rạc). Để hiển thị tín hiệu và kết quả tích phân trên cùng một cửa sổ, khối ghép kênh Mux từ thư viện Signal Routing được sử dụng và khối Scope từ thư viện Sinks được lựa chọn để hiển thị. Tín hiệu nguồn và đầu ra khối tích phân được kết nối với 2 đầu vào khối ghép kênh và đầu ra sau đó nối với đầu vào khối Scope. Mô hình sau khi kết nối có giao diện như trong hình 3-3.



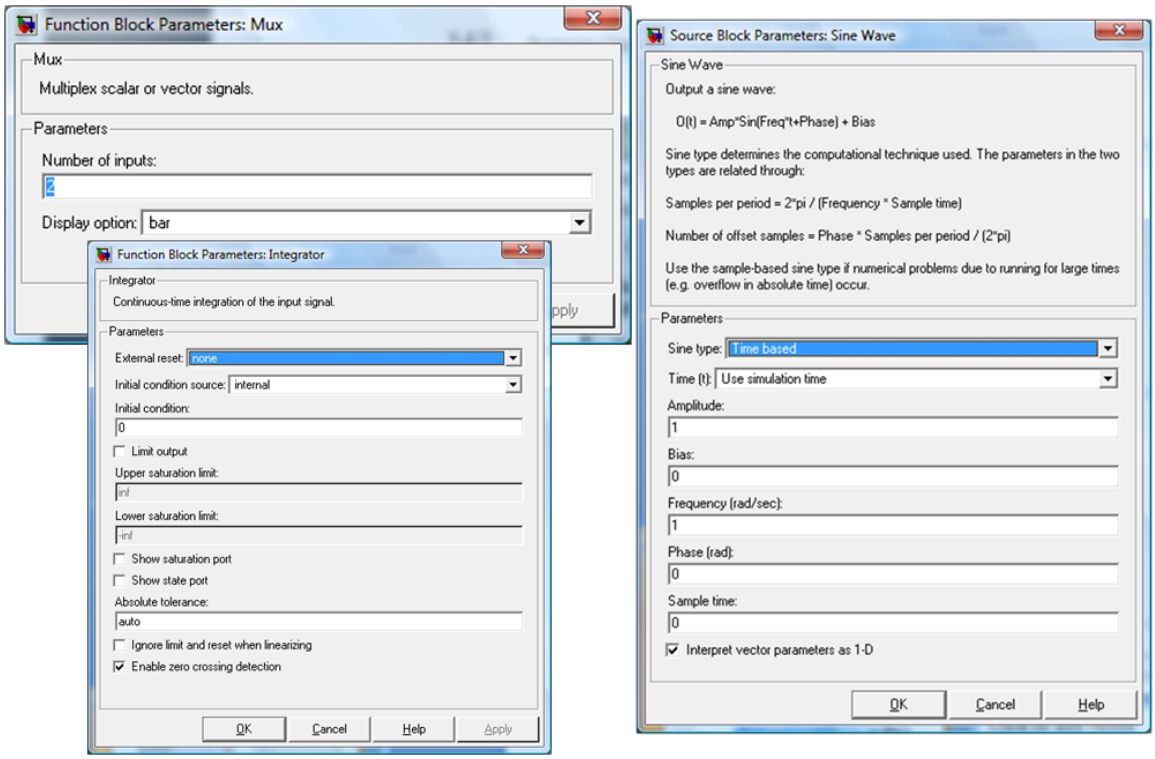
Hình 3-3 Mô hình ví dụ hệ thống tích phân test1.mdl

3.2.2 Tham số hóa các khối Simulink

Để thực hiện chạy mô hình mô phỏng Simulink trước hết các tham số của các khối sử dụng trong mô hình cần được thiết lập chính xác. Để thực hiện tốt bước này đòi hỏi người sử dụng phải hiểu rõ hệ thống đang mô phỏng và các tham số liên quan.

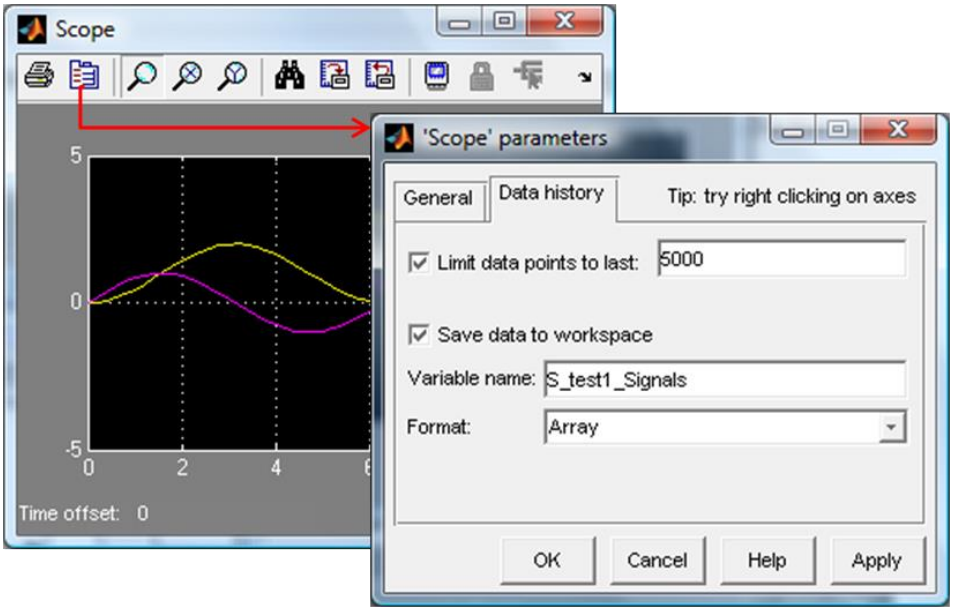
Thiết lập tham số cho khối chức năng được thực hiện bằng việc kích đúp khối chức năng đó và giao diện thiết lập tham số sẽ được hiện ra như cho thấy ví dụ

trong hình 3-4. Các giá trị phù hợp của các tham số sẽ được nhập vào trực tiếp hoặc lựa chọn các thuộc tính phù hợp sẵn có trong giao diện thiết lập của khối. Lấy ví dụ mô hình `test1.mdl`, các tham số khối nguồn sine có thể được thiết lập gồm tần số, biên độ và pha ban đầu của tín hiệu sine. Tuy nhiên có một lựa chọn quan trọng cần chú ý là kiểu tín hiệu Time based hoặc Sample based. Vì hệ thống tích phân xây dựng cho tín hiệu liên tục nên cần lựa chọn Time based cho phù hợp. Riêng đối với khối `Scope` để thiết lập tham số cần lựa chọn biểu tượng `Parameters` trên cửa số hiển thị để mở giao diện thiết lập như cho thấy trong hình 3-5. Tại cửa sổ này số lượng đồ thị hiển thị và lưu dữ liệu kết quả dễ dàng được thiết lập.

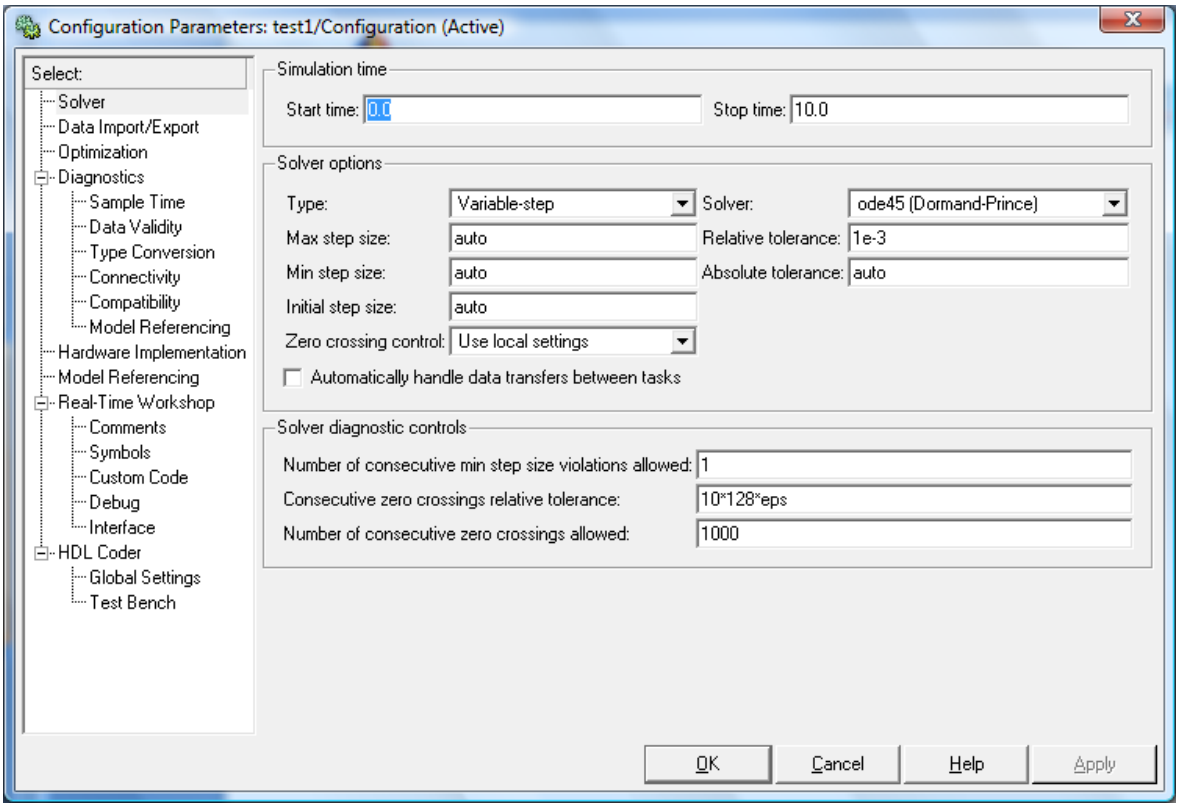


Hình 3-4 Các cửa sổ thiết lập tham số cho các khối chức năng trong mô hình

Ngoài ra, các thông tin về mô hình dạng văn bản cũng dễ dàng thêm vào bằng cách kích đúp trên cửa sổ mô hình. Tương tự tên các khối chức năng trong mô hình cũng có thể thay đổi theo ý muốn.



Hình 3-5 Cửa sổ thiết lập tham số trên khối `Scope`



Hình 3-6 Cửa sổ thiết lập tham số mô phỏng của mô hình

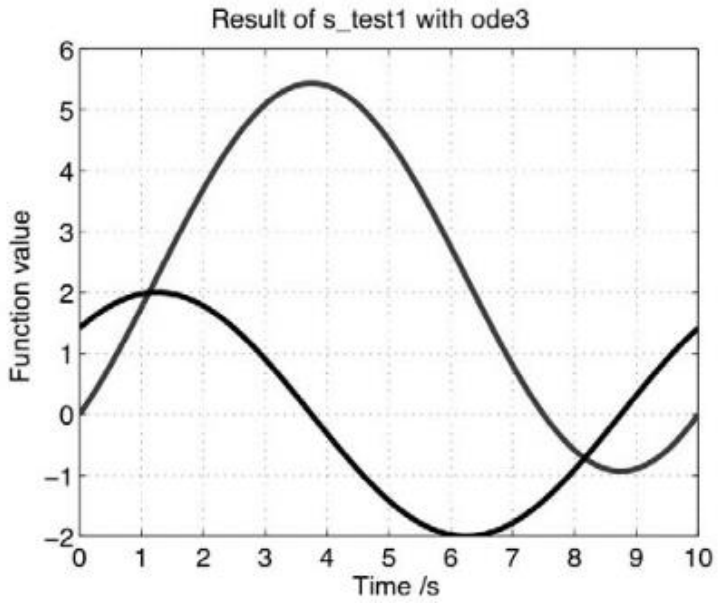
3.2.3 Mô phỏng bằng Simulink

Các tham số mô phỏng của mô hình được thiết lập qua lựa chọn menu `Simulation – Configuration Parameters` để mở cửa sổ như cho thấy trong

hình 3-6. Các thiết lập cơ bản cho mô hình là lựa chọn các tham số cho bộ giải phương trình vi phân và thời gian chạy mô phỏng. Ở chế độ mặc định thời gian mô phỏng được xác định từ 0 đến 10 giây. Ngoài ra mục `Data Import/Export` cho phép thiết lập và quản lý việc truy xuất dữ liệu kết quả trên cửa sổ quản lý biến workspace.

Mô phỏng được thực hiện bằng lựa chọn lệnh menu `Simulation – start` hoặc kích trực tiếp biểu tượng hình tam giác trên thanh công cụ. Khi chạy mô phỏng kết quả sẽ được hiển thị trên khối `Scope`. Do các kết quả được lưu vào biến `S_test1_signals` như thiết lập tham số cho khối `Scope` nên có thể sử dụng để vẽ đồ thị kết quả trong MATLAB như đoạn lệnh cho dưới đây. Kết quả được cho trong hình 3-7.

```
>> plot(S_test1_signals(:,1), ...
        [S_test1_signals(:,2),S_test1_signals(:,3)])
>> title('Result of s_test1 with ode3')
>> xlabel('Time /s')
>> ylabel('Function value')
>> grid
```

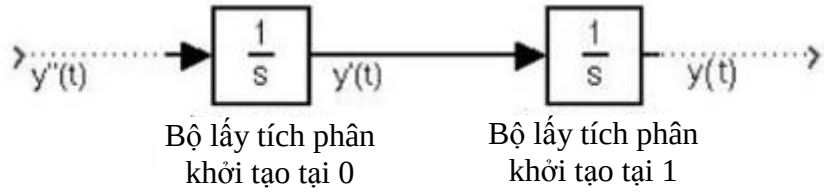


Hình 3-7 Kết quả được vẽ trong MATLAB

3.3 Giải phương trình vi phân bằng Simulink

Các phương trình vi phân có thể được giải trong MATLAB như đã trình bày trong Chương 2. Tuy nhiên đối với các phương trình vi phân phức tạp việc sử dụng Simulink để giải trở nên đơn giản và dễ dàng hơn. Nguyên tắc chung của giải

phương trình vi phân bằng Simulink là cần chuyển đổi phương trình vi phân thành một hệ thống động và được mô tả bằng sơ đồ khối.



Hình 3-8 Kỹ thuật tích phân để tìm $y(t)$

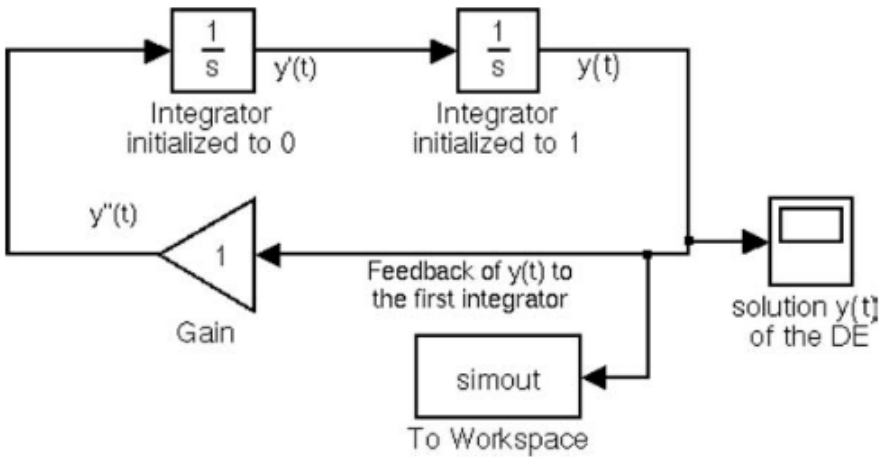
Để hiểu rõ cách thực hiện giải phương trình vi phân bằng Simulink ta hãy xét một ví dụ đơn giản là một bài toán phương trình vi phân bậc hai như sau

$$y''(t) = -y(t), \text{ với } \quad y(0) = 1, y'(0) = 0, \tag{3.1}$$

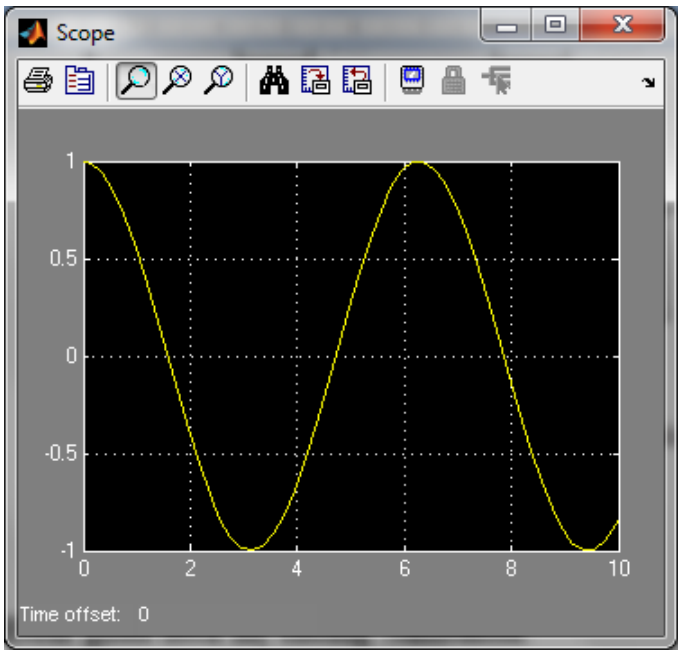
Nghiệm của phương trình vi phân này có thể được giải dễ dàng và ta có

$$y(t) = \cos(t) \tag{3.2}$$

Để giải phương trình (3.1) bằng Simulink ta có thể sử dụng kỹ thuật lấy tích phân các đạo hàm của nghiệm mong muốn bằng bộ tích phân trong Simulink như cho thấy trong hình 3-8. Các bộ lấy tích phân sẽ phải được khởi tạo tương ứng với các giá trị ban đầu mà đầu bài cho, cụ thể là bộ tích phân đầu tiên được đặt là 0 vì giá trị tín hiệu đầu ra $y'(t)$ tại thời điểm 0 (bắt đầu của mô phỏng) phải là 0 theo điều kiện ban đầu. Bộ tích phân thứ hai được đặt là 1 vì giá trị tín hiệu đầu ra $y(t)$ cần là 1 tại thời điểm 0 theo điều kiện ban đầu. Tuy nhiên để xây dựng mô hình thực hiện theo đúng phương trình vi phân (3.1) ta cần kết nối đầu ra bộ tích phân thứ hai với đầu vào bộ tích phân đầu tiên có đổi sang dấu âm. Việc chuyển dấu âm của đầu ra $y(t)$ được thực hiện bằng việc sử dụng khối khuyếch đại Gain trong thư viện Math Operations với hệ số khuyếch đại bằng -1. Hoạt động này tương đương với việc tín hiệu đầu vào được nhân với hệ số -1 để chuyển đổi dấu. Để hiển thị kết quả khối Scope sẽ được sử dụng với đầu vào được kết nối với đầu ra bộ tích phân thứ hai. Mô hình Simulink hoàn chỉnh để giải phương trình (3.1) được cho thấy trong hình 3-9. Sau khi chạy mô hình Simulink kết quả hiển thị trên đồ thị khối Scope là đường cosine giống như nghiệm chính xác của phương trình (Hình 3-10).



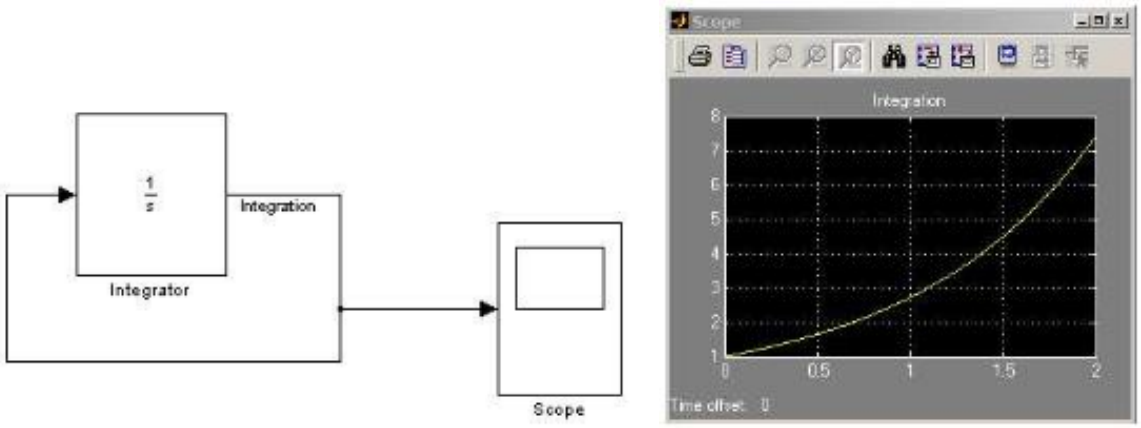
Hình 3-9 Mô hình Simulink để giải phương trình vi phân (3.1)



Hình 3-10 Kết quả giải phương trình bằng Simulink trên khối Scope

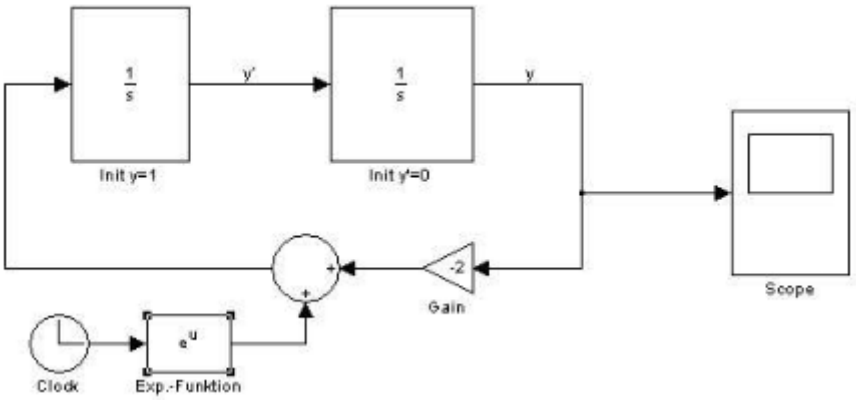
Để hiểu rõ cách thực hiện giải phương trình vi phân bằng Simulink, một số ví dụ các phương trình vi phân khác nhau được cho dưới đây.

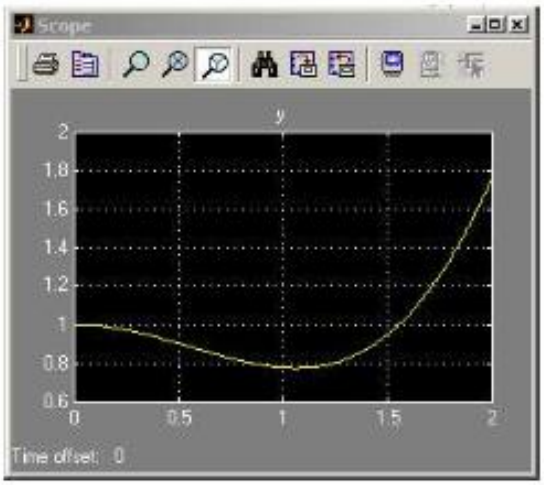
Ví dụ 3.1: Giải phương trình vi phân $y'(t) = y(t)$ với điều kiện ban đầu $y(0) = 1$. Mô hình Simulink và kết quả giải phương trình được cho trong hình 3-11.



Hình 3-11 (a) Mô hình Simulink và (b) kết quả giải phương trình vi phân trong ví dụ 3.1

Ví dụ 3.2: Giải phương trình vi phân $y''(t) = e^t - 2y(t)$ với điều kiện ban đầu $y(0) = 1$ và $y'(0) = 0$. Mô hình Simulink và kết quả giải phương trình được cho trong hình 3-12.





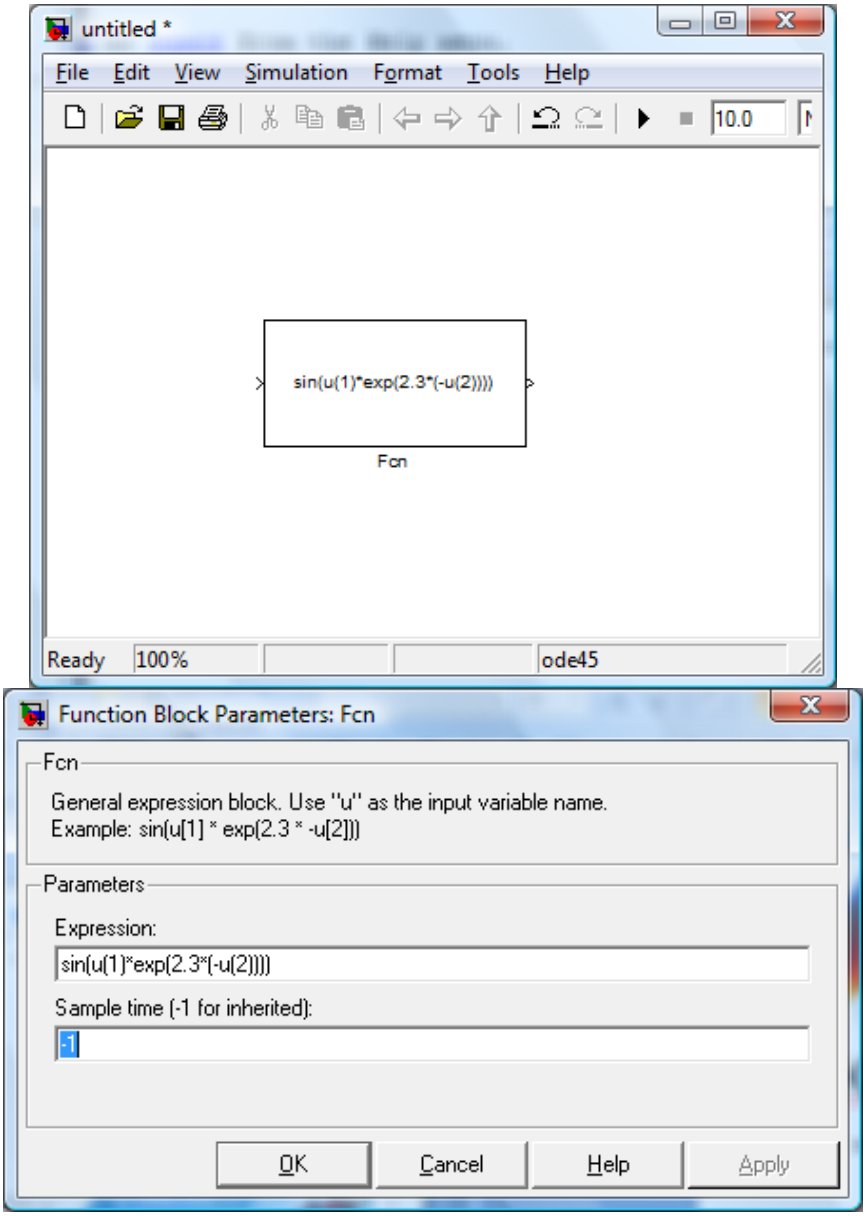
Hình 3-12 (a) Mô hình Simulink và (b) kết quả giải phương trình vi phân trong ví dụ 3.2

3.4 Đơn giản hóa hệ thống Simulink

Trong Simulink có một số cách giúp đơn giản hóa hệ thống Simulink đặc biệt đối với những mô hình có độ phức tạp cao. Thêm nữa các cách này cũng giúp việc quản lý và phân cấp sắp xếp hệ thống một cách tốt hơn.

3.4.1 Khối Fcn

Trong mô hình hệ thống Simulink các hoạt động tính toán đại số có thể được thực hiện dễ dàng bằng việc sử dụng



Hình 3-13 Khối Fcn và giao diện thiết lập tham số cho khối

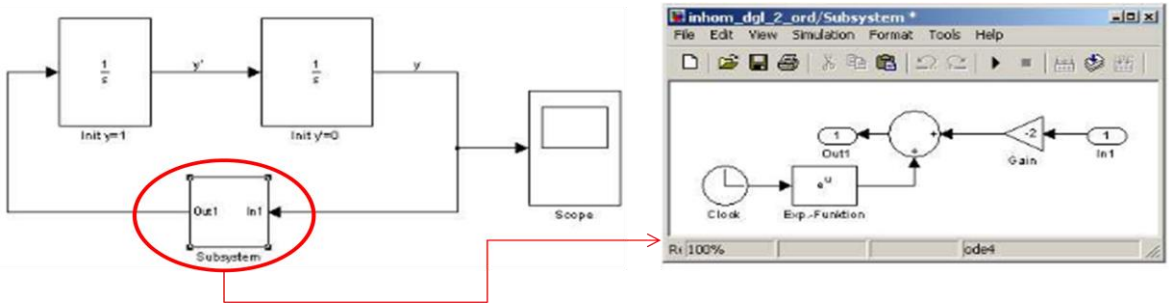
Khi sử dụng khối `Fcn` cần chú ý rằng tín hiệu đầu vào của khối luôn được gọi là u mà không cần quan tâm nó có thể được ký hiệu như thế nào trong hệ thống.

Tín hiệu đầu vào có thể là một đại lượng vô hướng hoặc vector. Nếu nhiều tín hiệu phải được đưa vào một khối thì chúng có thể được kết hợp lại trước thành một tín hiệu vector bằng việc sử dụng khối `Mux`. Với các tín hiệu đầu vào vector, các thành phần được xác định bằng việc đánh chỉ số *u*(1), *u*(2), … trong khối. Tín hiệu đầu ra luôn là một đại lượng vô hướng.

3.4.2 Xây dựng các phân hệ (Subsystem)

Trong các mô hình cỡ lớn việc sử dụng các khối `Fcn` sẽ gặp hạn chế trong việc đơn giản hóa mô hình. Một cách giúp đơn giản hóa mô hình Simulink một cách hiệu quả là xây dựng các phân hệ để kết hợp một số khối chức năng với nhau vào thành một khối.

Để nhóm một số khối chức năng vào thành một phân hệ, các khối chức năng và cả đường kết nối giữa các khối đó cần được lựa chọn. Sau khi lựa chọn, chọn menu `Edit – Create subsystem` để tạo ra phân hệ. Bước này cũng có thể thực hiện bằng cách bấm chuột phải và lựa chọn `Create subsystem`. Lấy mô hình ví dụ cho ở hình 3-12, các khối tính toán phép cộng, nhân (`Gain`) và hàm e mũ sẽ được lựa chọn để tạo ra một `subsystem` như cho thấy ở hình 3-14. Khi kích đúp vào khối `subsystem`, một cửa sổ mô hình tách biệt sẽ được mở cho thấy mô hình bên trong khối `subsystem`.



Hình 3-14 Xây dựng phân hệ Subsystem để đơn giản hóa mô hình Simulink

Một cách khác để tạo ra khối Subsystem bằng cách lấy khối `Subsystem` từ thư viện `Ports&Subsystems`. Kích đúp vào khối để mở cửa sổ mô hình và có thể xây dựng mô hình cho khối subsystem trong cửa sổ này. Khi khối subsystem được tạo ra, các khối cổng đầu vào `In1` và đầu ra `Out1` cũng được thêm vào mô hình. Các khối này cũng có thể lấy từ thư viện `Sources` và `Sinks` tương ứng. Số lượng công đầu vào và đầu ra tùy thuộc vào mô hình phân hệ xây dựng.

Việc xây dựng các subsystem cho phép tổ chức hệ thống Simulink ở dạng phân cấp cho mô hình mô phỏng như đề cập trong chương 1 một cách dễ dàng. Việc

duyệt từ các subsystem mức dưới lên mô hình hệ thống chính có thể thực hiện qua nút biểu tượng hình mũi tên trên thanh công cụ. Trong một số trường hợp các subsystem cũng cần tạo giao diện để thiết lập tham số giống như các khối chức năng cơ bản khác. Để tạo giao diện thiết lập tham số cho khối subsystem, lựa chọn menu `Edit – Mask subsystem`. Khi các khối subsystem được xây dựng, các khối này có thể được tập hợp lại để thành một thư viện cho mục đích riêng của người sử dụng và được sử dụng giống như các thư viện khác trong Simulink.

3.5 Tương tác với MATLAB

3.5.1 Truyền các biến giữa Simulink và MATLAB

Việc truyền các biến giữa Simulink và MATLAB có ý nghĩa trong nhiều trường hợp để kết nối các chương trình và chức năng xử lý giữa MATLAB và Simulink. Có nhiều cách để thực hiện truyền các biến trong Simulink với MATLAB:

- Sử dụng khối `To Workspace` trong thư viện `Sinks`, một tên biến có thể được thiết lập để tạo ra trong MATLAB sau khi chạy mô phỏng. Các dữ liệu được ghi trong biến tùy thuộc vào kết nối đầu vào của khối trong mô hình Simulink.
- Trong thiết lập tham số cấu hình Simulink: vào các tên biến phù hợp trong khối tham số ở mục `Data Imporrt/Export – Save to Workspace`.
- Sự dụng phần thiết lập tham số trong khối `Scope/Data History` như đã đề cập trong phần 3.2.

Ngoài ra các biến sử dụng trong chương trình Simulink có thể được định nghĩa trong MATLAB bằng việc viết một chương trình m-file. Khi các tên biến và giá trị của biến được định nghĩa trước thì các biến này có thể được sử dụng thay cho các giá trị cụ thể trong phần thiết lập tham số cho các khối trong mô hình. Như vậy việc thay đổi giá trị cho các biến có thể thực hiện qua chương trình MATLAB thay cho việc thiết lập ở từng khối của Simulink.

3.5.2 Lặp lại các mô phỏng Simulink trong MATLAB

Trong nhiều trường hợp một mô phỏng Simulink phụ thuộc một số lượng lớn các tham số. Do vậy nó sẽ rất hữu ích khi biến đổi một hoặc nhiều tham số để thiết

lập sự phụ thuộc hệ thống được mô phỏng lên các tham số này. Việc khảo sát này có thể được thực hiện tự động qua MATLAB.

Gọi chương trinh Simulink qua MATLAB: việc gọi này có thể được thực hiện ở ngoài cửa sổ lệnh hoặc trong một chương trình MATLAB qua sử dụng hàm sim. Thêm nữa cơ chế này được sử dụng để gọi một chương trình mô phỏng Simulink từ MATLAB mà không cần phải khởi tạo Simulink cho phép tốc độ thực hiện nhanh hơn. Cách sử dụng cơ bản hàm sim được cho thấy qua cú pháp sau:

```
>> [t,x,y] = sim('system',[starttime, endtime]);
```

Tham số `'system'` là tên của hệ thống Simulink để được thực hiện, vector tham số `[starttime, endtime]` xác định thời điểm bắt đầu và kết thúc của mô phỏng. Vector trả về `[t,x,y]` xuất hiện theo cùng trật tự trong cửa sổ tham số Simulink tại mục `Data Import/Export – Save to workspace.`

Thiết lập các tham số Simulink bằng hàm `set_param` cho phép định nghĩa các tham số của hệ thống ở bên ngoài chương trình. Cú pháp cơ bản của hàm

```
>> set_param('obj','parameter1',value1,'parameter2',…);
```

trong đó `'obj'` là tên một hệ thống hoặc tên khối có đường dẫn. Một số ví dụ sử dụng hàm `set_param` được cho dưới đây:

`>> set_param('vdp','Solver','ode15s','StopTime','3000');` thiết lập các tham số bộ giải và thời gian dừng mô phỏng của hệ thống vdp.

`>> set_param('vdp/Mu','Gain','1000');` thiết lập tham số hệ số khuyếch đại của khối Mu trong hệ thống vdp là 1000.

Ngược lại giá trị các tham số của các khối và của hệ thống có thể được lấy qua sử dụng hàm `get_param` qua cú pháp dưới đây:

```
>> pars = get_param('obj','parameter');
```

`>> cgain = get_param('vdp/Gain','Gain');` lấy giá trị tham số khuyếch đại hiện tại của khối Gain trong hệ thống vdp và gán vào biến cgain.

Quá trình lặp lại mô phỏng Simulink trong MATLAB được thực hiện bằng cách gọi chương trình Simulink trong vòng lặp `for`. Tại mỗi vòng lặp các tham số có thể được thiết lập lại qua sử dụng hàm `set_param`. Ví dụ về lặp lại mô phỏng Simulink cho trong đoạn mã MATLAB dưới đây.

```
% Ví dụ về quá trình lặp lại mô phỏng Simulink
for i=1:iterations
    % Set the block parameters with set_param
    % for each new iteration.
    reciprocalmass = num2str(1/M(i));
    set_param('s_denon3/invm','Gain',reciprocalmass);
    b = num2str(B(i));
    set_param('s_denon3/Factorb','Gain',b);
    c = num2str(Fc);
    set_param('s_denon3/Factorc','Gain',c);
    [t,x,y] = sim('s_denon3', [0,tm]);
    Y = [Y,y];
end;
```

3.5.3 Truyền các biến thông qua các biến toàn cục

Một khả năng nữa để liên lạc giữa các hàm MATLAB và Simulink đó là khai báo các tham số hay biến toàn cục. Cách khai báo đơn giản là:

```
global x y z p1 a2 ...
```

Nếu khai báo này được thêm vào trong hàm MATLAB thì các biến này cũng có thể được sử dụng dựa trên truyền các tham số cho Simulink để được sử dụng trong hệ thống mô phỏng.

Khả năng này thực sự rất thuận tiện nhưng nó che giấu một số mối nguy. Về cơ bản để tránh các ảnh hưởng không mong muốn trong tất cả các ngôn ngữ lập trình bậc cao bao gồm cả MATLAB, các biến toàn cục chỉ nên được sử dụng trong các trường hợp cấp thiết. Dưới các điều kiện thông thường việc sử dụng chúng nên tránh.

3.6 Tổng kết chương

Chương này đã cung cấp những đặc điểm cơ bản của bộ công cụ Simulink sử dụng để mô phỏng các hệ thống động. Thông qua các ví dụ để giải phương trình vi phân cách xây dựng mô hình mô phỏng trong Simulink được giới thiệu. Cách đơn giản hóa mô hình và cách thức tổ chức mô hình dạng phân cấp cũng được trình bày. Kết nối giữa MATLAB và Simulink cũng là đặc điểm quan trọng để kết hợp cả khả năng tính toán trong MATLAB với xây dựng mô phỏng dựa trên mô hình trong Simulink. Các kiến thức cơ bản này giúp người học tiếp tục phát triển kỹ năng sử dụng các thư viện trong Simulink để mô phỏng cho các hệ thống cụ thể như hệ thống truyền thông.

Câu hỏi/bài tập chương 3

1/ Thiết kế một hệ thống *Simulink* để giải bài toán giá trị ban đầu:

$$\ddot{y}(t) + y(t) = 0\,, \quad y(0) = 1, \dot{y}(0) = 0\,.$$

Chạy và so sánh kết quả với nghiệm chính xác.

Biến đổi hệ thống để mô phỏng một hàm nhiễu loạn (vế bên phải ≠ 0) được xác định là e^{-t}

2/ Giải bài toán giá trị ban đầu:

$$t\ddot{y}(t) + 2\dot{y}(t) + 4y(t) = 4\,, \quad y(1) = 1,\ \dot{y}(1) = 1$$

bằng một hệ thống *Simulink* phù hợp.

3/ Giải hệ phương trình vi phân sau:

$$\begin{aligned} \dot{y}_1(t) &= -3y_1(t) - 2y_2(t)\,, \quad y_1(0) = 1\,, \\ \dot{y}_2(t) &= \quad 4y_1(t) + 2y_2(t)\,, \quad y_2(0) = 1 \end{aligned}$$

bằng một hệ thống *Simulink* phù hợp.

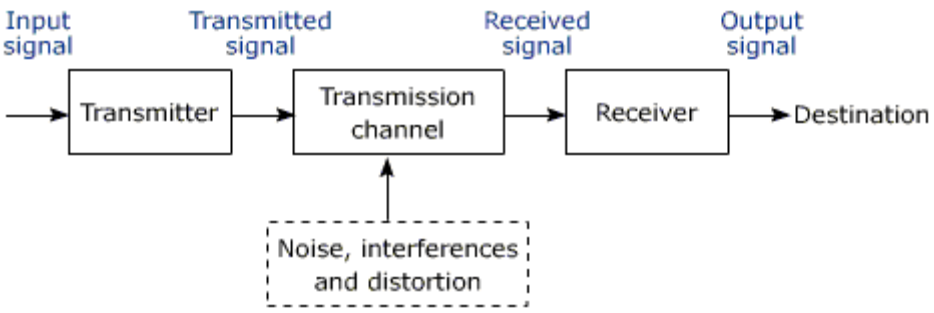
Chương 4 Mô phỏng tín hiệu và quá trình thu phát

Nội dung trong chương này sẽ tập trung vào mô phỏng các quá trình xảy ra tại hai khối cơ bản trong hệ thống truyền thông là khối phát và khối thu tín hiệu.

4.1 Giới thiệu

Một hệ thống thông tin bao gồm 3 thành phần cơ bản như mô tả trong hình 4-1 đó là:

- Khối phát tín hiệu: thực hiện xử lý tín hiệu đầu vào để tạo ra một tín hiệu truyền dẫn phù hợp với các đặc tính của kênh truyền. Một số chức năng xử lý quan trọng bao gồm quá trình mã hóa và quá trình điều chế.
- Kênh thông tin: là môi trường để truyền dẫn thông tin từ nguồn đến đích. Có nhiều loại kênh truyền với các đặc tính kênh khác nhau. Các kênh truyền gây ra suy hao và méo dạng tín hiệu làm tín hiệu bị suy giảm khi tới đầu thu.
- Khối thu tín hiệu: Tiếp nhận tín hiệu từ kênh truyền và khôi phục lại tín hiệu được phát đi. Các chức năng xử lý ở phía thu thường thực hiện ngược lại phía phát bao gồm các quá trình giải điều chế, giải mã và lọc tín hiệu.



Hình 4-1 Mô hình tổng quát hệ thống thông tin.

Mô phỏng hệ thống truyền thông cơ bản cũng được phân chia thành các quá trình mô phỏng như mô hình hệ thống trong hình 4-1. Tùy thuộc vào mục đích mô phỏng

và mức độ phức tạp của mô hình mô phỏng mà các khối cơ bản được chi tiết hóa theo các mức phân cấp như đã đề cập trong chương 1. Sơ đồ khối hệ thống truyền thông trong hình 1-5 cho thấy các quá trình cơ bản xảy ra trong các bộ thu phát tín hiệu và những vấn đề cơ bản để mô phỏng các quá trình này sẽ đề cập chi tiết trong chương này.

4.1.1 Mô hình mô phỏng tín hiệu băng gốc và thông dải

Tín hiệu trong hệ thống thông tin bao gồm hai loại cơ bản: tín hiệu băng gốc và tín hiệu thông dải. Tín hiệu băng gốc là tín hiệu có phổ tần tập trung quanh tần số $f = 0$, còn tín hiệu thông dải là tín hiệu có phổ tần tập trung quanh một tần số sóng mang f_c. Trong các hệ thống thông tin thực tế, các tín hiệu ban đầu từ nguồn tin được xem là các tín hiệu băng gốc thường được điều chế bởi một sóng mang phù hợp với kênh truyền, lúc này tín hiệu băng gốc bị dịch lên băng tần cao hơn bởi một lượng tương ứng với tần số sóng mang f_c và được xem là tín hiệu thông dải. Quá trình chuyển đổi từ tín hiệu băng gốc lên tín hiệu thông dải còn được gọi là chuyển đổi lên (up-conversion) được thực hiện qua quá trình điều chế sóng mang. Ngược lại quá trình chuyển đổi xuống (down-conversion) chuyển tín hiệu thông dải trở về băng gốc được thực hiện qua quá trình giải điều chế.

Tín hiệu thông dải hay điều chế sóng mang $x(t)$ có thể viết thành:

$$x(t) = r(t)\cos\left[2\pi f_c t + \phi(t)\right] = \mathrm{Re}\left[r(t)e^{j(2\pi f_c t + \phi(t))}\right] = \mathrm{Re}\left[r(t)e^{j\phi(t)}e^{j(2\pi f_c t)}\right] \quad (4.1)$$

trong đó $r(t)$ là điều chế biên độ, $\phi(t)$ là điều chế pha của tín hiệu và f_c là tần số sóng mang. Tín hiệu

$$\tilde{x}(t) = r(t)e^{j\phi(t)} \quad (4.2)$$

chứa cùng thông tin như $x(t)$ và được gọi là tín hiệu tương đương băng gốc hoặc đường bao phức của tín hiệu thông dải $x(t)$. Như vậy $\tilde{x}(t)$ là dạng tương đương băng gốc hay thông thấp của $x(t)$ có cùng độ rộng băng tần B nhưng được biểu diễn ở dạng phức.

Trong điều chế tín hiệu số, các tín hiệu điều chế biên độ và điều chế pha gồm có cả hai thành phần đồng pha và vuông pha luôn có thể được mô tả theo dạng:

$$x(t) = p(t)\cos\left(2\pi f_c t\right) - q(t)\sin\left(2\pi f_c t\right) \quad (4.3)$$

và có phổ tương ứng là khai triển Fourier của $x(t)$ như sau:

$$X(f)=\frac{1}{2}\left[P(f-f_c)+jQ(f-f_c)+P(f+f_c)-jQ(f+f_c)\right] \qquad (4.4)$$

trong đó $p(t)$ và $q(t)$ là các thành phần đồng pha và vuông pha của tín hiệu có độ rộng băng tần B, $P(f)$ và $Q(f)$ là phổ của $p(t)$ và $q(t)$ tương ứng. Tương tự như (4.1), tín hiệu điều chế có thể viết thành:

$$x(t)=\mathrm{Re}\left[\tilde{x}(t)e^{j2\pi f_c t}\right] \qquad (4.5)$$

trong đó $\tilde{x}(t)$ là tín hiệu tương đương băng gốc được xác định như sau:

$$\tilde{x}(t)=p(t)+jq(t) \qquad (4.6)$$

và có phổ tương ứng:

$$\tilde{X}(f)=P(f)+jQ(f) \qquad (4.7)$$

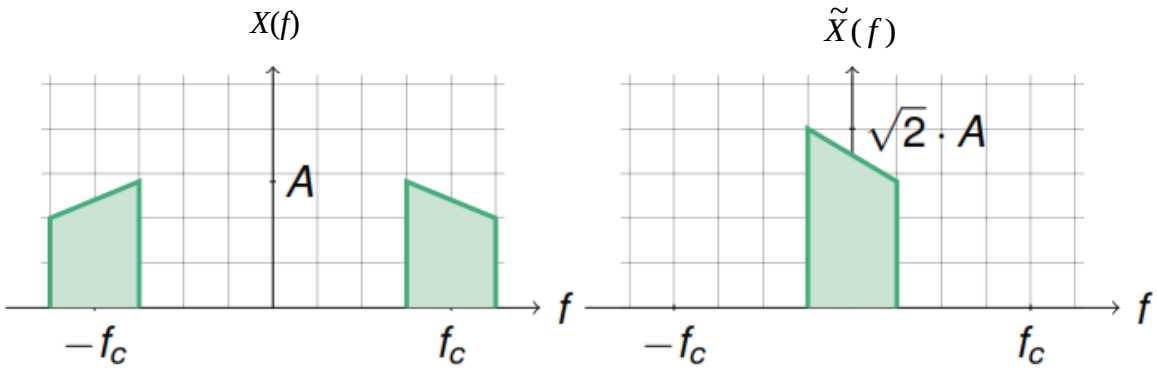
Như vậy trong mô phỏng tín hiệu $x(t)$ có thể được mô tả tốt bởi đường bao phức $\tilde{x}(t)$ mà không làm mất đi tín hiệu gốc ban đầu. Tín hiệu đường bao phức có thể được biểu diễn theo một cách khác như sau:

$$\tilde{x}(t)=a(t)e^{j\phi(t)} \qquad (4.8)$$

trong đó $a(t)=\sqrt{p^2(t)+q^2(t)}$ là điều chế biên độ và $\phi(t)=\tan^{-1}\left[q(t)/p(t)\right]$ là điều chế pha của tín hiệu. Phổ của tín hiệu tương đương băng gốc quan hệ với phổ của tín hiệu thông dải như sau:

$$\tilde{X}(f)=\begin{cases}2X(f+f_c) & \text{khi } f+f_c>0\\ 0 & \text{còn lại}\end{cases} \qquad (4.9)$$

Như vậy, phổ của tín hiệu tương đương băng gốc có thể coi như phần phổ dương của tín hiệu thông dải và được dịch xuống băng gốc một lượng bằng f_c như được mô tả trong hình 4-2

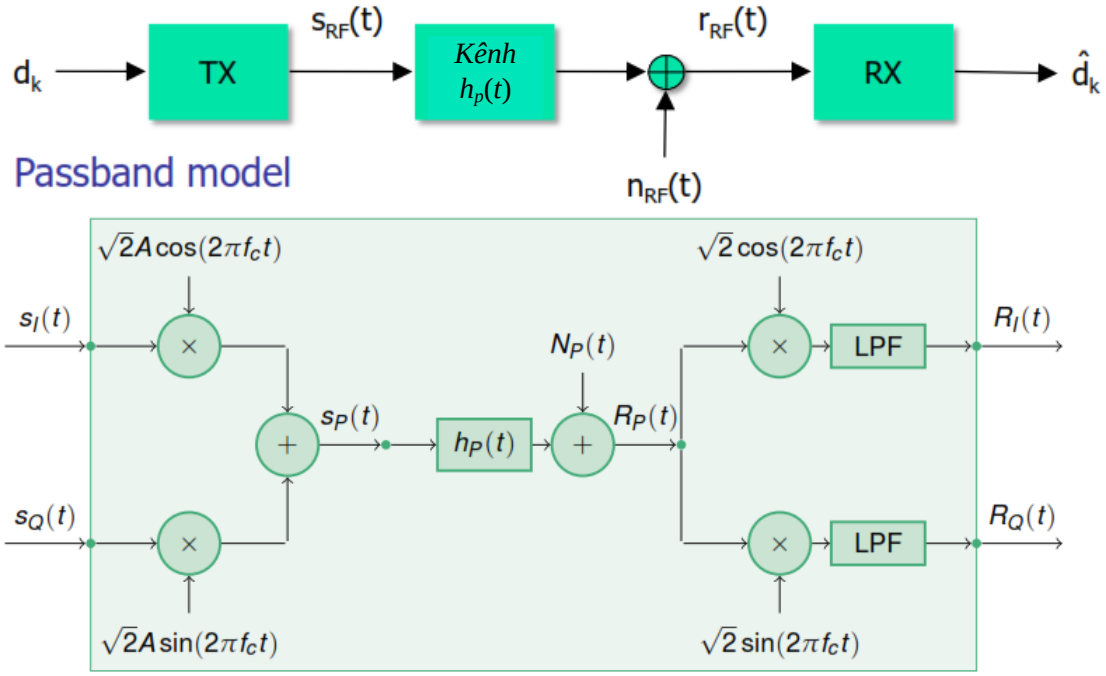


Hình 4-2 Phổ của tín hiệu thông dải và phổ của tín hiệu tương đương băng gốc

Tương ứng với các loại tín hiệu được sử dụng trong mô phỏng, có hai mô hình mô phỏng tương ứng đó là:

- Mô hình thông dải và
- Mô hình tương đương băng gốc

Trong mô phỏng hệ thống truyền thông sử dụng mô hình thông dải như được mô tả tổng quát trong hình 4-3 trong đó các tín hiệu bản tin gốc ban đầu sẽ được điều chế bởi sóng mang f_c để tạo ra tín hiệu thông dải s_p tại đầu ra bộ phát. Vì tín hiệu thông dải là tín hiệu thực nên các hàm mô tả các thành phần khác trong hệ thống như hàm đáp ứng kênh $h_p(t)$, nhiễu cộng $n_p(t)$ và hàm lọc đều có giá trị thực tương ứng. Tín hiệu thu được $r_p(t)$ tại bộ thu sẽ được giải điều chế để chuyển đổi tín hiệu thông dải về tín hiệu băng gốc khôi phục tín hiệu ban đầu. Tuy mô hình thông dải mô tả hoạt động sát với hệ thống thực tế nhưng mô phỏng sử dụng mô hình này sẽ đòi hỏi khối lượng tính toán lớn.



Hình 4-3 Mô hình thông dải dạng rút gọn và mở rộng với quá trình điều chế và giải điều chế.

Trong mô phỏng hệ thống truyền thông, các tín hiệu hay hàm thời gian đều cần được lấy mẫu. Để đảm bảo điều kiện lấy mẫu, phổ của tín hiệu thông dải nằm trong dải tần $f_c - B/2 < f < f_c + B/2$ sẽ cho thấy rằng tần số lấy mẫu đòi hỏi cỡ bậc $2f_c + B$. Ở đây B được xem là độ rộng băng tần của tín hiệu băng gốc ban đầu. Do vậy mô hình tương đương băng gốc được sử dụng trong hầu hết các mô phỏng hệ thống truyền thông hiện nay. Ở mô hình tương đương băng gốc như mô tả trong hình 4-4,

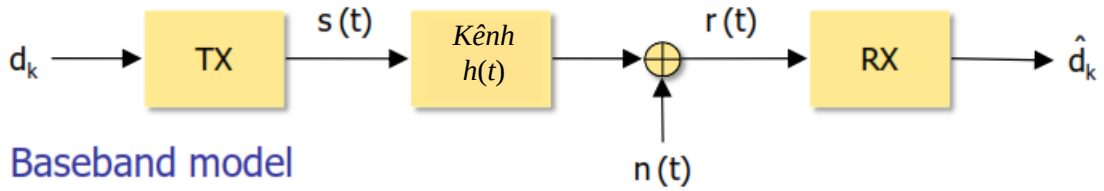
các tín hiệu là tín hiệu tương đương băng gốc chỉ xét đương bao phức của sóng mang chứ không xem xét đầy đủ cả sóng mang trong tín hiệu. Vì vậy các hàm trong mô hình tương đương băng gốc bao gồm đáp ứng xung của kênh $h(t)$, nhiễu cộng $n(t)$ và các hàm lọc đều được biểu diễn ở dạng phức. Cũng tương tự như các tín hiệu mô tả ở trên, các hàm phức quan hệ với các hàm trong mô hình thông dải như sau:

$$h_p(t) = 2\,\mathrm{Re}\left[h(t)e^{j2\pi f_c t}\right] \tag{4.10a}$$

$$n_p(t) = \mathrm{Re}\left[n(t)e^{j2\pi f_c t}\right] \tag{4.10b}$$

$$r_p(t) = \mathrm{Re}\left[r(t)e^{j2\pi f_c t}\right] \tag{4.10c}$$

Đối với mô hình tương đương băng gốc, tần số lấy mẫu đòi hỏi trong mô phỏng hệ thống chỉ cỡ bậc độ rộng băng tần B của tín hiệu, do vậy mô hình này gọn nhẹ và thuận tiện hơn khi thực hiện mô phỏng.



Hình 4-4 Mô hình tương đương băng gốc

Cũng giống như mô tả tín hiệu, mô hình hóa hay mô phỏng hệ thống truyền thông có thể được thực hiện trong miền thời gian hoặc miền tần số. Trong miền thời gian, tín hiệu tại bộ thu được xác định:

$$r(t) = s(t) * h(t) + n(t) \tag{4.11}$$

trong đó * là phép tích chập trong miền thời gian. Còn trong miền tần số, tín hiệu thu được xác định:

$$R(f) = S(f) \cdot H(f) + N(f) \tag{4.12}$$

trong đó $R(f)$, $S(f)$, $H(f)$ và $N(f)$ là các khai triển Fourier của $r(t)$, $s(t)$, $h(t)$ và $n(t)$ tương ứng.

4.1.2 Quá trình lấy mẫu và nội suy

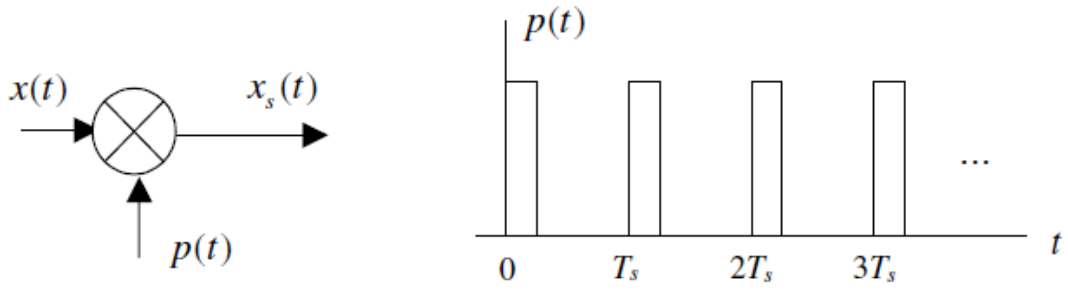
Trong mô phỏng hệ thống truyền thông trên hệ thống máy tính số, các dạng tín hiệu và các mô hình thời gian liên tục được đòi hỏi chuyển đổi thành dạng tín hiệu hay mô hình rời rạc về thời gian. Quá trình chuyển đổi này được thực hiện qua quá trình lấy mẫu tín hiệu và được biểu diễn bởi

$$x_s(t) = x(t)\,p(t) \tag{4.13}$$

với

$$p(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT_s) \tag{4.14}$$

trong đó chuỗi xung kim $p(t)$ được gọi là hàm lấy mẫu, $x(t)$ là tín hiệu liên tục về thời gian và $x_s(t) = x(nT_s) = x[n]$ là tín hiệu rời rạc về thời gian với T_s là chu kỳ lấy mẫu và $f_s = T_s^{-1}$ là tốc độ lấy mẫu.



(a) *Hoạt động lấy mẫu* (b) *Hàm lấy mẫu*

Hình 4-5 Mô tả quá trình lấy mẫu

Hàm $x_s(t)$ cũng là một chuỗi xung kim có trọng số bằng với giá trị của các mẫu $x(t)$ hay

$$x_s(t) = \sum_{n=-\infty}^{\infty} x(nT_s)\delta(t - nT_s) \tag{4.15}$$

Khai triển Fourier của $x_s(t)$ ta được

$$X_s(f) = \sum_{n=-\infty}^{\infty} x(nT_s)e^{-j2\pi nfT_s} \tag{4.16}$$

và quan hệ với phổ của tín hiệu liên tục bởi

$$X_s(f) = \frac{1}{T_s}\sum_{n=-\infty}^{\infty} X(f + nf_s) \tag{4.17}$$

Như vậy $X_s(f)$ là một hàm tuần hoàn gồm tổng các bản sao $X(f)$ bị định cỡ và bị dịch trong miền tần số.

Để tránh lỗi chồng phổ (aliasing), quá trình lấy mẫu một tín hiệu băng gốc có độ rộng băng tần B sẽ cần phải thỏa mãn điều kiện lấy mẫu Nyquist tức $f_s > 2B$ hay

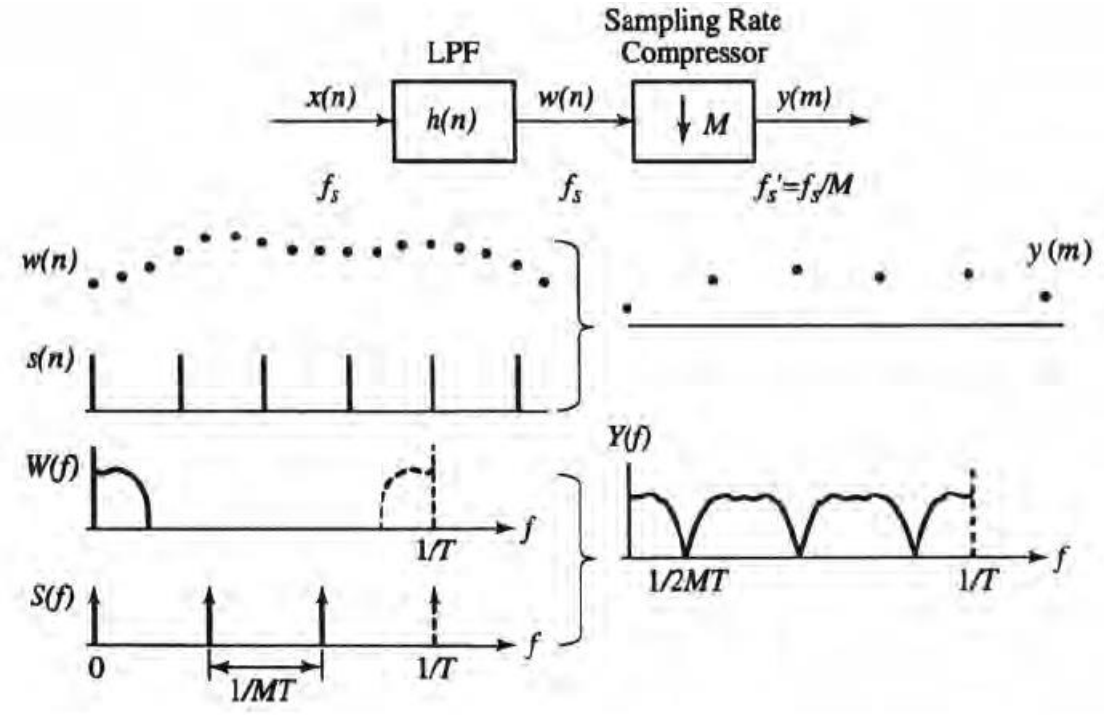
tần số lấy mẫu phải lớn hơn hai lần độ rộng băng tần của tín hiệu gốc. Nếu điều kiện lấy mẫu không thỏa mãn thì hiện tượng chồng phổ xảy ra và không khôi phục lại được tín hiệu gốc ban đầu. Trong mô phỏng tần số lấy mẫu cần được lựa chọn phù hợp để giảm sai số chồng phổ nhưng tránh tăng thời gian chạy mô phỏng.

Trong một số trường hợp mô phỏng hệ thống có các độ rộng băng tần khác nhau, quá trình chuyển đổi tốc độ mẫu sẽ được đòi hỏi tại một số giai đoạn mô phỏng. Sự chuyển đổi tốc độ mẫu đơn giản là tốc độ lấy mẫu dạng sóng sẽ được tăng lên hoặc giảm đi. Quá trình chuyển đổi lên một tốc độ lấy mẫu cao hơn gọi là quá trình tăng mẫu (upsampling) đòi hỏi sử dụng kỹ thuật nội suy (interpolation), còn quá trình chuyển đổi xuống một tốc độ thấp hơn được gọi là quá trình giảm mẫu (downsampling).

Quá trình giảm mẫu được thực hiện tại biên giữa phần tín hiệu băng rộng và băng hẹp và được mô tả trong hình 4-6. Tín hiệu giảm mẫu có chu kỳ lấy mẫu lớn hơn so với chu kỳ mẫu ban đầu và biểu diễn bởi

$$x(kT_s) \rightarrow x(kT_d) = x(kMT_s) \tag{4.18}$$

trong đó T_d là chu kỳ lấy mẫu sau giảm mẫu và lớn hơn M lần so với chu kỳ lấy mẫu ban đầu T_s, tương ứng với tần số lấy mẫu giảm M lần. Quá trình giảm mẫu có thể được thực hiện đơn giản bằng việc lựa chọn các mẫu với khoảng cách mẫu tăng M lần tương ứng từ tập mẫu tín hiệu ban đầu.

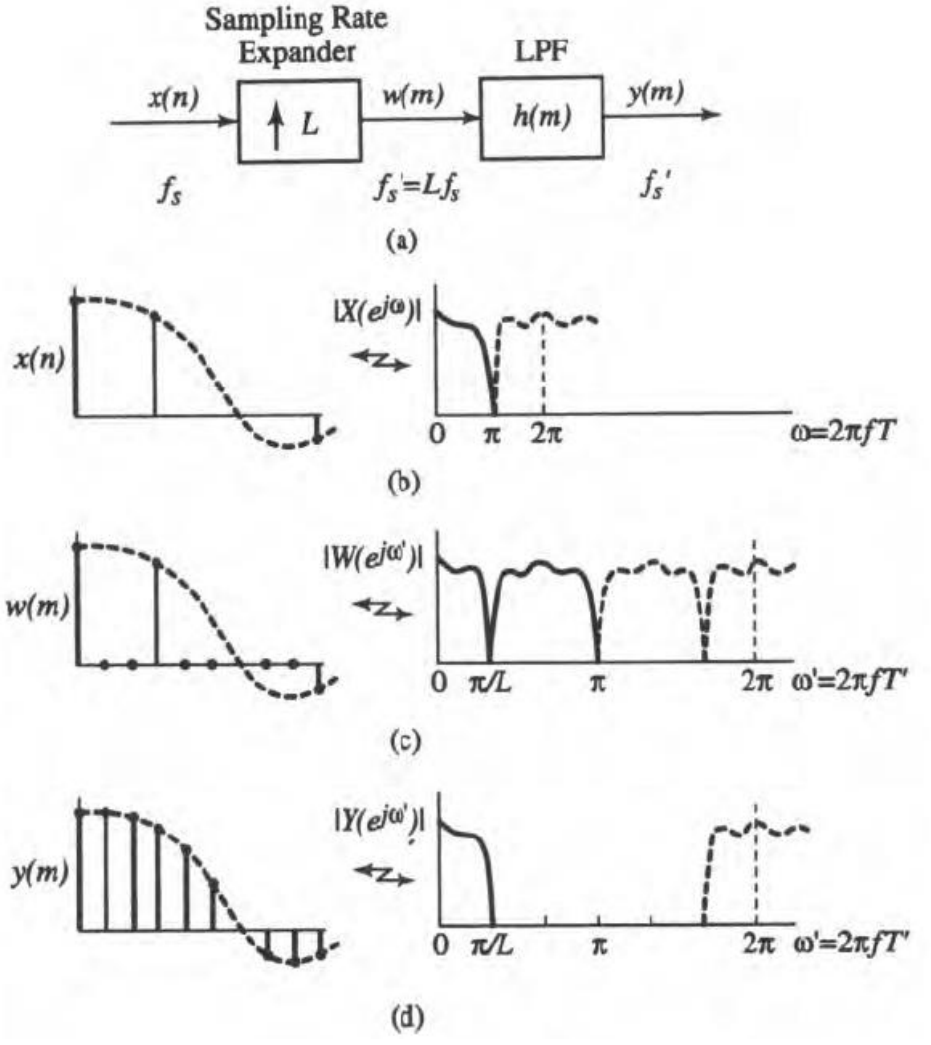


Hình 4-6 Ví dụ mô tả quá trình giảm mẫu

Quá trình tăng mẫu được thực hiện tại biên giữa phần tín hiệu băng hẹp và băng rộng và được mô tả trong hình 4-7. Tín hiệu tăng mẫu có chu kỳ lấy mẫu nhỏ hơn so với chu kỳ mẫu ban đầu và biểu diễn bởi

$$x(kT_s) \rightarrow x(kT_u) = x(kT_s/M) \tag{4.19}$$

trong đó T_u là chu kỳ lấy mẫu sau tăng mẫu và nhỏ hơn M lần so với chu kỳ lấy mẫu ban đầu T_s, tương ứng với tần số lấy mẫu tăng M lần. Tuy nhiên khác với quá trình giảm mẫu, quá trình tăng mẫu đòi hỏi việc xác định giá trị các mẫu mới tại các thời điểm nằm giữa các mẫu ban đầu. Việc tính toán các mẫu mới giữa hai mẫu đã biết được thực hiện bằng phương pháp nội suy.



Hình 4-7 Ví dụ mô tả quá trình tăng mẫu

Phép nội suy là một hoạt động quan trọng trong kỹ thuật đa tốc độ và cho một số mục đích trong mô phỏng. Có nhiều kỹ thuật nội suy bao gồm:

- Nội suy tuyến tính: đây là kỹ thuật nội suy đơn giản nhất khi ước tính các điểm mẫu mới nằm giữa hai mẫu đã biết bằng một hàm tuyến tính.
- Nội suy đa thức: đây là kỹ thuật nội suy bậc cao khi ước tính các điểm mẫu mới nằm giữa hai mẫu đã biết bằng một hàm đa thức bậc cao. Các hệ số của đa thức được ước tính từ tập mẫu ban đầu.
- Nội suy spline: phương pháp này được coi như một dạng biến đổi của nội suy đa thức bậc cao trong đó bao gồm một tập các hàm đa thức khác nhau được sử dụng để ước tính các điểm mới trên các khoảng mẫu khác nhau đảm bảo với sai số ước tính là nhỏ nhất trên khoảng đó.
- Nội suy hàm sinc: phép nội suy này được sử dụng cho tín hiệu mẫu bị giới hạn băng tần, đây cũng được coi là phép nội suy bộ lọc thông thấp trong đó các mẫu được nội suy qua phép tích chập với đáp ứng xung hàm sinc.

Để thực hiện phép nội suy trong MATLAB, nhiều hàm trong `Signal Processing Toolbox` có thể được sử dụng bao gồm:

- hàm `interp1` thực hiện nội suy 1 chiều với các phương pháp nội suy khác nhau có thể được thiết lập.

```
yi = interp1(x,Y,xi) % xác định yi theo xi từ cặp x,Y
```

- hàm `spline` thực hiện phép nội suy spline

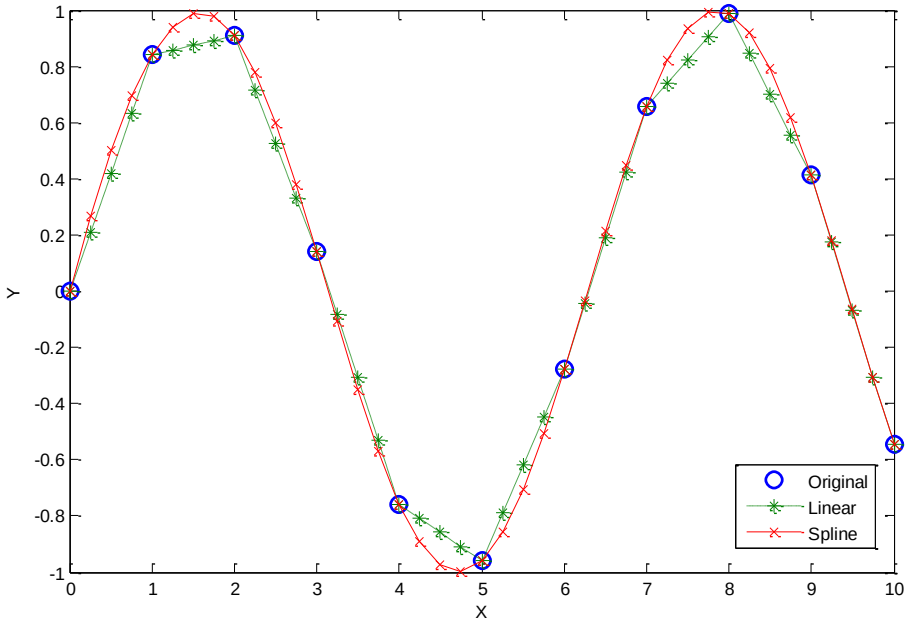
```
yy = spline(x,Y,xx) % xác định yy theo xx từ cặp x,Y
```

- hàm `interp` thực hiện nội suy để tăng tốc độ mẫu lên một số nguyên lần bằng phương pháp nội suy bộ lọc thông thấp.

```
y = interp(x,r) % tăng tốc độ mẫu của x lên r lần
```

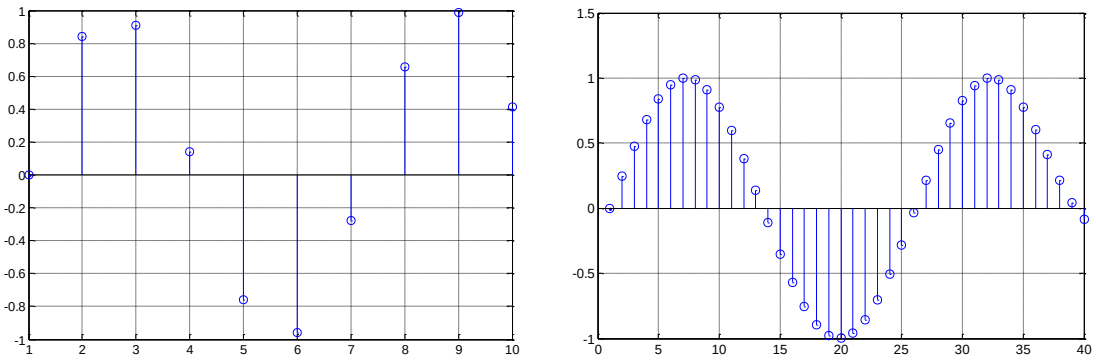
Một ví dụ về sử dụng các hàm nội suy này được cho trong đoạn mã MATLAB dưới đây:

```
% Ví dụ về sử dụng hàm interp1 trong phép tính nội suy
x = 0:10;
y = sin(x);
xi = 0:.25:10; % Các điểm lấy mẫu mới
ylin = interp1(x,y,xi,'linear'); % Nội suy tuyến tính
yspl = interp1(x,y,xi,'spline'); % Nội suy spline
plot(x,y,'o',xi,ylin,'-*', xi,yspl,'-x');
```



Hình 4-8 Ví dụ một số phép nội suy

```
% Ví dụ tăng mẫu sử dụng hàm interp trong MATLAB
x = 0:9;
y = sin(x);
stem(y);
yu = interp(y,4); % Tăng mẫu lên 4 lần
figure;
stem(yu);
```



Hình 4-9 Ví dụ về tăng mẫu sử dụng hàm interp trong MATLAB. (a) Tín hiệu mẫu ban đầu, (b) Tín hiệu được tăng mẫu.

4.1.3 Khai triển Fourier

Khai triển Fourier là công cụ quan trọng trong phân tích hệ thống và trong mô phỏng hệ thống, nó được áp dụng trong những lĩnh vực quan trọng:

- Khai triển giữa miền tần số và miền thời gian: Các tín hiệu và hệ thống có thể được mô tả trong cả hai miền thời gian và tần số.

- Quá trình lọc: Mô phỏng quá trình lọc có thể được thực hiện hiệu quả trong miền tần số qua khai triển Fourier rời rạc (DFT)
- Phân tích phổ: sử dụng phân tích các tín hiệu trong miền tần số.

Trong mô phỏng các tín hiệu biểu diễn đều ở dạng rời rạc về thời gian nên việc chuyển đổi qua lại giữa hai miền thời gian và tần số được thực hiện bởi cặp biến đổi Fourier rời rạc (DFT) và khai triển Fourier ngược rời rạc (IDFT). Khai triển Fourier của một vector mẫu rời rạc *f*[*k*] có N mẫu được thực bởi DFT như sau:

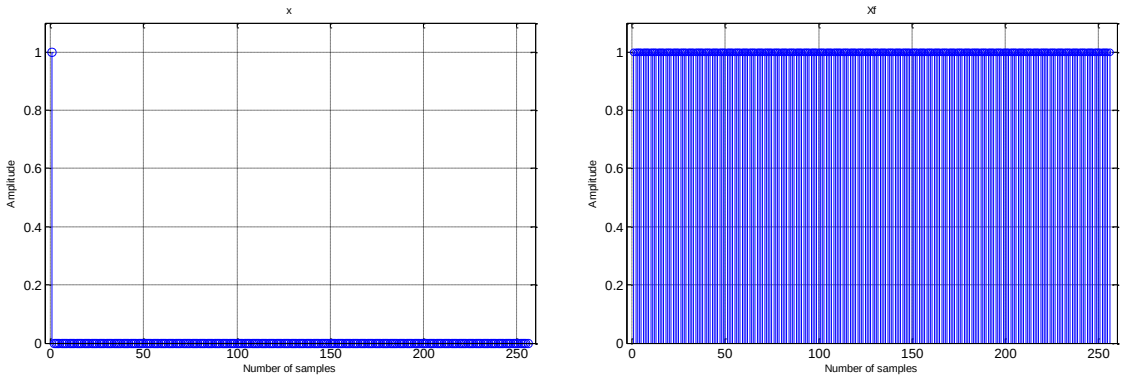
$$F[n]=\sum_{k=0}^{N-1} f[k]e^{-j\frac{2\pi}{N}nk} \qquad \text{với (n = 0÷N-1)} \qquad (4.20)$$

và khai triển IDFT là

$$f[k]=\frac{1}{N}\sum_{n=0}^{N-1} F[n]e^{+j\frac{2\pi}{N}nk} \qquad \text{với (k = 0÷N-1)} \qquad (4.21)$$

Chú ý các hệ số *F*[*n*] là phức. Trong MATLAB cặp biến đổi này được thực hiện qua hàm `fft` và `ifft`. Ngoài các hàm này, một số hàm khác về biến đổi Fourier trong MATLAB cũng hay được sử dụng trong biểu diễn phổ như hàm `fftshift`. Một ví dụ về sử dụng hàm `fft` để biểu diễn phổ tín hiệu được cho trong đoạn mã dưới đây.

```
% Ví dụ tăng mẫu sử dụng hàm fft trong MATLAB
x = zeros(1,256);
x(1) = 1;
N = length(x);
stem(x);
xf = fft(x,N); % Tăng mẫu lên 4 lần
figure;
stem(xf);
```

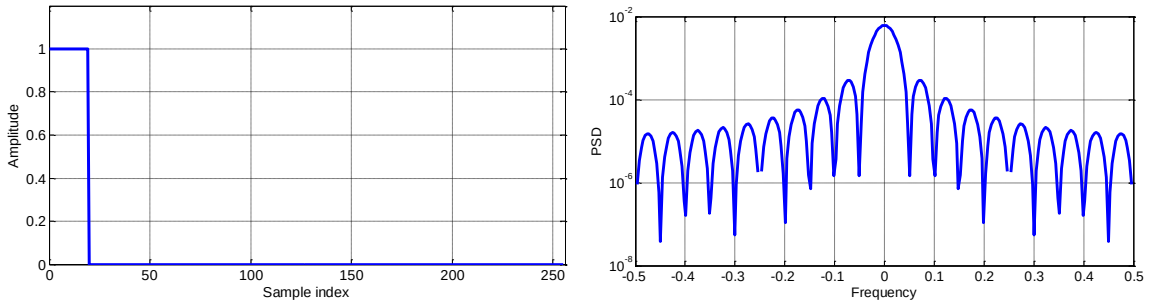


Hình 4-10 Ví dụ thực hiện khai triển Fourier sử dụng hàm `fft`

Trong mô phỏng, đại lượng mật độ phổ công suất (PSD) thường hay được sử dụng trong phân tích một tín hiệu. Một hàm tính toán PSD có thể dễ dàng được xây dựng từ hàm `fft` như cho ở đoạn mã dưới đây.

```
% Ví dụ tăng mẫu sử dụng hàm fft trong MATLAB
function [f,Pf] = spectrocal(t,x)
% Vi du chuong trinh tinh toan spectrum
% t - vector thời gian
% x - vector mẫu đầu vào
% f - vector tần số
% Pf - estimated PSD of x

Ns = length(x);
Ts = t(2)-t(1);

f = (-Ns/2:Ns/2-1)/(Ns*Ts);    % Tính vector tần số
Pf = fft(x,Ns);                % Khai triển Fourier
Pf = fftshift(Pf)/Ns;    % Dịch thành phần tần số 0 về trung tâm phổ
Pf = abs(Pf).^2;
```

```
% Ví dụ sử dụng hàm ước tính PSD spectrocal
>> t = 0:255;
>> x = zeros(1,length(t));
>> x(1:20) = 1;
>> plot(t,x);grid;
>> [f,Xf] = spectrocal(t,x);
>> semilogy(f,Xf);grid;
```



Hình 4-11 Ví dụ ước tính PSD của một xung vuông.

4.2 Mô phỏng nguồn tín hiệu

Khối đầu tiên trong hệ thống thông tin để mô phỏng đó chính là nguồn tín hiệu. Có hai kiểu nguồn cơ bản đó là: nguồn tín hiệu tương tự và nguồn tín hiệu số.

4.2.1 Nguồn tín hiệu tương tự

Để mô phỏng tường minh hoạt động các khối mã hóa nguồn, nguồn tín hiệu tương tự phải được sử dụng và cần được mô phỏng. Trong thực tế, các tín hiệu tương tự dạng đo kiểm được sử dụng trong mô phỏng hơn là mô hình hóa một

nguồn tín hiệu tương tự cụ thể nào đó. Các tín hiệu đo thử này có thể được sử dụng như các tín hiệu đầu vào các khối mã hóa nguồn hoặc khối điều chế trong hệ thống truyền dẫn tương tự.

Nguồn đơn tần thường là tín hiệu đo kiểm phổ biến được sử dụng trong hệ thống thông tin. Trong mô phỏng, tín hiệu đơn tần này sẽ cần được lấy mẫu và biểu diễn ở dạng rời rạc về thời gian

$$x(t_k) = A\cos(2\pi f_0 t_k + \varphi) \tag{4.22}$$

Chuỗi thời điểm lấy mẫu $\{t_k\}$ luôn có khoảng cách bằng nhau trong mô phỏng và bằng với chu kỳ lấy mẫu T_s hay $t_k = kT_s$. Tần số f_0 và biên độ A có thể được biến đổi để thu được đáp ứng tần và đáp ứng công suất hay biên độ của hệ thống. Trong các mô hình lý thuyết, tham số pha φ thường được giả sử có phân bố đều trên khoảng [0, 2π]. Tuy nhiên, đáp ứng của một hệ thống thường không nhạy cảm với giá trị của φ, do vậy trong mô phỏng φ được cố định tại một số giá trị bất kì. Trong các hệ thống thông dải, tần số f_0 được đo từ tần số trung tâm f_c. Tín hiệu ở dạng liên tục là

$$x(t) = A\cos[2\pi(f_c + f_0)t + \varphi] \tag{4.23}$$

và tín hiệu đường bao phức của tín hiệu này được lấy mẫu để cho mô phỏng sẽ là

$$\tilde{x}(k) = A\exp(j2\pi k f_0 / f_s)\exp(j\varphi) \tag{4.24}$$

trong đó $f_s = T_s^{-1}$ là tần số lấy mẫu. Chú ý f_s thường được thiết lập ở giá trị bằng 8 đến 16 lần f_0 để đảm bảo biểu diễn chính xác dạng sóng và phổ của tín hiệu.

Một tập các tín hiệu đơn tần được gọi là tín hiệu đa tần cũng thường được sử dụng để đánh giá méo điều chế tương hỗ trong các hệ thống phi tuyến. Đặc trưng biểu diễn đường bao phức cho tín hiệu đa tần có dạng

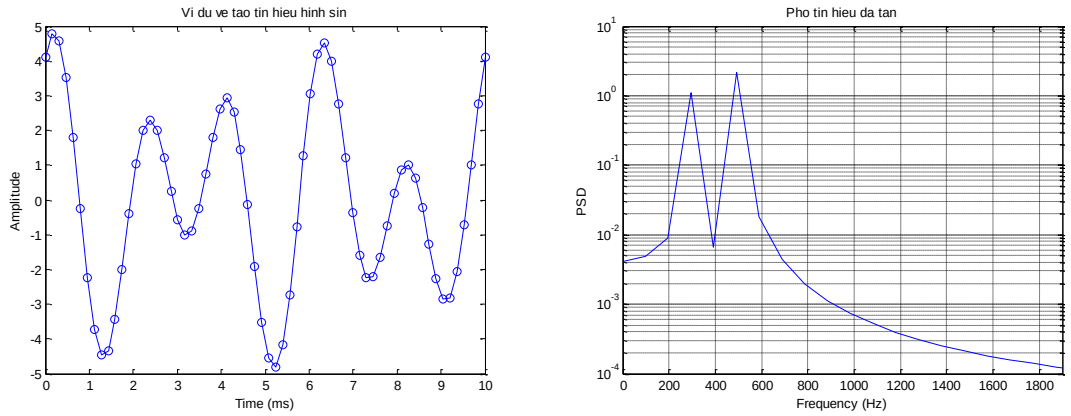
$$\tilde{x}(k) = \sum_{n=1}^{M} A_n \exp\left(j2\pi k \frac{f_n}{f_s} + j\varphi_n \right) \tag{4.25}$$

trong đó A_n, φ_n và f_n đặc trưng cho biên độ, pha và tần số của tín hiệu thứ n:

$$x_n(t) = A_n \cos[2\pi(f_c + f_n)t + \varphi_n] \tag{4.26}$$

Trừ phi có một mối quan hệ cụ thể đã biết giữa φ_n, đại lượng này có thể được giả sử độc lập nhau và có phân bố đều trên khoảng [0, 2π]. Đoạn mã MATLAB dưới đây cho ví dụ về cách tạo một tín hiệu đa tần và biểu diễn phổ của nó.

```
% Chương trình ví dụ tạo tín hiệu đa tần và vẽ phổ
% Thiết lập các tham số
A1 = 3;             % amplitude
f1 = 500;           % frequency [Hz]
phi1 = -pi/4;       % Phase [rad]
A2 = 2;             % amplitude
f2 = 300;           % frequency [Hz]
phi2 = 0;           % Phase [rad]
N = 2^6;            % number of samples
T0 = 0;             % start time [s]
Tf = 10e-3;         % end time [s]
Ts = (Tf-T0)/(N-1);  % Tính chu kỳ mẫu
fs = 1/Ts;          % Tần số lấy mẫu [Hz]
% Tạo tín hiệu đa tần
t = T0:Ts:Tf;
x = A1*cos(2*pi*f1*t+phi1)+A2*cos(2*pi*f2*t+phi2);
% Biểu diễn kết quả
figure(1);
plot(t*1e3,x,'o-');
xlabel('Time (ms)');
ylabel('Amplitude');
title('Vi du ve tao tin hieu hinh sin');
% Ước tính phổ
figure(2);
[f,Xf] = spectrocal(t,x);
semilogy(f,Xf);
xlabel('Frequency (Hz)');
ylabel('PSD');
title('Pho tin hieu da tan');
grid;
axis([0 2000, 3 1e-4]);
```



Hình 4-12 Ví dụ nguồn tín hiệu tương tự đa tần và phổ tương ứng

4.2.2 Nguồn tín hiệu số

Một nguồn số đơn giản là một nguồn rời rạc và có giá trị hữu hạn trong một bảng mẫu tự xác định. Các phần tử của nguồn số được xem như là các thực thể rút gọn hoặc logic. Tín hiệu số là một dạng sóng mang thông tin số. Sự khác biệt này là cần thiết vì có nhiều cách khác nhau để tạo ra dạng sóng cho cùng một nguồn số.

Nguồn tín hiệu số trong mô phỏng có thể được đặc trưng bởi ba tham số chính:

- Kiểu mẫu tự hay danh sách các ký hiệu thông tin (symbol) có thể mà nguồn sẽ tạo ra. Trong hệ thống viễn thông, tập kiểu mẫu tự thường bao gồm M ký hiệu với $M = 2^n$ trong đó n là một số nguyên được biểu diễn như $A = \{0, 1, \ldots, M\text{-}1\}$ hoặc $A = \{\pm 1, \pm 3, \ldots, \pm (M\text{-}1)\}$. Trường hợp đơn giản nhất và thường sử dụng với $M = 1$ ta có $A = \{0, 1\}$ cho các nguồn nhị phân và các ký hiệu lúc này được gọi là bit. Ngoài ra các ký hiệu có thể có giá trị phức cũng được sử dụng, ví dụ như $A = \{\pm 1, \pm j\}$.
- Xác suất ưu tiên phát hay tần số xuất hiện tương đối của mỗi ký hiệu mà nguồn tạo ra. Nguồn số sinh ra các ký hiệu trong bảng mẫu tự một cách ngẫu nhiên giống với thực tế. Ví dụ nguồn nhị phân cho mô phỏng sinh ra các bít 0 và 1 với xác suất bằng nhau.
- Tốc độ ký hiệu (symbol rate) tức số lượng ký hiệu thông tin mà nguồn sinh ra trong một đơn vị thời gian (baudrate). Tốc độ này quan hệ với tốc độ bít (R_b) như sau:

$$R_b = R.\log_2(M) \tag{4.27}$$

Như vậy trong mô phỏng các nguồn số thường tạo ra một cách ngẫu nhiên, do đó nó sẽ đòi hỏi các hàm tạo tín hiệu ngẫu nhiên khi thực hiện mô phỏng.

4.2.3 Nguồn tín hiệu ngẫu nhiên

Trong các hệ thống thông tin, các tín hiệu dạng sóng điện áp hoặc dòng trong truyền phát, xử lý thông tin cũng như cho điều khiển các hệ thống thiết bị là hàm của thời gian và được phân loại là xác định hoặc ngẫu nhiên. Các tín hiệu xác định có thể được mô tả bởi các hàm toán học với thời gian t như là một biến độc lập. Ngược với tín hiệu xác định, một tín hiệu ngẫu nhiên luôn có một số phần tử không xác định trong nó và do đó không thể xác định được giá trị xác định của nó tại bất kỳ thời điểm nào.

Các mô hình bất định hay ngẫu nhiên có vai trò quan trọng trong phân tích và thiết kế các hệ thống thông tin và do vậy được sử dụng nhiều trong mô phỏng hệ thống truyền thông. Các mô hình này được sử dụng trong nhiều ứng dụng khác nhau trong đó các tín hiệu cũng như các tham số hệ thống có thể thay đổi một cách ngẫu nhiên. Như có thể thấy qua các dạng sóng tín hiệu truyền qua kênh truyền bị

tác động của nhiễu. Tập hợp các dạng sóng biểu thị các biến đổi ngẫu nhiên được mô phỏng sử dụng các biến và các quá trình ngẫu nhiên. Đối với mục đích phân tích, tập các dạng sóng có thể được mô hình hóa bởi một quá trình ngẫu nhiên $x(t)$, trái lại tập các bít nhị phân có thể được mô hình hóa bởi một chuỗi ngẫu nhiên $\{A(k)\}$. Lưu ý, trong mô phỏng các quá trình ngẫu nhiên cũng được lấy mẫu.

Trong khi mô hình quá trình ngẫu nhiên mô tả một tập các hàm thời gian, thì giá trị tức thời của một tập có thể được mô hình hóa bởi một biến ngẫu nhiên. Ví dụ giá trị tức thời của $x(t)$ tại $t = t_0$ có thể được mô hình hóa như một biến ngẫu nhiên có giá trị nhị phân với các xác suất p_1 và p_2, trong đó p_1 là xác suất để $x(t)$ tại $t = t_0$ có giá trị +1 và p_2 là xác suất để $x(t)$ tại $t = t_0$ có giá trị -1. Phân bố biên độ tức thời của nhiễu $n(t)$ có thể được mô hình hóa như một biến ngẫu nhiên liên tục có một hàm mật độ xác suất phù hợp. Do vậy các kiến thức về toán xác suất thống kê sẽ cung cấp cơ sở về các mô hình các biến và các quá trình ngẫu nhiên sử dụng trong mô phỏng hệ thống truyền thông.

Trong mô phỏng hệ thống truyền thông, các mô hình cho các biến ngẫu nhiên hay được sử dụng trong mô phỏng bao gồm mô hình phân bố đều và mô hình phân bố chuẩn.

- Phân bố đều: trong đó tất cả các giá trị trong một khoảng xác định có cùng xác suất và được biểu diễn qua hàm mật độ xác suất

$$f(x) = \frac{1}{b-a} \quad \text{khi } a \le x \le b \tag{4.28}$$

  trong đó a và b là giá trị nhỏ nhất và lớn nhất tương ứng trong khoảng giá trị. Trong mô phỏng, các giá trị mẫu từ phân bố đều có thể được sử dụng để tạo ra các mẫu từ các phân bố xác suất khác.

- Phân bố chuẩn (Gauss): có biến ngẫu nhiên được mô tả qua hàm mật độ xác suất

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[\frac{-(x-\mu)^2}{2\sigma^2}\right] \quad \text{cho } -\infty < x < \infty \tag{4.29}$$

  trong đó μ và σ^2 là giá trị trung bình và phương sai của hàm phân bố. Mô hình phân bố này thường được sử dụng trong mô phỏng nhiễu và cũng được sử dụng như một sự gần đúng cho một số phân bố khác.

- Phân bố Rayleigh: có biến ngẫu nhiên $X = \left(X_1^2 + X_2^2\right)^{1/2}$, trong đó X_1 và X_2 là các biến ngẫu nhiên Gauss độc lập nhau có trung bình bằng 0 và phương sai σ^2, được mô tả qua hàm mật độ xác suất

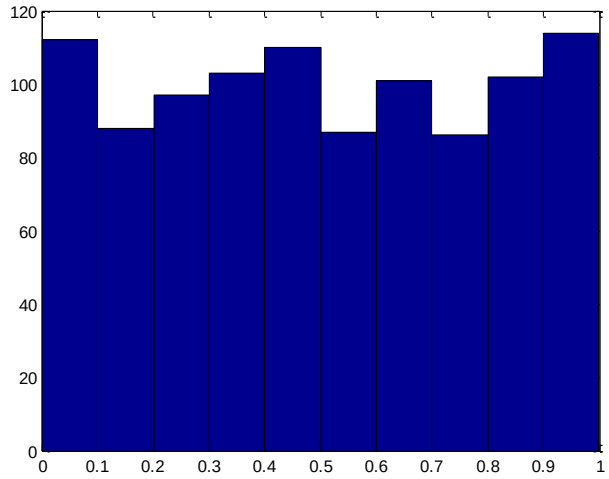
$$f(x) = \frac{x}{\sigma^2}\exp\left[\frac{-x^2}{2\sigma^2}\right] \quad \text{cho } x > 0 \tag{4.30}$$

Mô hình phân bố này được sử dụng trong mô hình kênh pha đinh, đường bao của nhiễu Gauss băng hẹp.

Trong MATLAB có các hàm sẵn có để tạo ra các số ngẫu nhiên theo các hàm phân bố khác nhau.

- Tạo các số ngẫu nhiên phân bố đều: sử dụng hàm `rand`

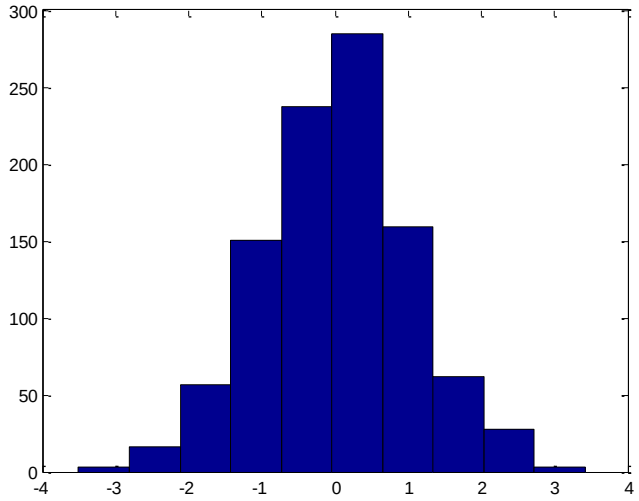
```
% Ví dụ tạo số ngẫu nhiên phân bố đều trong MATLAB
% Tạo ma trận 5x10 các số ngẫu nhiên phân bố đều trong khoảng [0,1];
>> x = rand(5,10)
% Tạo ma trận số ngẫu nhiên mxn phân bố đều trong khoảng [a, b]
>> x = a + (b-a) * rand(m,n)
% Tạo các số nguyên ngẫu nhiên phân bố đều trên tập 1:n
>> x = ceil(n.*rand(100,1));
% Tạo vectơ hàng 1000 số ngẫu nhiên phân bố đều trong khoảng [0,1],
% và vẽ biểu đồ phân bố xác suất
>>  x = rand(1,1000);
>>  hist(x,10);
```



Hình 4-13 Biểu đồ mật độ xác suất phân bố đều của 1000 mẫu.

- Tạo các số ngẫu nhiên phân bố chuẩn: sử dụng hàm `randn`

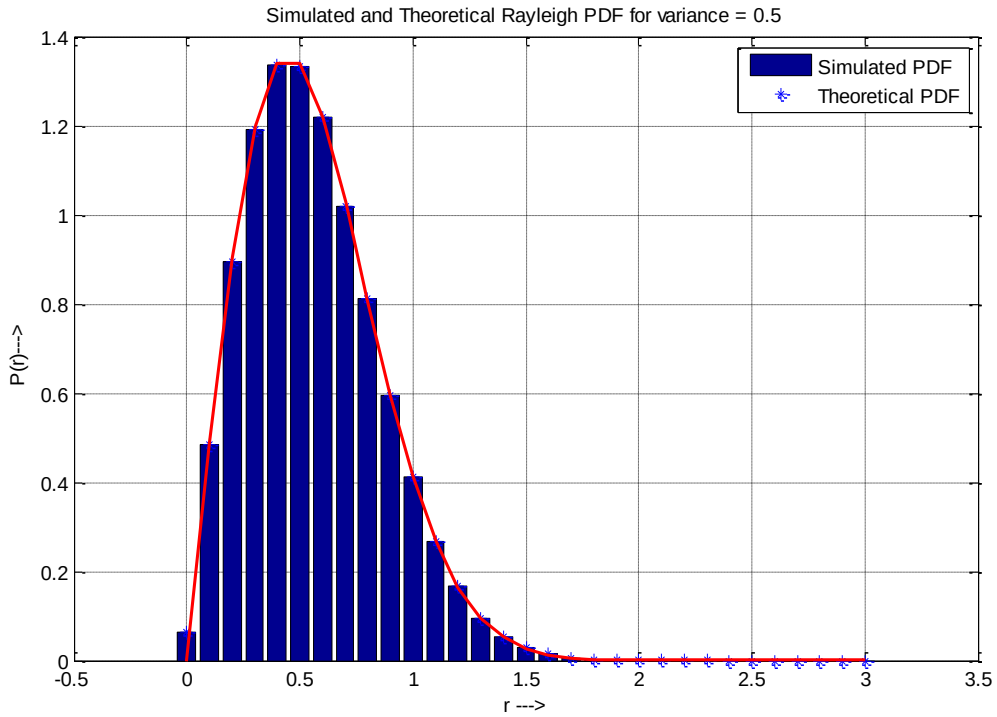
```
% Ví dụ tạo số ngẫu nhiên phân bố chuẩn trong MATLAB
% Tạo các số ngẫu nhiên phân bố chuẩn có trung bình bằng 0
% và độ lệch chuẩn bằng 1
>> x = randn(m,n)
% Tạo các số ngẫu nhiên phân bố chuẩn có trung bình bằng m
% và phương sai v
>> x = m + sqrt(v) * randn(m,n)
% Tạo vectơ hàng 1000 số ngẫu nhiên phân bố chuẩn có trung bình 0
% và độ lệch chuẩn bằng 1
>>  x = randn(1,1000);
>>  hist(x,10);
```



Hình 4-14 Biểu đồ mật độ xác suất phân bố chuẩn của 1000 mẫu.

- Tạo các số ngẫu nhiên phân bố Rayleigh: sử dụng hàm `rand` hoặc `randn`

```
% Ví dụ tạo số ngẫu nhiên phân bố Rayleigh trong MATLAB
N = 1000000;      % Số lượng mẫu tạo ra
v = 0.2;          % Phương sai của biến ngẫu nhiên Gauss
%---------------------------------------
% Các biến ngẫu nhiên Gauss độc lập x và y có trung bình 0
% và phương sai đơn vị
x = randn(1, N);
y = randn(1, N);

r = sqrt(v*(x.^2 + y.^2)); % Biến ngẫu nhiên Rayleigh
% Xác định cỡ bước và dải giá trị để vẽ biệu đồ mật độ xác suất
step = 0.1; range = 0:step:3;
% Xác định giá trị tần suất và tính gần đúng bằng đường cong pdf
h = hist(r,range);
approxPDF = h/(step*sum(h)); % Mật độ xác suất mô phỏng
% Mật độ xác suất lý thuyết
theoretical = (range/v).*exp(-range.^2/(2*v));
bar(range, approxPDF); hold on;
plot(range, theoretical,'r'); hold off;
```



Hình 4-15 Mật độ xác suất của biến ngẫu nhiên phân bố Rayleigh

- Một số hàm tạo số ngẫu nhiên khác trong MATLAB:

```
% Hàm randint
% Tạo ma trận mxn các số 0 và 1 có xác suất bằng nhau
>>  x = randint(m,n);
% Ví dụ:
>>  x = randint(1,10)
x =
     0     0     1     1     1     0     1     1     0     0
% Tạo ma trận mxn có các giá trị phân bố đều trong dải từ 0 đến 7
>> x = randint(m, n, [0, 7]); hoặc
>> x = randint(m,n, 8);
```

```
% Hàm randsrc
% Tạo ma trận mxn các số -1 và 1 có xác suất bằng nhau
>>  x = randsrc(m,n);
% Ví dụ:
>>  x = randsrc(1,10)
x =
    -1     1     1    -1    -1    -1    -1     1     1     1
% Tạo ma trận mxn có các giá trị phân bố đều trong tập {-3,-1,1,3}
>> x = randsrc(10,10,[-3 -1 1 3]); hoặc
>> x = randsrc(10,10,[-3 -1 1 3; .25 .25 .25 .25]);

% Hàm randerr tạo nguồn lỗi ngẫu nhiên
>> err = randerr(3,4)
```

4.3 Mã hóa

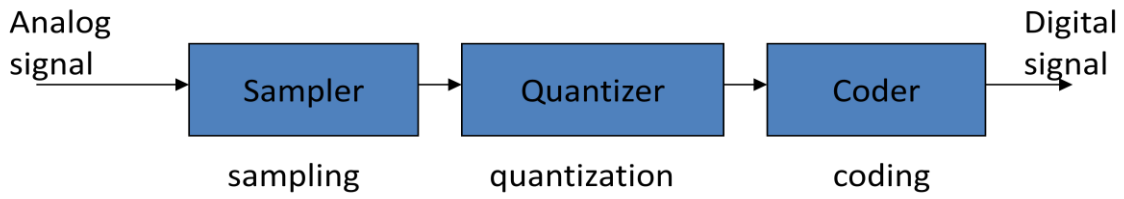
Quá trình mã hóa trong hệ thống truyền thông được phân ra thành các loại cơ bản bao gồm: mã hóa nguồn, mã hóa đường và mã hóa kênh.

4.3.1 Mã hóa nguồn

Mã hóa nguồn thực hiện chuyển đổi nguồn tin thành một dạng phù hợp cho truyền dẫn số trên một kênh truyền xác định. Nếu nguồn tin là rời rạc, quá trình mã hóa sẽ chuyển đổi các ký tự đơn hoặc nhóm các ký tự thành các nhóm ký tự logic tương ứng. Quá trình sắp xếp này thường nhằm giảm các phần dư thừa để nén dữ liệu. Nếu nguồn tin là liên tục, quá trình mã hóa nguồn liên quan đến quá trình chuyển đổi từ tương tự sang số (ADC) và sẽ được đề cập cụ thể trong phần này.

Quá trình chuyển đổi từ tương tự sang số như mô tả trong hình 4- gồm 3 bước chính đó là: lấy mẫu, lượng tử hóa và mã hóa thành chuỗi nhị phân. Bước đầu tiên là quá trình lấy mẫu đã được mô tả trong phần trước. Bước thứ hai là lượng tử hóa được coi là trái tim của quá trình ADC. Tại bước này, các mẫu $x(kT_s)$ được chuyển đổi thành một biến rời rạc x_q có Q giá trị.

Trong quá trình lượng tử hóa, dải biên độ tín hiệu đầu vào được phân chia thành các khoảng có kích cỡ Δ; khoảng thứ i tương ứng với dải $i\Delta \pm (\Delta/2)$. Nếu giá trị mẫu nằm trong khoảng thứ i thì nó sẽ được gán số nguyên i và i có thể được chuyển thành dạng nhị phân (bước mã hóa) cho ra chuỗi bít đầu ra. Có hai kiểu lượng tử hóa đó là: lượng tử hóa đều và lượng tử hóa không đều. Quá trình này cũng còn được gọi là quá trình điều xung mã (PCM). Lượng tử hóa đều được thực hiện khi bước lượng tử Δ không đổi hay bằng nhau. Quá trình lượng tử hóa sẽ gây ra sai số $x_q - x$ gọi là nhiễu lượng tử.

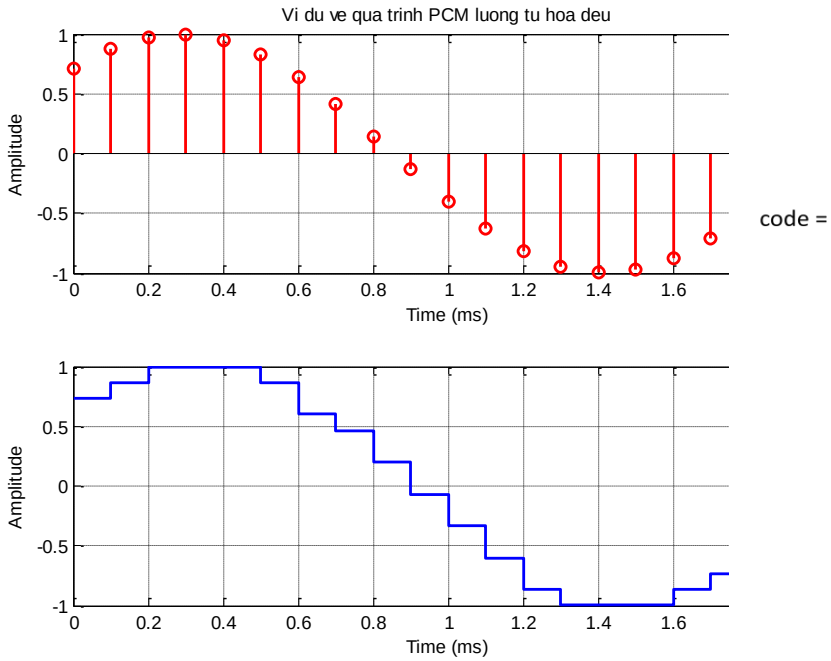


Hình 4-16 Quá trình PCM thực hiện chuyển đổi tín hiệu tương tự sang số (ADC)

Một chương trình MATLAB thực hiện quá trình PCM lượng tử hóa đều được viết dưới dạng hàm function được cho dưới đây và ví dụ kết quả sử dụng hàm này được cho trong hình 4-17.

```
% Ví dụ về mã hóa nguồn sử dụng quá trình PCM lượng tử hóa đều
function [code,xq,sqnr] = uniform_pcm(x,M)
% x = input sequence
% M = number of quantization levels
% code = the encoded output
% xq = quantized sequence before encoding
% sqnr = signal to quantization noise ratio in dB

Nb = log2(M);      % Số bit trên mỗi từ mã
Amax = max(abs(x));
delta = 2*Amax/(M-1);
Mq = -Amax:delta:Amax;
Ml = 0:M-1;

xq = zeros(size(x));
xcode = xq;
for k = 1:M
    ind = find(x > Mq(k)-delta/2 & x <= Mq(k)+delta/2);
    xq(ind) = Mq(k);
    xcode(ind) = Ml(k);
end
sqnr = 20*log10(norm(x)/norm(x-xq)); % Tính tỉ số trên nhiễu lượng tử
code = de2bi(xcode,Nb,'left-msb');  % Chuyển đổi sang từ mã nhị phân
```

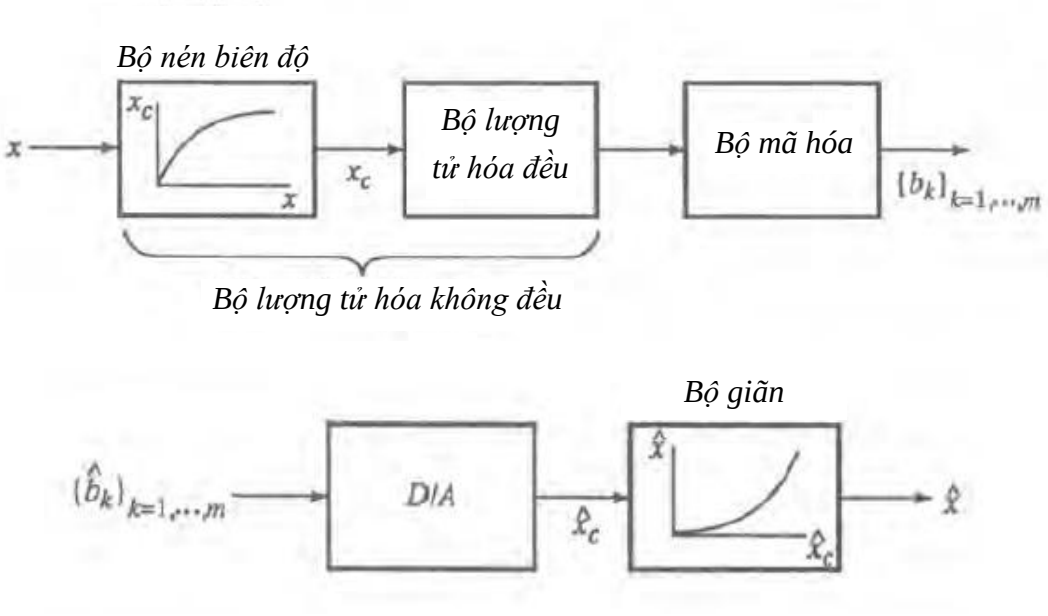
```
>> [code,xq,sqnr] = uniform_pcm(x,16);
```




```
code =

     1     1     0     1
     1     1     1     0
     1     1     1     1
     1     1     1     1
     1     1     1     1
```

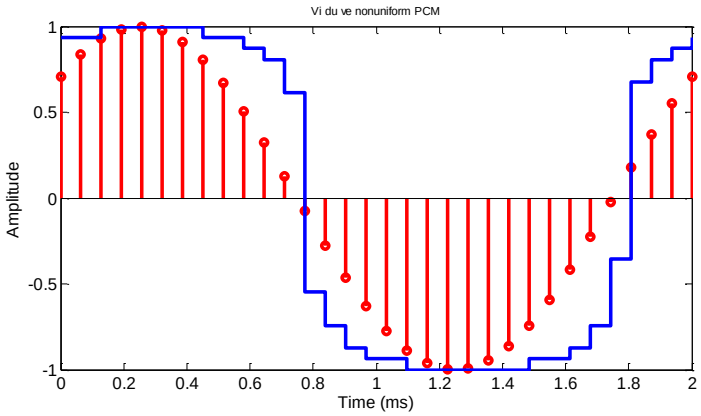
Hình 4-17 Ví dụ kết quả sử dụng hàm `uniform_pcm` cho quá trình ADC

Quá trình lượng tử hóa không đều chia dải biên độ thành các khoảng không đều nhau. Trong thực tế quá trình này được thực hiện thành 2 bước như mô tả trong hình 4-18. Trước hết mẫu tín hiệu được xử lý qua một phần tử phi tuyến gọi là bộ nén và sau đó được đưa vào bộ lượng tử hóa đều. Quá trình 2 bước này tương đương với lượng tử hóa không đều của mẫu đầu vào ban đầu.

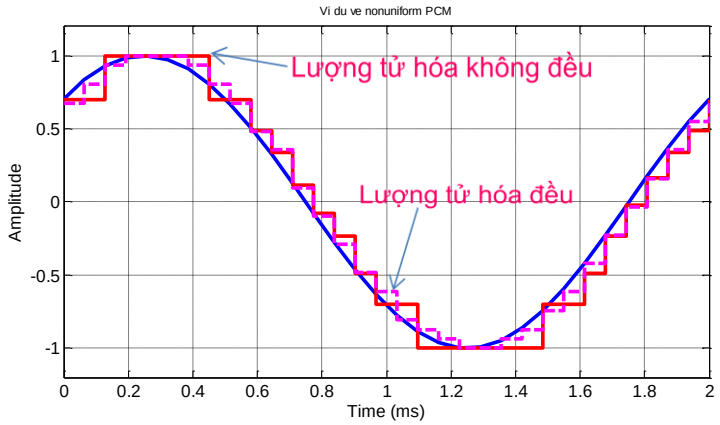


Hình 4-18 Quá trình thực hiện lượng tử hóa không đều

```
% Ví dụ về quá trình PCM lượng tử hóa không đều
mu = 255;     % Hệ số nén
M = 32;       % Số lượng mức lượng tử
[y,amax] = mulaw(x,mu);  % Nén tín hiệu
[code,yq,sqnr] = uniform_pcm(y,M);  % Mã hóa
xq = invmulaw(yq,mu);    % Giải nén tín hiệu
xq = xq*amax;
sqnr = 20*log10(norm(x)/norm(x-xq)); % in dB

%% Hàm nén theo luật μ
function [y,amax] = mulaw(x,mu)
% Hàm nén theo luật μ với x là vector đầu vào, mu là hệ số nén
amax = max(abs(x));      % Xác định biên độ cực đại tín hiệu đầu vào
xn = x/amax;             % Chuẩn hóa tín hiệu đầu vào
y = sign(x).*log(1+mu*abs(xn))/log(1+mu);
y = y*amax;
```



Hình 4-19a Ví dụ quá trình lượng tử hóa không đều sử dụng hàm nén μ.

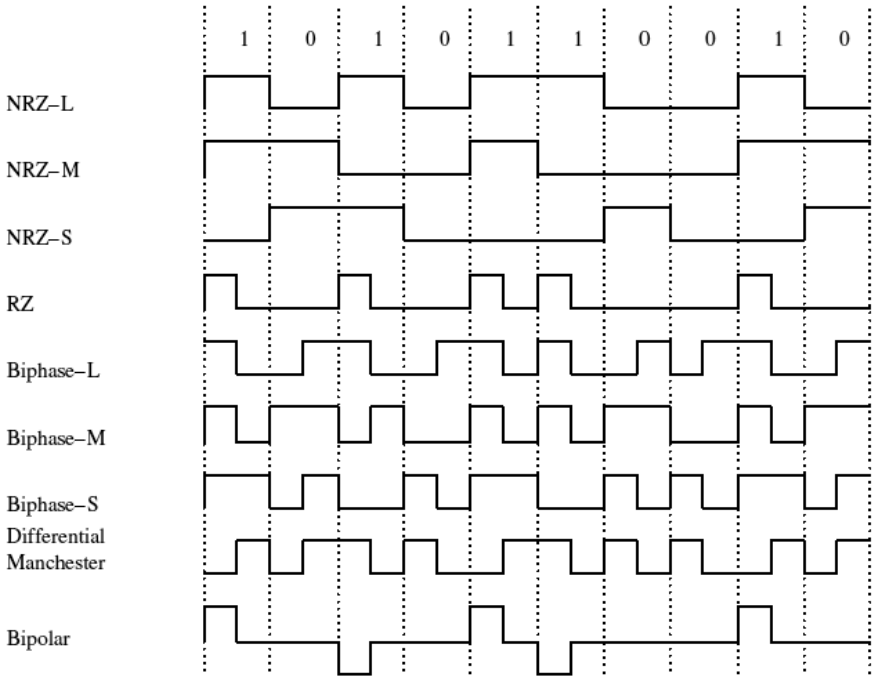


Hình 4-19b Ví dụ quá trình lượng tử hóa không đều sử dụng hàm nén μ.

Quá trình giải mã được thực hiện ngược lại quá trình mã hóa và dễ dàng xây dựng các hàm tương như quá trình mã hóa. Để đảm bảo khôi phục lại dạng sóng tương tự từ các mẫu chuyển đổi, các kỹ thuật nội suy như đã trình bày trong phần 4.1 cần thiết được sử dụng.

4.3.2 Mã đường truyền

Mã hóa đường truyền là một kiểu mã hóa được thực hiện để tạo dạng phổ tín hiệu hay để đi kèm các tính chất thống kê xác định trong chuỗi tín hiệu đảm bảo một mật độ chuyển tiếp xác định để hỗ trợ quá trình đồng bộ. Một số các mã đường truyền thông dụng được cho thấy trong hình 4-20.



Hình 4-20 Một số mã đường phổ biến

Quá trình mã hóa đường truyền về cơ bản được thực hiện theo 2 bước:

- Bước 1: Sắp xếp logic thực hiện chuyển đổi một chuỗi nhị phân hay M-mức thành một chuỗi khác có các tính chất mong muốn.
- Bước 2: Chuyển đổi các ký hiệu thành dạng sóng, quá trình này đôi khi cũng được xem như là điều chế băng gốc.

Đối với bước đầu tiên khi mô phỏng mã đường truyền chỉ là sự sắp xếp logic theo luật chuyển đổi của từng loại mã khác nhau. Còn bước thứ hai chuyển đổi sang dạng sóng sẽ đòi hỏi việc biểu diễn dạng sóng cho mỗi ký hiệu trên một cửa sổ thời gian tương ứng với chu kỳ của ký hiệu. Quá trình này cũng sẽ liên quan đến quá trình lấy mẫu để đảm bảo biểu diễn chính xác dạng sóng yêu cầu. Quá trình tạo dạng sóng đơn giản để biểu diễn một chuỗi xung tuần hoàn được cho thấy trong ví dụ tạo chuỗi xung vuông sau đây.

- Chuỗi xung vuông tuần hoàn được thực hiện bằng việc biểu diễn dạng sóng trên một cửa sổ thời gian tương ứng với chu kỳ của xung, sau đó dạng sóng này sẽ được dịch hay lặp lại trên vector thời gian và có thể được biểu diễn như sau:

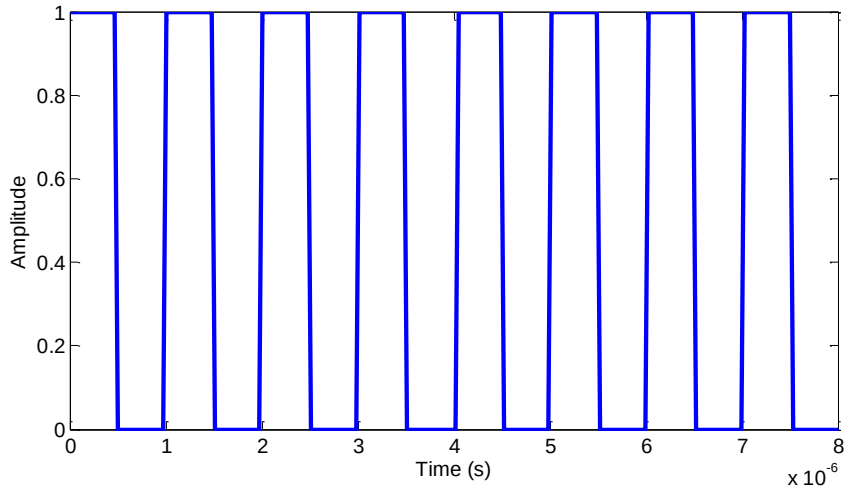
$$x(t) = \sum_{k=0}^{N} p(t - kT) \tag{4.31}$$

trong đó p là hàm dạng xung, T là chu kỳ lặp của xung và N là số lượng xung cần biểu diễn. Đối với chuỗi xung vuông, hàm xung vuông được biểu diễn bởi

$$p(t) = \begin{cases} 1 & \text{khi} t \le T_w \\ 0 & \text{khi} t > T_w \end{cases} \tag{4.32}$$

với T_w là độ rộng của xung biểu diễn có giá trị thường nhỏ hơn chu kỳ xung. Đoạn mã MATLAB ví dụ về biểu diễn chuỗi xung vuông được viết dưới dạng hàm function và sử dụng hàm này được cho dưới đây:

```
% Ví dụ về biểu diễn chuỗi xung vuông
function [t,y] = myrectpulse(Tw,Rp,Ns,Np)
% Chuong trinh vi du tao chuoi xung vuong
% Tw - độ rộng xung
% Rp - tốc độ lặp xung
% Ns - số lượng mẫu
% Np - số lượng xung cần biểu diễn
% t - vector thời gian
% y - vector mẫu đầu ra

Tp = 1/Rp;               % pulse period
Timewindow = Np*Tp;      % time window
ts = Timewindow/(Ns-1); % sampling time
t = 0:ts:Timewindow;     % time vector
Nsp = round(Tp/ts);      % number of samples within Tp

y = zeros(size(t));
for k = 1:Ns
    if mod(t(k),Nsp*ts) <= Tw
        y(k) = 1;
    else
        y(k) = 0;
    end
end

%% Ví dụ về sử dụng hàm myrectpulse để vẽ chuỗi xung vuông
>> [t,y]=myrectpulse(0.5e-6,1e6,256,8);
>> plot(t,y);
```



Hình 4-21 Chuỗi xung vuông được tạo ra sử dụng hàm `myrectpulse`.

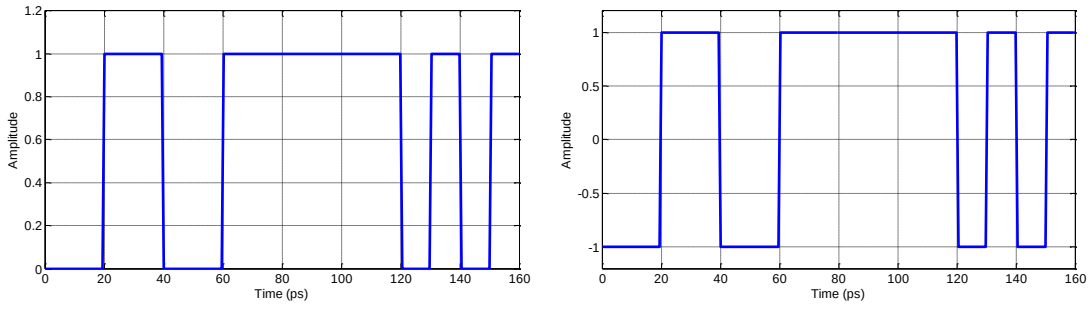
Dựa trên các bước cơ bản trong thực hiện mô phỏng mã đường truyền đã trình bày ở trên, dạng sóng một mã đường truyền có thể được biểu diễn như sau

$$x(t) = \sum_{k=0}^{N} \alpha_k \, p(t - kT) \tag{4.33}$$

trong đó α_k là ký hiệu logic được sắp xếp tại bước đầu tiên theo quy luật chuyển đổi mã đường, *p(t)* là hàm dạng xung và thường là dạng xung vuông xác định theo công

thức (4.32). Một số ví dụ về biểu diễn các mã đường truyền đơn giản được cho trong các đoạn mã dưới đây được xây dựng dưới dạng hàm function và cách sử dụng hàm.

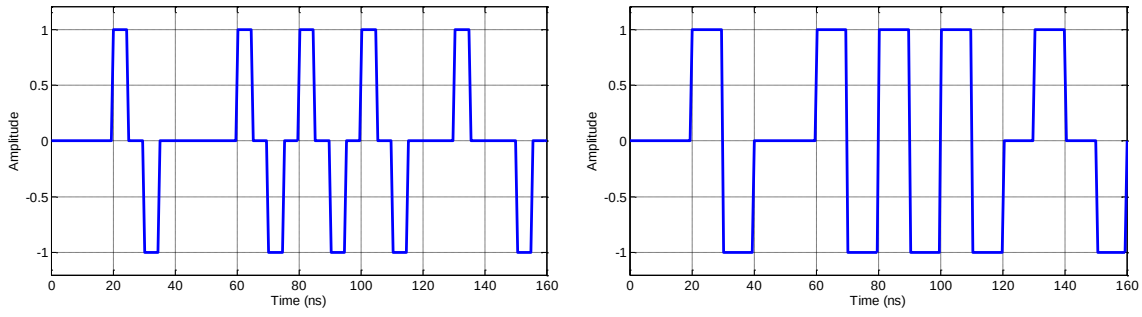
```
% Ví dụ về biểu diễn mã đường NRZ
function [t,y,code] = nrzcode(d,R,Ns,type)
% Chuong trinh vi du ve ma duong truyen NRZ
% d - chuỗi dữ liệu
% R - tốc độ dữ liệu
% Ns - tổng số mẫu biểu diễn
% t - vector thời gian
% y - vector mẫu đầu ra
% type - kiểu mã (unipolar - 'unipol' or polar - 'pol')

Tb = 1/R;                % chu kỳ bit
Nb = length(d);          % số lượng bit
Timewindow = Nb*Tb;      % cửa sổ thời gian biểu diễn
ts = Timewindow/(Ns-1); % chu kỳ lấy mẫu
t = 0:ts:Timewindow;     % vector thời gian
y = zeros(size(t));
code = [];
if nargin <=3
    type = 'unipol';
end
for k = 1:Ns
    n = fix(t(k)/Tb)+1;
    if n >= Nb
        n = Nb;
    end
    switch (type)
        case 'unipol'
            y(k) = d(n);
            code(n) = d(n);
        case 'pol'
            y(k) = 2*d(n)-1;
            code(n) = 2*d(n)-1;
    end
end

%% Ví dụ về sử dụng hàm nrzcode để vẽ chuỗi xung vuông
>>h = [0 0 1 1 0 0 1 1 1 1 1 1 0 1 0 1];
>>[t,y]=nrzcode(h,1e6,256);
>> plot(t,y);
```



Hình 4-22 Biểu diễn mã đường NRZ sử dụng hàm `nrzcode`

Ví dụ biểu diễn mã AMI:

```
% Ví dụ về biểu diễn mã đường NRZ
function [t,y,code] = amicode(d,R,Ns,type)
% Chuong trinh vi du ve ma AMI
% d - the data sequence
% R - the data rate
% Ns - the number of samples
% t - the time vector output
% y - the vector output of the pulse samples
% type - the type of code (NRZ - 'NRZ' or RZ - 'RZ')
...
y = zeros(size(t));                    } Giống với mã đường NRZ
code = [];
...
s = 1;                                 }
for k = 1:Nb                           }
    if d(k) == 0                       }
        code(k) = 0;                   } Sắp xếp logic theo quy tắc mã AMI
    else                               }
        s = s+1;                       }
        if mod(s,2)==0                 }
            code(k) = 1;               }
        else                           }
            code(k) = -1;              }
        end                            }
    end                                }
end                                    }
...                                    } Giống với mã đường NRZ
%% Ví dụ về sử dụng hàm amicode
>>h = [0 0 1 1 0 0 1 1 1 1 1 1 0 1 0 1];
>>[t,y,code]=amicode(h,1e8,256,'RZ');
>> plot(t,y);
```



Hình 4-23 Biểu diễn mã đường AMI sử dụng hàm `amicode`

4.3.3 Mã hóa kênh

Mục đích của mã hóa kênh là để cải thiện hiệu năng của kênh truyền bằng việc phát hiện lỗi và/hoặc sửa lỗi. Nhìn chung, các phương pháp mã hóa thuộc một trong hai loại: mã khối và mã xoắn. Trong mã hóa khối, chuỗi nguồn nhị phân có độ dài *k* bít sẽ được sắp xếp thành chuỗi nhị phân đầu ra có độ dài *n* bít; do đó tốc độ

của mã thực hiện là k/n. Một mã như vậy được xem là một mã khối (n,k) và gồm có 2^k từ mã độ dài n bít ký hiệu là c_1, c_2, …, c_2^k. Việc sắp xếp nguồn thông tin thành chuỗi bít đầu vào kênh truyền được thực hiện một cách độc lập và đầu ra của bộ mã hóa chỉ phụ thuộc vào chuỗi đầu vào hiện tại có độ dài k bít và không phụ thuộc vào chuỗi đầu vào trước đó. Trong mã xoắn, chuỗi nguồn có độ dài k_0 được sắp xếp thành n_0 bít chuỗi đầu ra nhưng các chuỗi đầu ra này không chỉ phụ thuộc vào k_0 mà còn phụ thuộc vào m đầu vào cuối cùng của bộ mã hóa, m thường được gọi là độ dài bộ nhớ. Các thuật toán về mã hóa và giải mã các mã kênh truyền không phải là nội dung của môn học này mà thuộc về các môn học trước. Do đó, nội dung phần này sẽ không đề cập đến giải thuật của các mã mà chỉ đề cập đến cách thức cơ bản để thực hiện mã hóa trong hoạt động mô phỏng.

Do quá trình mã hóa và giải mã kênh truyền đều được thực hiện trong miền số nên các hoạt động mã hóa hay giải mã được tiến hành qua các hoạt động tính toán ma trận từ mã ở trạng thái logic. Lấy mã khối tuyến tính để làm ví dụ cho các hoạt động này trong mô phỏng. Các mã hóa khối tuyến tính là lớp mã khối quan trọng và được sử dụng rộng rãi. Một mã khối là tuyến tính nếu bất kỳ tổ hợp tuyến tính của hai từ mã là một từ mã. Trong trường hợp nhị phân điều này muốn nói rằng tổng của bất kỳ hai từ mã nào là một từ mã. Trong mã khối tuyến tính, các từ mã hình thành một không gian con k chiều của một không gian n chiều. Các mã khối tuyến tính được mô tả theo ma trận tạo mã *G*, là một ma trận nhị phân k x n và mỗi từ mã *c* đầu ra bộ mã hóa được xác định dưới dạng

$$c = uG \tag{4.34}$$

trong đó *u* là chuỗi dữ liệu nhị phân có độ dài k bít (đầu vào bộ mã hóa). Một tham số quan trọng trong mã khối tuyến tính cái xác định khả năng sửa lỗi của nó đó là khoảng cách mã hay khoảng cách Hamming tối thiểu. Khoảng cách này được ký hiệu d_{min} và được xác định bởi

$$d_{\min} = \min_{i \neq j} d_H\left(c_i, c_j\right) \tag{4.35}$$

Đối với mã tuyến tính, khoảng cách nhỏ nhất bằng với trọng số nhỏ nhất của mã định nghĩa bởi

$$w_{\min} = \min_{c_i \neq 0} w\left(c_i\right) \tag{4.36}$$

có nghĩa là số lượng bít 1 nhỏ nhất trong bất kỳ từ mã khác không nào. Đoạn mã MATLAB sau minh họa hoạt động mã hóa khối tuyến tính.

```
% Ví dụ về mã khối tuyến tính (n,k)
% k = 4, n = 10
k = 4;
for i=1:2^4
    for j=k:-1:1
        if rem(i-1,2^(-j+k+1))>=2^(-j+k)
            u(i,j)=1;
        else
            u(i,j)=0;
        end
    end
end
% Định nghĩa G, ma trận tạo mã
g = [1 0 0 1 1 1 0 1 1 1;
       1 1 1 0 0 0 1 1 1 0;
       0 1 1 0 1 1 0 1 0 1;
       1 1 0 1 1 1 1 0 0 1];
c = rem(u*g,2); % Xác định từ mã đầu ra
w_min = min(sum((c(2:2^k,:))')); % Xác định khoảng cách Hamming
```

Định nghĩa tổ hợp các từ mã 4 bít đầu vào bộ mã hóa

Kết quả:

```
u =                          c =
     0 0 0 0                      0 0 0 0 0 0 0 0 0 0
     0 0 0 1                      1 1 0 1 1 1 1 0 0 1
     0 0 1 0                      0 1 1 0 1 1 0 1 0 1
     0 0 1 1                      1 0 1 1 0 0 1 1 0 0
     0 1 0 0                      1 1 1 0 0 0 1 1 1 0
     0 1 0 1                      0 0 1 1 1 1 0 1 1 1
     0 1 1 0                      1 0 0 0 1 1 1 0 1 1
     0 1 1 1                      0 1 0 1 0 0 0 0 1 0
     1 0 0 0                      1 0 0 1 1 1 0 1 1 1
     1 0 0 1                      0 1 0 0 0 0 1 1 1 0
     1 0 1 0                      1 1 1 1 0 0 0 0 1 0
     1 0 1 1                      0 0 1 0 1 1 1 0 1 1
     1 1 0 0                      0 1 1 1 1 1 1 0 0 1
     1 1 0 1                      1 0 1 0 0 0 0 0 0 0
     1 1 1 0                      0 0 0 1 0 0 1 1 0 0
```

Do hoạt động trong miền số rời rạc nên bên cạnh mô phỏng dạng sóng, mô phỏng các sự kiện rời rạc hay mô phỏng kênh rời rạc thường hay được sử dụng để đánh giá hiệu năng mã hóa kênh. Vấn đề mô phỏng kênh rời rạc sẽ được đề cập đến trong Chương 5.

Thư viện `Communications Toolbox` trong MATLAB cũng chứa các hàm mã hóa kênh sẵn có cho mục đích mô phỏng. Các hàm này được liệt kê trong phụ lục A và một ví dụ sử dụng các hàm này cũng được cho trong Chương 5.

4.4 Điều chế và giải điều chế

4.4.1 Điều chế tín hiệu tương tự

Trong điều chế tương tự, tín hiệu điều chế là tín hiệu mà biên độ của nó có thể nhận các giá trị một cách liên tục như trong thoại hay hình ảnh. Tín hiệu

$$Y(t) = AX(t)\cos\left(2\pi f_c t + \varphi\right) \tag{4.37}$$

được gọi là dạng điều biến biên độ vì $X(t)$ biến đổi biên độ của $C(t)$. Trong một kiểu tương tự, quá trình điều chế pha hoặc tần số $C(t)$ có thể được biểu diễn

$$Z(t) = A\cos\left(2\pi f_c t + \varphi + k_p X(t)\right) \tag{4.38}$$

hoặc

$$W(t) = A\cos\left(2\pi f_c t + k_f 2\pi \int X(t)dt + \varphi\right) \tag{4.39}$$

trong đó độ lệch tần số tức thời được xác định bởi

$$f_i(t) = k_f X(t) \tag{4.40}$$

và k_p và k_f là các hằng số. Khi $X(t)$ được chuẩn hóa do đó $|X(t)| \le 1$, k_p và k_f được xem như là độ lệch pha và độ lệch tần tương ứng; và tỉ lệ độ lệch tần số đỉnh và tần số cao nhất trong tín hiệu điều chế được gọi là chỉ số hay độ sâu điều chế. $Z(t)$ được gọi là tín hiệu điều chế pha (PM) và $W(t)$ được gọi là tín hiệu điều tần (FM). Lưu ý trong mô phỏng khi chỉ số điều chế lớn, độ rộng băng tần $W(t)$ có thể lớn hơn nhiều lần của $X(t)$, do vậy tốc độ lấy mẫu cần lựa chọn thích hợp.

Một cách tổng quát, một tín hiệu điều chế có thể được mô tả theo dạng cầu phương như sau

$$Y(t) = X_1(t)\cos\left(2\pi f_c t + \varphi\right) - X_2(t)\sin\left(2\pi f_c t + \varphi\right) \tag{4.41}$$

trong đó $X_1(t)$ và $X_2(t)$ là các tín hiệu băng gốc (thông thấp) có độ rộng băng tần B, và tần số sóng mang f_c thường >> B. $X_1(t)$ và $X_2(t)$ có thể là hai tín hiệu tương tự độc lập hoặc chúng có thể liên hệ duy nhất với tín hiệu chung $X(t)$ như trong các chế độ điều chế triệt một phần dải biên (VSB) hoặc điều biến đơn biên (SSB). Bảng 4.1 liệt kê các tín hiệu $X_1(t)$ và $X_2(t)$ cho một số loại điều chế tương tự.

Bảng 4-1 Bảng các thành phần của các chế độ điều chế tương tự

Chế độ điều chế	$X_1(t)$	$X_2(t)$	Chú thích
Điều chế biên độ (AM)	$a[1+k_a X(t)]$	0	k_a - chỉ số điều chế
AM cầu phương	$X_1(t)$	$X_2(t)$	
Dải diên kép (DSB)	$X(t)$	0	
Đơn biên (SSB)	$X(t)$	$\tilde{X}$(t) – khai triển Hilbert của $X(t)$	
Điều chế pha (PM)	$\cos[k_p X(t)]$	$\sin[k_p X(t)]$	k_p - độ sâu điều chế
Điều chế tần số (FM)	$\cos\left[k_f \int X(\alpha)d\alpha\right]$	$\sin\left[k_f \int X(\alpha)d\alpha\right]$	k_f – độ sâu điều chế

Trong trường hợp mô phỏng tương đương băng gốc, dạng sóng tín hiệu điều biến được biểu diễn ở dạng đường bao phức

$$\tilde{Y}(t) = [X_1(t) + jX_2(t)]e^{j\varphi} \tag{4.42}$$

Độ lệch pha ngẫu nhiên của sóng mang φ thường được giả sử là 0. Đường bao phức $\tilde{Y}(t)$ cũng có thể được biểu diễn như sau

$$\tilde{Y}(t) = R(t)e^{j\psi(t)} \tag{4.43}$$

trong đó đường bao thực $R(t)$ là

$$R(t) = \left[X_1^2(t) + X_2^2(t)\right]^{1/2} \tag{4.44}$$

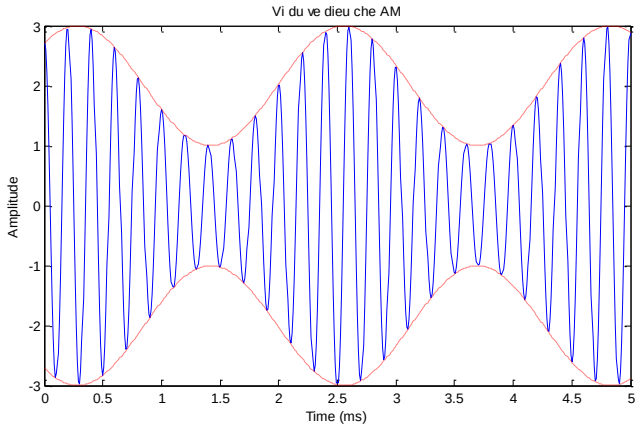
và

$$\psi(t) = \varphi + \tan^{-1}\left[\frac{X_2(t)}{X_1(t)}\right] \tag{4.45}$$

Các giá trị mẫu của $\tilde{Y}(t)$ sẽ được sử dụng trong mô phỏng. Mặc dù ta không lấy mẫu tín hiệu điều chế thông dải, các ảnh hưởng như lệch tần và đặc tính lựa chọn tần số có thể được mô phỏng sử dụng mô hình tương đương băng gốc.

Trong trường hợp điều chế có sóng mang, một ví dụ về điều chế AM được cho trong đoạn mã MATLAB dưới đây. Các kiểu điều chế khác cũng dễ dàng được thực hiện dựa trên bảng 4-1.

```
% Chuong trinh vi du ve dieu che AM
%% Thiết lập tham số
% Bản tin
A = 1;              % amplitude
f = 440;            % frequency [Hz]
phi = -pi/4;        % Phase [rad]
% Sóng mang
m = 0.5;            % modulation index
Ac = A/m;           % amplitude
fc = 5e3;           % frequency [Hz]
phi_c = 0;          % Phase [rad]

N = 2^9;            % tổng số mẫu
T0 = 0;             % thời điểm bắt đầu [s]
Tf = 5e-3;          % thời điểm kết thúc [s]
Ts = (Tf-T0)/(N-1); % chu kỳ lấy mẫu
fs = 1/Ts;          % tần số lấy mẫu [Hz]

%% Điều chế biên độ AM
% Generate sinusoid
t = T0:Ts:Tf;                   % vector thời gian
x = A*cos(2*pi*f*t+phi);        % tín hiệu bản tin
xc = Ac*cos(2*pi*fc*t+phi_c); % tín hiệu sóng mang
% Modulation
y = (1+x/Ac).*xc;
```



Hình 4-24 Ví dụ về điều chế AM với chỉ số điều chế m = 50%.

Trong thư viện `Communications Toolbox` của MATLAB có các hàm sử dụng cho điều chế tín hiệu tương tự bao gồm:

`y = ammod(x,Fc,Fs)` – sử dụng tín hiệu x để điều chế AM sóng mang Fc tại tần số lấy mẫu Fs.

`y = ssbmod(x,Fc,Fs)` - sử dụng tín hiệu x để điều chế SSB sóng mang Fc tại tần số lấy mẫu Fs.

`y = fmmod(x,Fc,Fs,freqdev)` – sử dụng tín hiệu x để điều chế FM sóng mang Fc tại tần số lấy mẫu Fs với độ lệch tần freqdev.

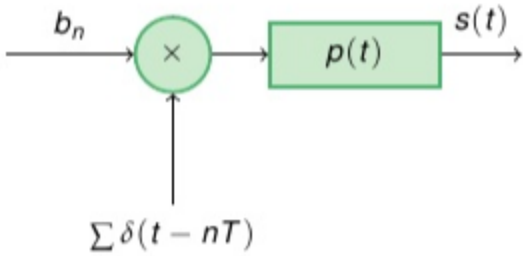
`y = pmmod(x,Fc,Fs,phasedev)` – sử dụng tín hiệu x để điều chế PM sóng mang Fc tại tần số lấy mẫu Fs với độ lệch pha phasedev.

4.4.2Điều chế tín hiệu số

Quá trình điều chế tín hiệu số trong mô phỏng thường sử dụng mô hình tương đương băng gốc được mô tả tổng quát trong hình 4-25 và có thể biểu diễn bởi

$$x(t) = \sum_{k=0}^{N-1} b_k \, p(t - kT) \tag{4.46}$$

trong đó b_k là các mức giá trị trong bảng mẫu ký tự phù hợp với các định dạng điều chế khác nhau, ví dụ: điều chế BPSK ta có $b_k \in \{1, -1\}$; điều chế OOK ta có $b_k \in \{0, 1\}$; hay điều chế PAM ta có $b_k \in \{\pm 1, \ldots, \pm(M-1)\}$. Hàm $p(t)$ xác định dạng xung đầu ra tín hiệu điều chế (ví dụ hàm xung vuông hay sinc). So sánh biểu thức (4.46) và (4.33) ta sẽ thấy sự tương tự trong mô phỏng mã đường và điều chế số.



Hình 4-25 Mô tả quá trình mô phỏng điều chế số

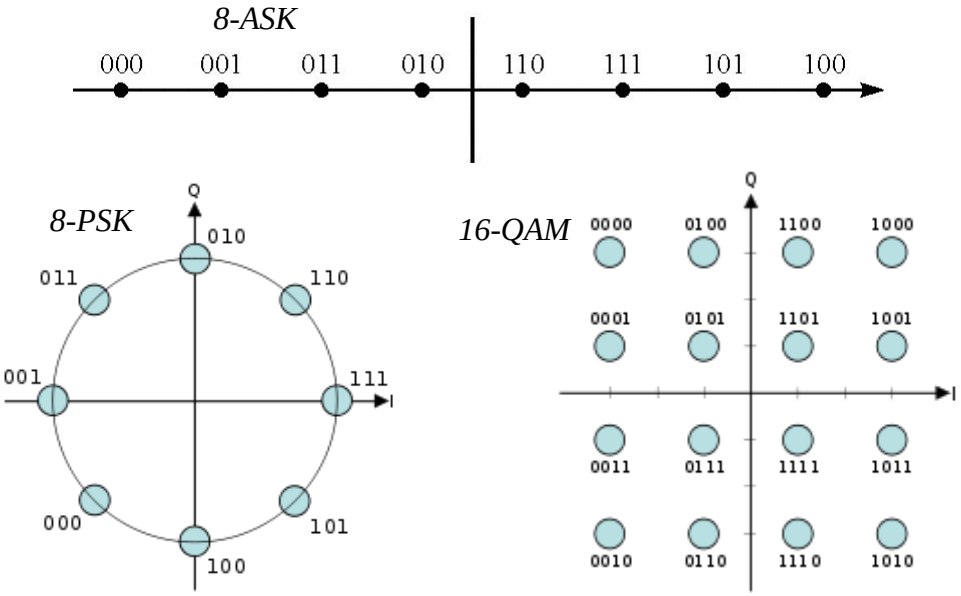
Quá trình điều chế số cũng có thể được mô tả dưới dạng cầu phương gồm hai thành phần dạng sóng khác nhau $X_1(t)$ và $X_2(t)$ với

$$X_1(t) = \sum_{k=0}^{N-1} a_k \, p_1(t - kT_1) \tag{4.47}$$

và

$$X_2(t) = \sum_{k=0}^{N-1} b_k \, p_2(t - kT_2) \tag{4.48}$$

trong đó $p_1(t)$ và $p_2(t)$ là các hàm dạng xung giống như $p(t)$; a_k và b_k là các chuỗi biến ngẫu nhiên rời rạc trong bảng mẫu tự với tốc độ ký hiệu là $1/T_1$ và $1/T_2$ tương ứng. Trong các mô phỏng hệ thống truyền thông thông thường, $T_1 = T_2$ và $p_1(t) = p_2(t)$. Để mô tả các loại điều chế số khác nhau, biểu đồ chòm sao thường được sử dụng như cho thấy trong hình 4-26 với một số kiểu điều chế hay gặp.



Hình 4-26 Một số ví dụ biểu đồ chòm sao trong điều chế số.

Một số chế độ điều chế phổ biến trong hệ thống thực tế được mô tả bởi biểu thức (4.47) và (4.48) với các tham số cụ thể được cho trong bảng 4-2. Các công thức (4.47) và (4.48) được sử dụng để xây dựng các hàm điều chế số trong mô phỏng.

Bảng 4-2 Bảng các thành phần của các chế độ điều chế số

Chế độ điều chế	$\{a_k, b_k\}$	$p_1(t), p_2(t)$
Khóa dịch biên độ (M-ASK)	$a_k = \pm nd$, $n = 1, 2, \ldots, M/2$ $b_k = 0$	$p_1(t) = 1$, $0 \le t \le T$; $p_2(t) = 0$
Khóa dịch pha (M-PSK)	$a_k + jb_k = \exp(j\phi_k)$, $\phi_k = 2\pi n/M$, $n = 0, 1, \ldots, M-1$	$p_1(t) = 1$, $0 \le t \le T$; $p_2(t) = p_1(t)$
QPSK (M-PSK, M = 4)	$\{a_k, b_k\} = \{\pm 1, \pm 1\}$ hoặc $\phi_k = 45^o, 135^o, 225^o, 315^o$	$p_1(t), p_2(t)$ giống như M-PSK
O-QPSK	$\{a_k, b_k\} = \{\pm 1, \pm 1\}$	$p_1(t) = 1$, $0 \le t \le T$; $p_2(t) = 1$, $\frac{1}{2}T \le t \le \frac{3}{2}T$
Khóa dịch tối thiểu (MSK)	$\{a_k, b_k\} = \{\pm 1, \pm 1\}$	$p_1(t) = \sin(\pi t/2T)$, $0 \le t \le T$; $p_2(t) = \sin(\pi t/2T - T/2)$, $\frac{1}{2}T \le t \le \frac{3}{2}T$
Điều chế biên độ cầu phương (M-QAM)	$\{a_k, b_k\} \in \{\pm 1, \pm 3, \ldots, \pm\sqrt{M}-1\}$	$p_1(t), p_2(t)$ giống như M-PSK

Trong thư viện `Communications Toolbox` của MATLAB có các hàm sử dụng cho điều chế tín hiệu số băng gốc bao gồm:

`y = pammod(x,M)` – điều chế bản tin số x sử dụng PAM hoặc M-ASK

`y = pskmod(x,M)` - điều chế bản tin số x sử dụng M-PSK

`y = dpskmod(x,M)` - điều chế bản tin số x sử dụng M-DPSK

`y = qammod(x,M)` - điều chế bản tin số x sử dụng M-QAM

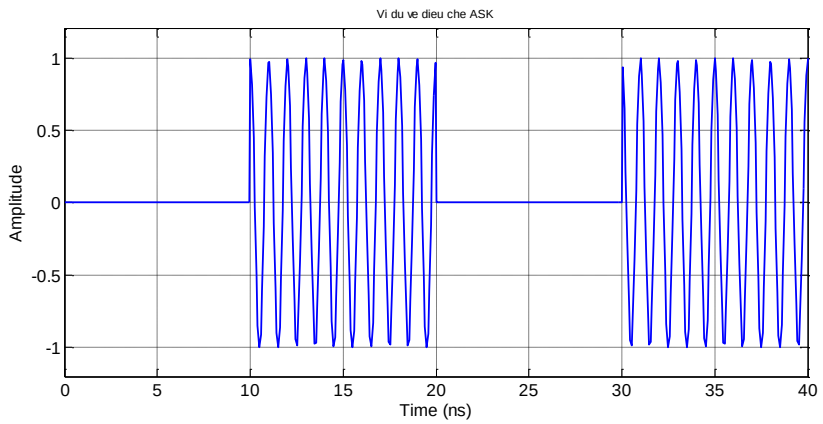
`y = fskmod(x,M,freq_sep,nsamp)` – điều chế bản tín số x sử dụng M-FSK với khoảng cách tần số freq_sep, nsamp là số mẫu trên một symbol.

Một ví dụ về điều chế 4-ASK qua đoạn mã đơn giản sau:

```
% Signal generator
dm = randint(1,1000,4);
% PAM modulation
s = pammod(dm,4);
% PAM demodulation
r = pamdemod(s,4);
```

Ngoài biểu diễn tín hiệu điều chế dưới dạng tương đương băng gốc, tín hiệu điều chế số dạng thông dải có cả sóng mang cũng có thể được thực hiện bằng cách kết hợp với các hàm điều chế tương tự. Ví dụ điều chế 2-ASK có sóng mang được cho trong đoạn mã sau:

```
% Chuỗi dữ liệu
h = [0 1 0 1];
% NRZ modulation
[t,y,code]=nrzcode(h,1e6,512); % Tạo dạng sóng sử dụng mã NRZ
% Điều chế AM
ts = t(2)-t(1);    % sampling time
fs = 1/ts;         % sampling freq.
fc = 10e6;         % carrier freq.
yd = ammod(y,fc,fs);
plot(t,yd) ;
```

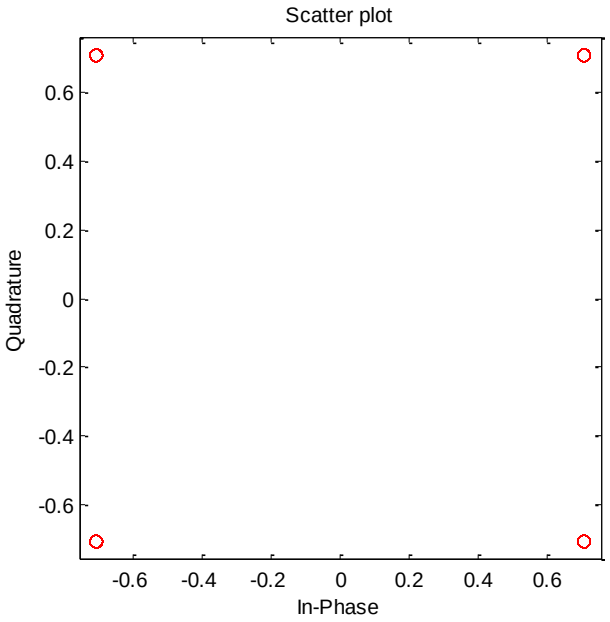


Hình 4-27 Kết quả ví dụ điều chế số 2-ASK có sóng mang.

Một ví dụ điều chế số khác sử dụng `modem object` được cho trong đoạn mã sau:

```
% Tạo bản tin số ngẫu nhiên
M = 4; % Bậc điều chế
x = randint(5000,1,M);  % Tạo bản tin số
% Sử dụng điều chế QPSK để tạo y.
h = modem.pskmod(M,pi/4);  % Thiết lập thuộc tính điều chế dùng QPSK
h.symbolorder = 'gray'; % Thiết lập thuộc tính điều chế dùng mã gray
y = modulate(h,x);  % Thực hiện điều chế
scatterplot(y);   % Vẽ biểu đồ chòm sao
```

Trong ví dụ trên biểu đồ chòm sao của tín hiệu sau khi điều chế được vẽ nhờ sử dụng hàm `scatterplot` và kết quả cho trên hình 4-28.



Hình 4-28 Biểu đồ chòm sao tín hiệu điều chế QPSK

Tuy nhiên cần lưu ý, các hàm MATLAB trên chỉ thực hiện việc chuyển đổi từ bản tin gốc thành các symbol theo kỹ thuật điều chế tương ứng. Do vậy để tạo ra tín hiệu điều chế dạng sóng điều chế đầu ra thì cần thực hiện kỹ thuật tương tự như bước 2 trong mô phỏng mã đường truyền.

4.4.3 Quá trình thu và giải điều chế

Quá trình giải điều chế là quá trình ngược với quá trình điều chế chuyển tín hiệu thông dải trở lại tín hiệu băng gốc. Hầu hết các bộ giải điều chế thuộc về một trong hai loại: giải điều chế kết hợp (coherent) và giải điều chế không kết hợp (incoherent).

Tuy nhiên để khôi phục tín hiệu ban đầu một số hoạt động xử lý tín hiệu sẽ được thực hiện để thu được bản sao giống nhất trong trường hợp tương tự hoặc ước tính tin cậy nhất luồng ký hiệu trong trường hợp số. Hoạt động xử lý thường là quá trình lọc, tiếp sau đó có thể là quá trình thu (detection) trong bộ thu số. Quá trình thu đơn giản là thực hiện lấy mẫu đầu ra bộ lọc và so sánh với một ngưỡng để quyết định tín hiệu thu được (mạch quyết định).

Giải điều chế kết hợp:

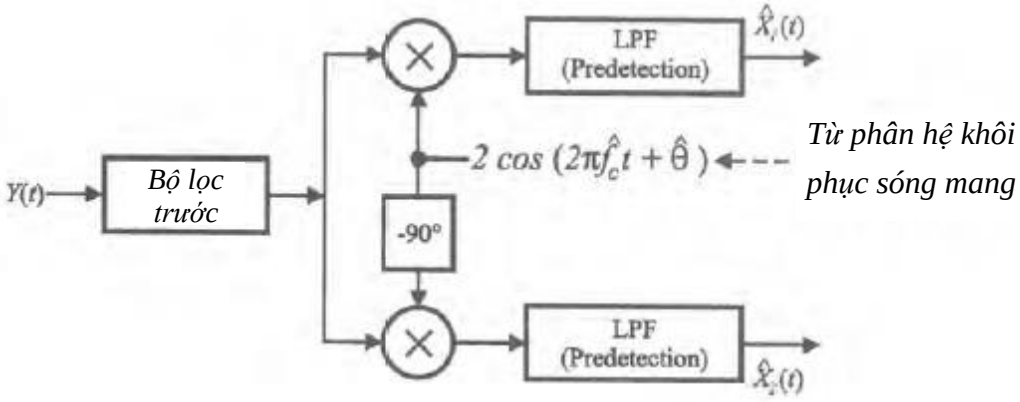
Quá trình giải điều chế này đòi hỏi một tín hiệu dao động nội trong bộ thu có dạng:

$$\hat{c}(t) = 2\cos\left(2\pi\hat{f}_c t + \hat{\theta}\right)$$

trong đó $\hat{f}_c$ và $\hat{\theta}$ là các ước tính tương ứng của tần số sóng mang và pha được sinh ra từ cơ chế khôi phục sóng mang trong bộ thu. Khi đó đối với mô tả tổng quát theo (4.41), các ước tính của $X_1(t)$ và $X_2(t)$ có thể được xác định theo sơ đồ giải điều chế như hình 4-28. Các bộ lọc trước quá trình thu thường là các bộ lọc phối hợp sẽ được đề cập đến ở phần 4.5.3. Các tín hiệu được ước tính có dạng:

$$\begin{aligned}\hat{X}_1(t) = {} & X_1(t)\cos 2\pi\left[\left(f_c - \hat{f}_c\right)t + \left(\theta - \hat{\theta}\right)\right] \\ & - X_2(t)\sin 2\pi\left[\left(f_c - \hat{f}_c\right)t + \left(\theta - \hat{\theta}\right)\right]\end{aligned} \tag{4.49a}$$

$$\begin{aligned}\hat{X}_2(t) = {} & X_2(t)\cos 2\pi\left[\left(f_c - \hat{f}_c\right)t + \left(\theta - \hat{\theta}\right)\right] \\ & + X_1(t)\sin 2\pi\left[\left(f_c - \hat{f}_c\right)t + \left(\theta - \hat{\theta}\right)\right]\end{aligned} \tag{4.49b}$$



Hình 4-28 Bộ giải điều chế cầu phương tổng quát

Nếu $\hat{f}_c = f_c$ và $\hat{\theta} = \theta$ thì khi đó $\hat{X}_1(t) = X_1(t)$ và $\hat{X}_2(t) = X_2(t)$. Cũng như bộ điều chế, quá trình mô phỏng sự không hoàn hảo của bộ giải điều chế cũng có thể dễ dàng thực hiện bằng việc biến đổi (4.49) với một vài số hạng được thêm vào.

Trong các hệ thống thực tế, tín hiệu đầu vào bộ thu là một phiên bản bị méo dạng và nhiễu của $Y(t)$ và có dạng:

$$z(t) = w(t) + n(t) \tag{4.50}$$

trong đó $w(t)$ là đường bao phức của thành phần tín hiệu bị méo dạng và $n(t)$ là đường bao phức của nhiễu. Theo mô hình đường bao phức (tương đương băng gốc), hoạt động bộ giải điều chế có thể được mô tả như:

$$\begin{aligned}\hat{X}_1(t) + j\hat{X}_2(t) = \{&w_R(t) + n_c(t) + j[w_I(t) + n_s(t)]\} \\ &\times \exp\left[j2\pi\left(f_c - \hat{f}_c\right)t + j\left(\theta - \hat{\theta}\right)\right]\end{aligned} \tag{4.51}$$

trong đó $w(t) = w_R(t) + jw_I(t)$ và $n(t) = n_c(t) + jn_s(t)$. Các giá trị mẫu của phương trình trước chính là những cái được thực hiện trong mô phỏng một bộ giải điều chế coherent. Các tín hiệu $\hat{X}_1(t)$ và $\hat{X}_2(t)$ thường được xem như là các tín hiệu băng gốc.

Nếu $X_1(t)$ và $X_2(t)$ là các tín hiệu tương tự thông thấp thì $\hat{X}_1(t)$ và $\hat{X}_2(t)$ thường chỉ cần được lọc để giảm thiểu nhiễu. Còn nếu các tín hiệu được giải điều chế $\hat{X}_1(t)$ và $\hat{X}_2(t)$ là các tín hiệu số thì chúng cần phải được xử lý để khôi phục các chuỗi ban đầu $\{a_k\}$ và $\{b_k\}$ với lỗi xảy ra ít nhất có thể. Do đó hai hoạt động cơ bản được thực hiện đối với tín hiệu giải điều chế đó là hoạt động lọc thường là lọc phối hợp và quá trình lấy mẫu tiếp sau đó để đưa vào bộ quyết định. Các mẫu tín hiệu được sử dụng để quyết định ký hiệu nào đã được gửi đi. Trong trường hợp đồng bộ hoàn toàn, giả sử các méo dạng sinh ra từ kênh có đáp ứng $h_c(t)$ và đáp ứng của bộ lọc phối hợp là $h_m(t)$, khi đó các dạng sóng đầu vào bộ lấy mẫu có thể viết thành:

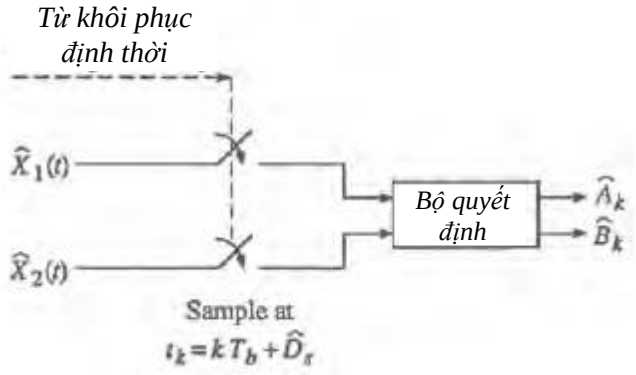
$$s_1(t) = \sum_{k=-\infty}^{\infty} a_k\, p(t - kT - D) * [h_c(t) * h_m(t)] + n_1(t) \tag{4.52a}$$

$$s_2(t) = \sum_{k=-\infty}^{\infty} b_k\, p(t - kT - D) * [h_c(t) * h_m(t)] + n_2(t) \tag{4.52b}$$

trong đó D là đại lượng trễ truyền dẫn chưa biết và cố định, dấu * mô tả phép tích chập, và $n_1(t) = n_c(t) * h_m(t)$, $n_2(t) = n_s(t) * h_m(t)$. Để khôi phục $\{a_k\}$ và $\{b_k\}$ dạng sóng (4.52) phải được lấy mẫu tại các thời điểm thích hợp. Hoạt động lấy mẫu sẽ được thực hiện tại các thời điểm:

$$t_k = kT_b + \hat{D} \tag{4.53}$$

trong đó $\hat{D}$ là một ước tính trễ hệ thống. Ước tính $\hat{D}$ được xác định qua hệ thống khôi phục định thời trong bộ thu như mô tả trong hình 4-29. Trong ngữ cảnh mô phỏng, thường không có trễ truyền dẫn và trễ hiệu dụng vì quá trình lọc có thể được ước tính khá chính xác trước. Các vấn đề về đồng bộ và quá trình lọc sẽ được trình bày trong các phần sau.



Hình 4-29 Cấu hình lấy mẫu và quyết định

Giải điều chế không kết hợp:

Quá trình giải điều chế không kết hợp bao hàm bộ giải điều chế không đòi hỏi một tín hiệu dao động nội đồng bộ để thực hiện. Xét trường hợp đơn giản khi $y(t)$ là một tín hiệu AM có dạng

$$y(t) = [1 + kx(t)]\cos(2\pi f_c t + \theta) \qquad (4.54)$$

Nếu $|kx(t)| \le 1$ thì khi đó bằng việc quan sát đường bao thực của $y(t)$ là $[1+kx(t)]$ tín hiệu điều chế $x(t)$ có thể thu được vì $|1+kx(t)| = 1+kx(t)$. Như vậy hoạt động giải điều chế này thường được gọi là kỹ thuật tách đường bao. Vì do nhiễu và méo dạng, $\tilde{y}(t)$ nói chung là dạng phức, do đó hoạt động tách đường bao đơn giản là lấy giá trị tuyệt đối của đường bao phức. Trong một số bộ thu khác như bộ thu quang, quá trình giải điều chế biên độ không kết hợp đòi hỏi lấy bình phương biên độ của đường bao phức. Phương pháp này còn được gọi là giải điều chế theo quy luật bình phương. Nếu tín hiệu đầu vào được xác định bởi (4.54) thì quá trình giải điều chế không kết hợp có dạng

$$[1+kx(t)]^2 = 1 + k^2x^2(t) + 2kx(t) \approx 1 + 2kx(t) \quad \text{khi} \quad |x(t)| << 1$$

Như vậy phương pháp này đòi hỏi một độ sâu điều chế nhỏ. Theo mô hình đường bao phức cả hai phương pháp không kết hợp trên có thể được biểu diễn như sau

$$\hat{x}(t) = \begin{cases} |\tilde{z}(t)| \\ |\tilde{z}(t)|^2 \end{cases} \quad (4.55)$$

trong đó $\tilde{z}(t)$ là đường bao phức của tín hiệu đầu vào bộ giải điều chế và $\hat{x}(t)$ là tín hiệu thực đầu ra bộ giải điều chế.

Các tín hiệu điều chế pha và điều tần có thể được giải điều chế bằng nhiều chế độ khác nhau. Chế độ cổ điển cho tín hiệu FM được gọi là thu sai biệt (discriminator). Do pha là tích phân của tần số nên một bộ giải điều chế PM là một bộ discriminator theo sau bởi một bộ lọc tích phân. Về nguyên tắc hoạt động này được thực hiện ngược với quá trình điều chế FM. Nếu $\tilde{w}(t) = r(t)e^{j\psi(t)}$ là tín hiệu đường bao phức thu được của tín hiệu điều chế thì trong mô phỏng quá trình thu discriminator để khôi phục tín hiệu được thực hiện bằng việc lấy vi phân thành phần pha của lớp vỏ phức

$$\tilde{x}(t) = \frac{d}{dt}\psi(t) \quad (4.56)$$

Chú ý trong mô phỏng việc lấy vi phân được thực hiện trong miền rời rạc. Trong MATLAB quá trình này có thể được thực hiện bằng việc sử dụng các hàm `diff` và `gradient`.

Trong thư viện `Communications Toolbox` của MATLAB có các hàm sử dụng cho quá trình giải điều chế tín hiệu tương tự có sóng mang. Cách sử dụng các hàm giải điều chế cũng tương tự như các hàm điều chế với đuôi tên hàm được thay bằng demod: `amdemod`, `fmdemod` và `pmdemod`. Đối với tín hiệu số quá trình giải điều chế băng gốc được thực hiện qua các hàm: `pamdemod`, `pskdemod`, `dpskdemod`, `qamdemod` và `fskdemod`. Ngoài ra quá trình giải điều chế số cũng có thể sử dụng `modem object` như được cho ví dụ trong đoạn mã sau:

```
%% Quá trình giải điều chế để khôi phục bản tin sử dụng modem object
% yn là tín hiệu đầu vào bộ giải điều chế.

% Thiết lập thuộc tính giải điều chế dùng QPSK
h = modem.pskdemod(M,pi/4);
% Thiết lập thuộc tính giải điều chế dùng mã gray
h.symbolorder = 'gray';
% Thực hiện giải điều chế theo thuộc tính đã thiết lập
z=demodulate(h,yn);
```

4.5 Quá trình lọc

Quá trình lọc được thực hiện tại nhiều vị trí trong hệ thống cho các mục đích khác nhau. Các ứng dụng cơ bản của quá trình lọc trong các hệ thống truyền thông bao gồm lựa chọn các tín hiệu mong muốn, giảm ảnh hưởng của nhiễu và giao thoa, biến đổi phổ tín hiệu và tạo dạng các tính chất miền thời gian (điểm cắt 0 và các dạng xung) của dạng sóng tín hiệu số để cải thiện khả năng thu.

4.5.1 Lọc tạo dạng phổ

Mật độ phổ công suất $S_y(f)$ của tín hiệu đầu ra $y(t)$ của một hệ thống tuyến tính bất biến theo thời gian (LTI) được biểu diễn bởi

$$S_y(f) = S_x(f)\left|H(f)\right|^2 \tag{4.57}$$

trong đó $S_x(f)$ là mật độ phổ công suất (PSD) của tín hiệu đầu vào $x(t)$ và $H(f)$ là hàm truyền đạt của hệ thống. Bằng việc lựa chọn cẩn thận $H(f)$, các thành phần phổ của tín hiệu đầu vào được lựa chọn có thể được làm nổi bật hoặc bị suy yếu. Hoạt động lọc lựa chọn tần số này được sử dụng phổ biến trong các hệ thống truyền thông. Nếu $S_x(f)$ và $H(f)$ được xác định trước thì việc tính toán $S_y(f)$ là rõ ràng. Nói cách khác, giả thiết rằng $S_y(f)$ là PSD mong muốn đầu ra được xác định từ $S_x(f)$ là PSD đầu vào. Viết lại (4.57) có được

$$\left|H(f)\right|^2 = \frac{S_y(f)}{S_x(f)} \tag{4.58}$$

và bài toán ở đây trở thành bài toán tổng hợp một bộ lọc thực tế $H(f)$ để thỏa mãn (4.57). Điều này có thể được thực hiện bằng việc xác định thừa số vế phải của biểu thức (4.58) và bao gồm chỉ các pole và các zeros ở một nửa trái của mặt phẳng s cho việc xác định $H(s)$. (Khai triển $s = j2\pi f$ được áp dụng trước khi thực hiện …).

Ví dụ thực hiện quá trình lọc tín hiệu bằng bộ lọc Butterworth có hàm truyền được mô tả bởi

$$\left|H(f)\right|^2 = \frac{1}{1+\left(\omega/\omega_b\right)^{2n}} \tag{4.59}$$

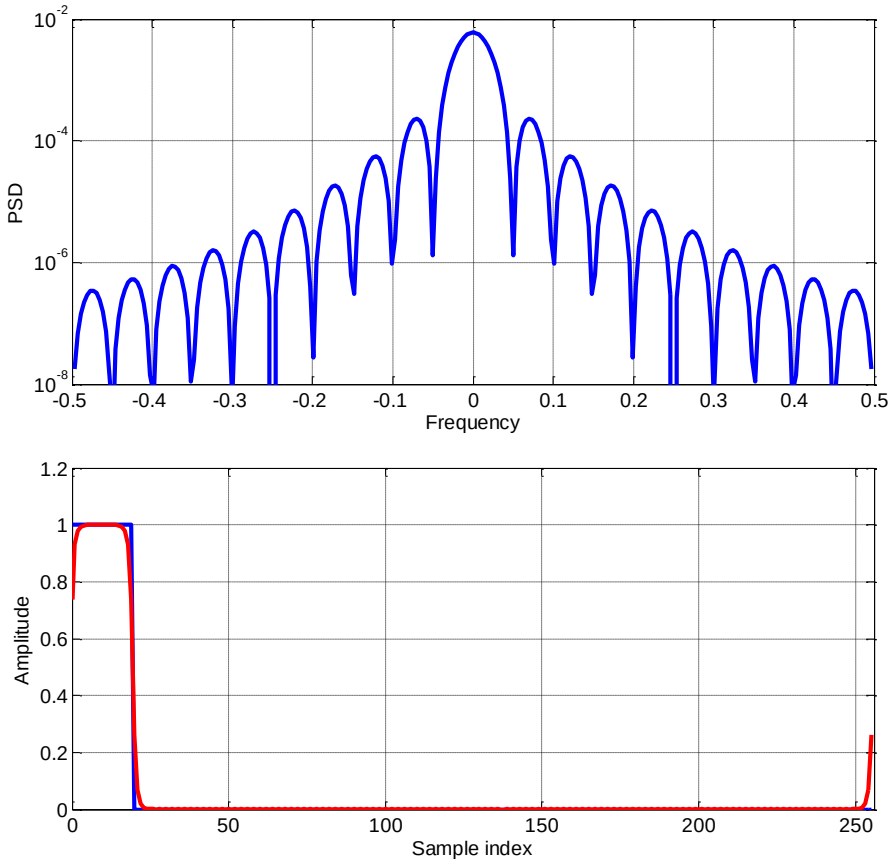
trong đó n là bậc bộ lọc và ω_b xác định độ rộng băng tần bộ lọc. Bộ lọc được viết dưới dạng hàm MATLAB cho trong đoạn mã dưới đây:

```
function y = butterwflt(x,n,B,Ts)
% Function bo loc butterworth
% B - filter bandwidth, Ts - sampling time, n - filter order
% y - filtered output
Ns = length(x);
% Frequency domain
f = [0:Ns/2-1 -Ns/2:-1]/(Ns*Ts);
Xf = fft(x);
Hf = 1./(1+(f./B).^(2*n)); % transfer func.
Yf = Xf.*Hf;
% Convert into time domain
y = ifft(Yf);
```

Ví dụ sử dụng hàm bộ lọc áp dụng cho xung vuông:

```
>> y = butterwflt(x,1,0.2,1);

>> [f,Xf]=spectrocal(t,y);

>> semilogy(f,Xf); grid;

>> figure(2);

>> plot(t,x,t,y); grid;
```

Kết quả lọc được cho trong hình 4-28.



Hình 4-28 Ví dụ quá trình lọc sử dụng bộ lọc Butterworth.

4.5.2 Lọc tạo dạng xung

Mỗi xung trong các dạng sóng được sử dụng trong các hệ thống truyền dẫn số được yêu cầu thỏa mãn hai điều kiện quan trọng: độ rộng băng tần và các điểm cắt không (zero crossings). Nếu chuỗi số có tốc độ ký hiệu là R_b thì khi đó độ rộng băng tần của dạng sóng số cần cỡ bậc R_b và nó có thể đòi hỏi có các điểm cắt không trong miền thời gian cứ mỗi chu kỳ T_b, trong đó $T_b = 1/R_b$. Nyquist đã chứng minh rằng một xung miền thời gian $p(t)$ sẽ có các điểm cắt không ở mỗi chu kỳ T_b nếu khai triển của nó $P(f)$ đáp ứng được ràng buộc sau:

$$\sum_{k=-\infty}^{\infty} P\left(f + kR_b\right) = T_b \quad \text{đối với } |f| < R_b/2 \tag{4.60}$$

Nếu $P(f)$ thỏa mãn (4.60) thì $p(t)$ có các tính chất sau:

$$\begin{aligned} p(0) &= 1 \\ p(kT_b) &= 0, \qquad k = \pm 1, \pm 2, \ldots \end{aligned} \tag{4.61}$$

Các yêu cầu điểm cắt không được đưa ra để đảm bảo giao thoa giữa các ký hiệu (ISI) bằng không và trợ giúp quá trình khôi phục tín hiệu định thời.

Một họ $P(f)$ đáp ứng được điều kiện Nyquist cho trong (4.60) là họ raised cosine với

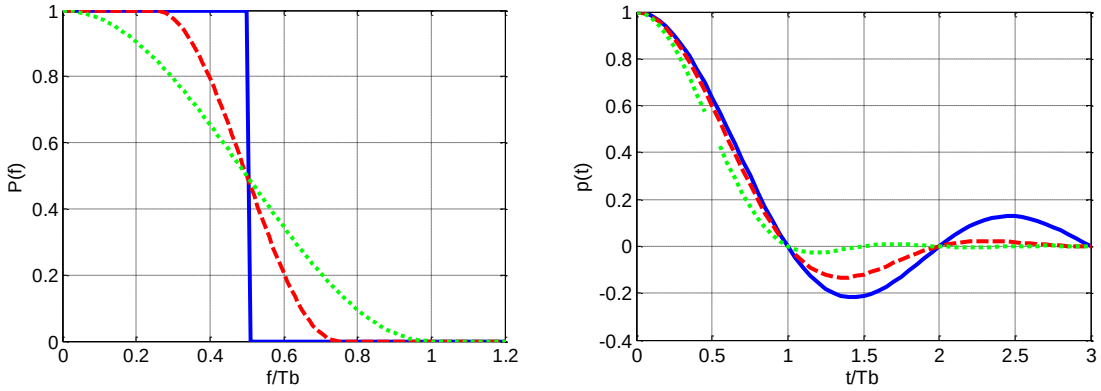
$$P(f) = \begin{cases} T_{b,} & |f| \le R_b/2 - \beta \\ T_b \cos^2 \dfrac{\pi}{4\beta}\left(|f| - \dfrac{R_b}{2} + \beta\right), & \dfrac{R_b}{2} - \beta < |f| \le \dfrac{R_b}{2} + \beta \\ 0, & |f| \le \dfrac{R_b}{2} + \beta \end{cases} \tag{4.62}$$

trong đó β là tham số băng tần trội hay còn gọi là hệ số uốn (rolloff). Chú ý rằng $P(f)$ raised cosine bị giới hạn băng tần đến $\beta+(R_b/2)$ và do đó nó có thể đáp ứng các điều kiện về băng tần chọn lọc.

Họ raised cosine có đáp ứng xung có dạng

$$p(t) = \frac{\cos 2\pi\beta t}{1 - (4\beta t)^2}\left(\frac{\sin \pi R_b t}{\pi R_b t}\right) \tag{4.63}$$

và có các điểm cắt không ở mỗi chu kỳ T_b như cho thấy trong hình 4-31.

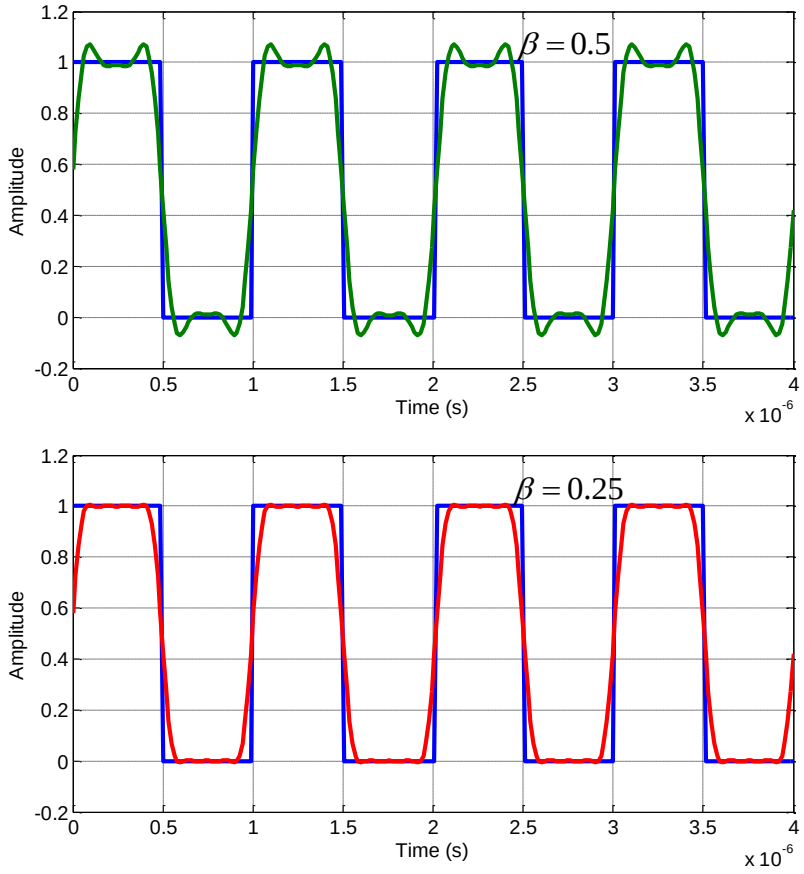


Hình 4-29 Đặc tính của họ xung raised cosine tại các giá trị β khác nhau. (a) Đáp ứng tần $P(f)$, chú ý $P(f) = P(-f)$; (b) Đáp ứng xung $p(t)$.

Xung raised cosine có thể được sinh ra bằng cách cho một xung kim qua một bộ lọc có $H(f) = P(f)$. Nếu đầu vào bộ lọc là $g(t)$ không phải là xung kim thì khi đó để sinh ra một đầu ra thuộc họ raised cosine, hàm truyền bộ lọc sẽ là

$$H(f) = \frac{P(f)}{G(f)} \tag{4.64}$$

trong đó $G(f)$ là khai triển Fourier của $g(t)$.



Hình 4-30 Ví dụ dạng xung được lọc bởi bộ lọc raised cosine tại các giá trị β khác nhau.

Trong hầu hết các ứng dụng, hoạt động lọc để tạo ra $P(f)$ được chia ra giữa hai bộ lọc, một bộ lọc tại bộ phát có hàm truyền ký hiệu là $H_T(f)$ và một bộ lọc tại bộ thu có hàm truyền ký hiệu là $H_R(f)$. Với một xung kim tại đầu vào của $H_T(f)$ và một kênh truyền lý tưởng, ta có $H_T(f)H_R(f) = P(f)$. Trong trường hợp đơn giản hệ thống có nguồn nhiễu cộng tại đầu vào bộ thu, nó có thể được chứng minh rằng sự phân chia tối ưu để đảm bảo tối ưu hóa tỉ số tín hiệu trên nhiễu tại đầu ra bộ thu là chia tách $P(f)$ bằng nhau giữa bộ phát và bộ thu, tức là

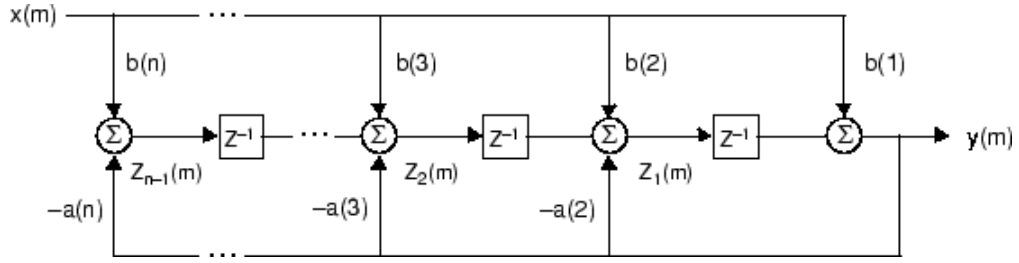
$$H_T(f) = H_R(f) = [P(f)]^{1/2} \tag{4.65}$$

Bộ lọc này còn được gọi là bộ lọc raised cosine căn bậc hai và có đáp ứng xung:

$$h_R(t) = h_T(t) = 8\beta \frac{\cos[(R_b + 2\beta)\pi t] + \sin[(R_b - 2\beta)\pi t](8\beta t)^{-1}}{(\pi R_b^{1/2})[1-(8\beta t)^2]} \tag{4.66}$$

Khi thực hiện một bộ lọc có đáp ứng như (4.66) trong miền thời gian, cần chú ý một số khó khăn trong phương pháp số có thể nảy sinh, ví dụ x = 0 trong tính sinc(x). Để tránh vấn đề này khi thực hiện trong MATLAB, giá trị x = 0 được thay thế bằng giá trị gần đúng `eps`, là một biến đặc biệt về độ chính xác tương đối trong MATLAB.

Ngoài việc tự xây dựng các hàm lọc như những ví dụ cho ở trên, trong MATLAB quá trình lọc có thể được thực hiện bằng việc sử dụng hàm `filter` trong thư viện `Signal processing toolbox` theo cấu trúc tổng quát cho trong hình 4-31.



Hình 4-31 Cấu trúc giải thuật quá trình lọc sử dụng hàm filter trong MATLAB.

Cách sử dụng hàm `filter` được thực hiện theo cú pháp cơ bản sau:

```
y = filter(b,a,x);
```

trong đó a, b là các vector có độ dài *n* chứa các hệ số của bộ lọc được đặc trưng bởi hàm truyền trong miền *z*:

$$H(z) = \frac{b(1) + b(2)z^{-1} + \ldots + b(nb+1)z^{-nb}}{1 + a(2)z^{-1} + \ldots + a(na+1)z^{-na}}$$

và quá trình lọc có thể được mô tả phương trình sai phân:

$$y(n) = b(1)*x(n) + b(2)*x(n-1) + \ldots + b(nb+1)*x(n-nb) \\ -a(2)*y(n-1) - \ldots - a(na+1)*y(n-na)$$

với $n-1$ là bậc của bộ lọc cho cả hai kiểu bộ lọc FIR và IIR, *na* là bậc bộ lọc hồi tiếp và *nb* là bậc bộ lọc nạp trước. Các hệ số lọc a, b có thể được xác định từ các hàm thiết kế bộ lọc sẵn có trong thư viện như:

- Bộ lọc Butterworth: `[b,a] = butter(n,Wn)` với n là bậc bộ lọc, Wn là tần số cắt chuẩn hóa theo nửa tốc độ lấy mẫu.
- Bộ lọc Bessel: `[b,a] = besself(n,Wo)` với n là bậc bộ lọc, Wo là tần số tại đó trễ nhóm của bộ lọc gần như không đổi.
- Các bộ lọc Chebyshev: `[b,a] = cheby1(n,R,Wp)` và `[b,a] = cheby2(n,R,Wst)` với n là bậc bộ lọc, R là độ gợn sóng đỉnh-đỉnh theo dB, và Wp và Wst là các tần số biên dải thông và tần số biên băng chặn được chuẩn hóa.
- Bộ lọc raised cosine: `y = rcosflt(x,Fd,Fs)` thực hiện lọc tín hiệu đầu vào x có tần số lấy mẫu là Fd bằng bộ lọc raised cosine cho tín hiệu đầu ra y tại tần số lấy mẫu Fs.

Lưu ý là hoạt động lọc có thể được thực hiện trong miền thời gian qua phép tích chập giữa tín hiệu đầu vào và đáp ứng xung của bộ lọc. Trong MATLAB phép tích chập có thể được thực hiện qua sử dụng hàm conv theo cú pháp cơ bản sau:

- `y = conv(x,h)` thực hiện tích chập giữa 2 vector x có độ dài m và h có độ dài n, kết quả đầu ra y có độ dài (m+n-1).

4.5.3 Các bộ lọc phối hợp

Bộ lọc phối hợp được sử dụng trong thiết kế các bộ thu tối ưu trong trường hợp bị ảnh hưởng bởi nhiễu cộng. Xét bài toán xác định có hoặc không có một xung với một dạng xung đã biết $p(t)$ có chu kỳ là T từ việc quan sát tín hiệu thu được $y(t)$

$$y(t) = \begin{cases} p(t) + n(t) \\ or \\ n(t) \end{cases}$$

Tín hiệu thu $y(t)$ được quan sát trong T giây trong khoảng $[0, T]$ và được xác định xem có hoặc không có xung $p(t)$ bằng việc xử lý $y(t)$. Quá trình xử lý $y(t)$ được thực

hiện qua một bộ lọc, do vậy việc quyết định sự có mặt xung $p(t)$ được dựa trên đầu ra của một bộ lọc tại thời điểm $t = T$. Tín hiệu đầu ra bộ lọc có dạng

$$z = \int_0^T y(\tau)h(T-\tau)d\tau \tag{4.67}$$

Nó đã được chứng minh rằng xác suất quyết định sai hoặc lỗi trung bình nhỏ nhất khi bộ lọc có hàm truyền đạt

$$H(f) = K\frac{P^*(f)\exp(j2\pi fT)}{S_N(f)} \tag{4.68}$$

trong đó K là một hằng số thực dương. Nếu $n(t)$ là nhiễu trắng thì ta có

$$H(f) = KP^*(f)\exp(j2\pi fT) \tag{4.69}$$

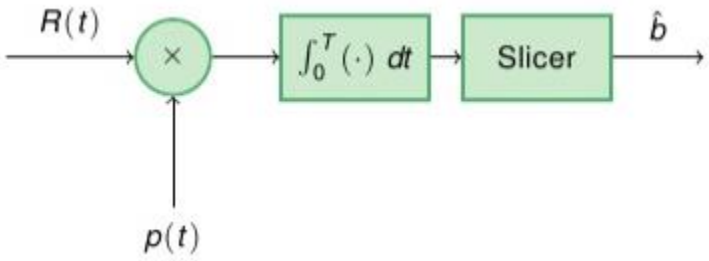
Bằng việc lấy khai triển Fourier ngược của $H(f)$, hàm đáp ứng xung của bộ lọc được xác định là

$$h(t) = p(T-t) \tag{4.70}$$

do đó

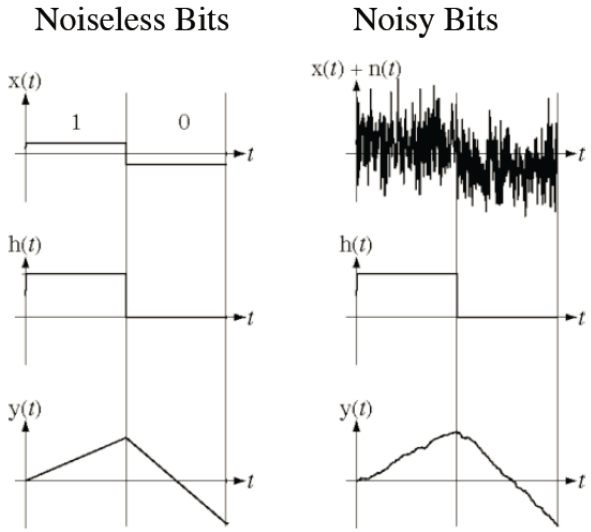
$$z = \int_0^T y(\tau)h(T-\tau)d\tau = \int_0^T y(\tau)p(\tau)d\tau \tag{4.71}$$

Các phương trình (4.70) và (4.71) cho thấy rằng đáp ứng xung của bộ lọc được phối hợp với $p(t)$ và đây là lí do bộ lọc kiểu này được gọi là bộ lọc phối hợp. Chú ý rằng khi $p(t)$ là xung vuông đơn vị như mô tả bởi (4.5) thì z chỉ là tích phân của tín hiệu đầu vào. Nếu một quyết định phải được thực hiện trong các khoảng thời gian liên tiếp thì (4.71) bao hàm rằng tích phân này bắt đầu thực hiện tại đầu mỗi khoảng chu kỳ. Như vậy, giá trị của tích phân tại điểm cuối khoảng trước phải được loại bỏ. Do vậy bộ lọc này còn được gọi là bộ lọc integrate-and-dump (I&D) như mô tả trong hình 4-31.



Hình 4-31 Mô tả thực hiện bộ lọc phối hợp trong bộ thu

Quá trình lọc phối hợp về mặt nguyên lý có thể bù méo tuyến tính trong các xung khi méo dạng đó được biết, nhưng không thể bù giao thoa giữa các ký hiệu. Để có thể bù được méo bao gồm cả méo phi tuyến và biến đổi theo thời gian thì bộ lọc thích ứng hay kỹ thuật cân bằng sẽ cần được sử dụng tại bộ thu.

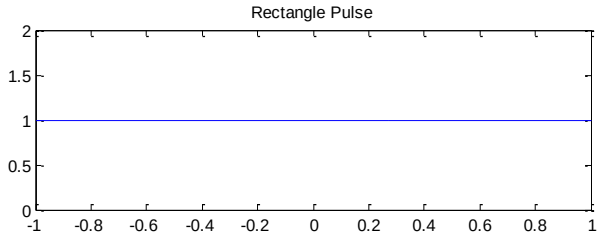


Hình 4-32 Dạng sóng tín hiệu xung vuông trong trường hợp lý tưởng và bị nhiễu tại đầu vào và đầu ra bộ lọc phối hợp.

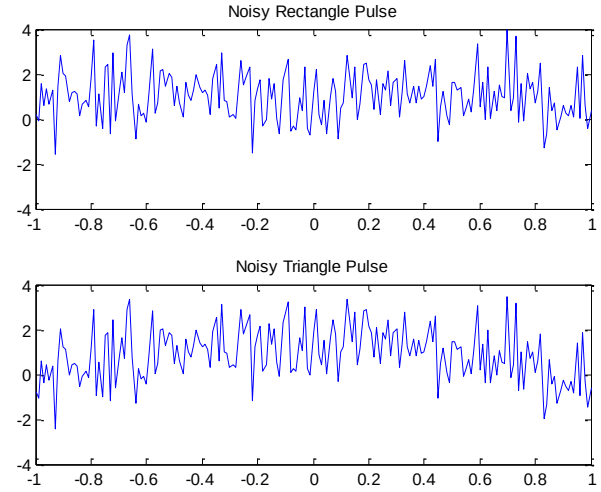
Một ví dụ về bộ lọc phối hợp với các loại xung cơ bản bao gồm xung vuông và xung tam giác trên một chu kỳ *T* được thực hiện cho trong đoạn mã dưới đây:

```
% Ví dụ về bộ lọc phối hợp cho dạng xung vuông và xung tam giác
Ts = .01;   % Chu kỳ lấy mẫu
t=-1:Ts:1;
T = length(t);
% Tạo các dạng xung
prect = ones(1,T);      % Xung vuông
ptri(1:(T-1)/2) = t(1:(T-1)/2)+1;   % Xung tam giác
ptri((T-1)/2+1) = 1;                % Xung tam giác
ptri((T+1)/2:T) = 1-t((T+1)/2:T);   % Xung tam giác
ptri = ptri * sqrt(3); % Định cỡ  energy
% Tạo nhiễu có biên độ an
an = 0.5;
n = an*randn(1,T);  % Gaussian noise
% Cộng nhiễu vào các xung
pw1 = prect + n;
pw2 = ptri + n;
% Lọc phối hợp các xung
rf1 = pw1(1);
rf2 = pw2(1);
mf2 = ptri(1)*pw2(1);
for k=2:T
  rf1(k) = rf1(k-1) + pw1(k); % Lọc xung vuông lên xung vuông
  rf2(k) = rf2(k-1) + pw2(k); % Lọc xung vuông lên xung tam giác
  % Lọc xung tam giác lên xung tam giác
  mf2(k) = mf2(k-1) + ptri(k)*pw2(k);
end
```

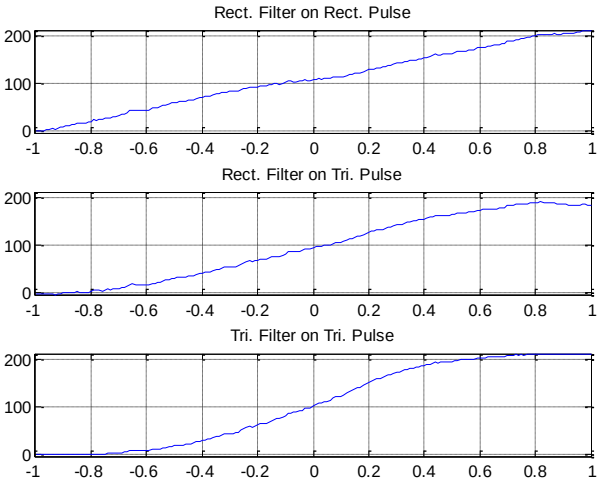
Kết quả được cho thấy trong hình 4-33 dưới đây.



(a)



(b)



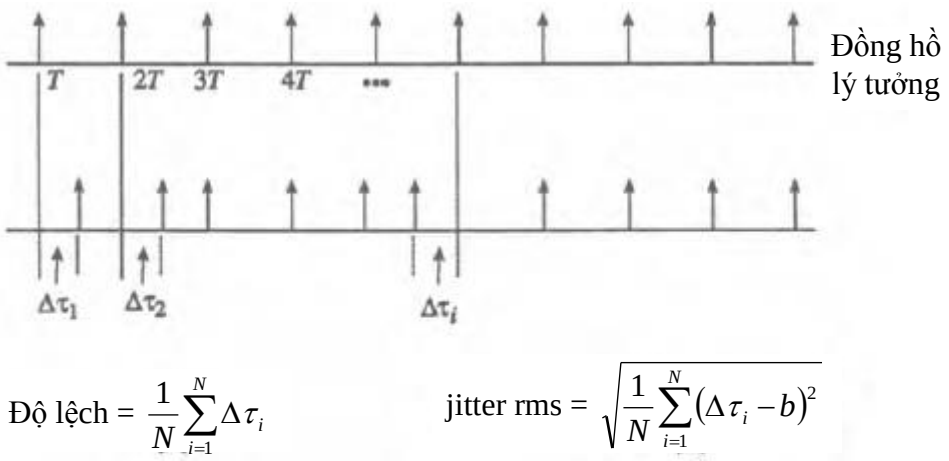
(c)

Hình 4-33 Kết quả ví dụ về bộ lọc phối hợp: (a) Dạng xung gốc ban đầu, (b) Dạng xung bị nhiễu, (c) Dạng sóng đầu ra bộ lọc phối hợp (I&D).

4.6 Quá trình đồng bộ

Trong hệ thống truyền thông, quá trình đồng bộ là một trong những chức năng chính của bộ thu bao gồm quá trình khôi phục sóng mang và khôi phục tín hiệu đồng hồ. Mặc dù có nhiều cấp đồng bộ ngoài việc đồng bộ sóng mang và định thời trong các hệ thống truyền thông lớn như đồng bộ khung, gói hay từ mã, các kiểu đồng bộ này thường được thực hiện qua việc sử dụng các chuỗi ký hiệu đặc biệt được ghép cùng với các chuỗi tin. Quá trình đồng bộ sóng mang và định thời có thể được thực hiện có hoặc không có sử dụng tín hiệu trợ giúp. Hầu hết các hệ trong đó các tín hiệu sóng mang và định thời tham khảo được tách ra từ dạng sóng tín hiệu tại bộ thu.

Các kỹ thuật đồng bộ biến đổi phụ thuộc vào chế độ điều biến, môi trường hoạt động và mức độ chính xác mong muốn. Hiệu năng của quá trình khôi phục sóng mang và định thời thường được đo bởi các tham số độ lệch thời gian và jitter (rung pha) hiệu dụng như mô tả trong hình 4-34.



Hình 4-34 Độ lệch thời gian và jitter trong một tín hiệu dạng sóng định thời.

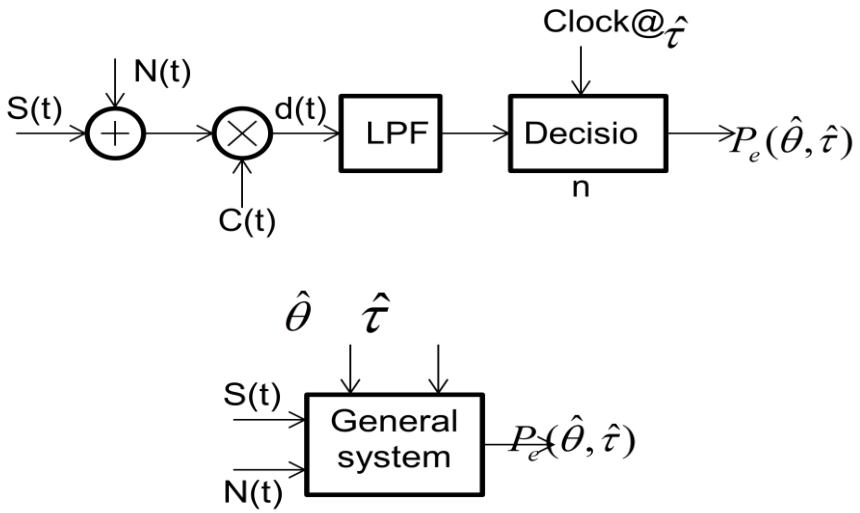
4.6.1 Quá trình đồng bộ trong mô phỏng

Mô phỏng các chức năng đồng bộ có thể được thực hiện vì hai lí do:

- Thứ nhất là để nghiên cứu đặc tính của các quá trình đồng bộ cụ thể để xem liệu các tiêu chí hiệu năng xác định đã được thỏa mãn hay chưa. Nói cách khác, phân hệ đồng bộ chính là đối tượng nghiên cứu. Trong trường hợp này, các hoạt động phần cứng hoặc phần mềm cụ thể sẽ cần được mô phỏng. Trong các nghiên cứu phân hệ như vậy, các tiêu chí hiệu năng thường liên hệ với đặc tính chuyển tiếp của cấu trúc: các đại lượng như thời gian thu nạp hay dải co kéo là hàm của các tham số đầu vào quá độ

chính là các tham số quan tâm. Vấn đề mô phỏng này liên quan đến mô phỏng mạch vòng khóa pha (PLL).

- Thứ hai là sự cần thiết để mô tả quá trình đồng bộ đơn giản là để giải thích ảnh hưởng của nó trong mô phỏng mức hệ thống mà ở đó đại lượng hiệu năng mức hệ thống (như BER) chính là đối tượng quan tâm. Trong trường hợp này, đặc tính chuyển tiếp của phân hệ đồng bộ ít quan trọng hơn hiệu năng trạng thái tĩnh được đo theo các tham số độ lệch thời gian và jitter như đã đề cập ở trên. Xác suất lỗi hệ thống liên quan đến tín hiệu giải điều chế $d(t)$ và là một hàm của các tham số về đồng bộ biểu diễn như $\Xi(d) = P_e(\hat{\theta}, \hat{\tau})$ như mô tả trong hình 4-35, trong đó Ξ là toán tử quyết định, $\hat{\theta}$ là tham số pha của sóng mang được sinh ra từ bộ dao động nội, và $\hat{\tau}$ là thời điểm lấy mẫu của xung đồng hồ.

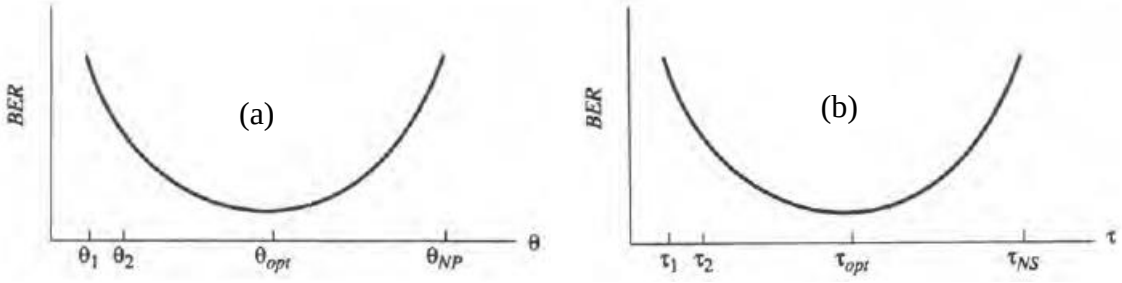


Hình 4-35 Tác động của lỗi pha và lỗi định thời lên BER. (a) Mô hình thu coherent đơn giản; (b) hệ thống tổng quát.

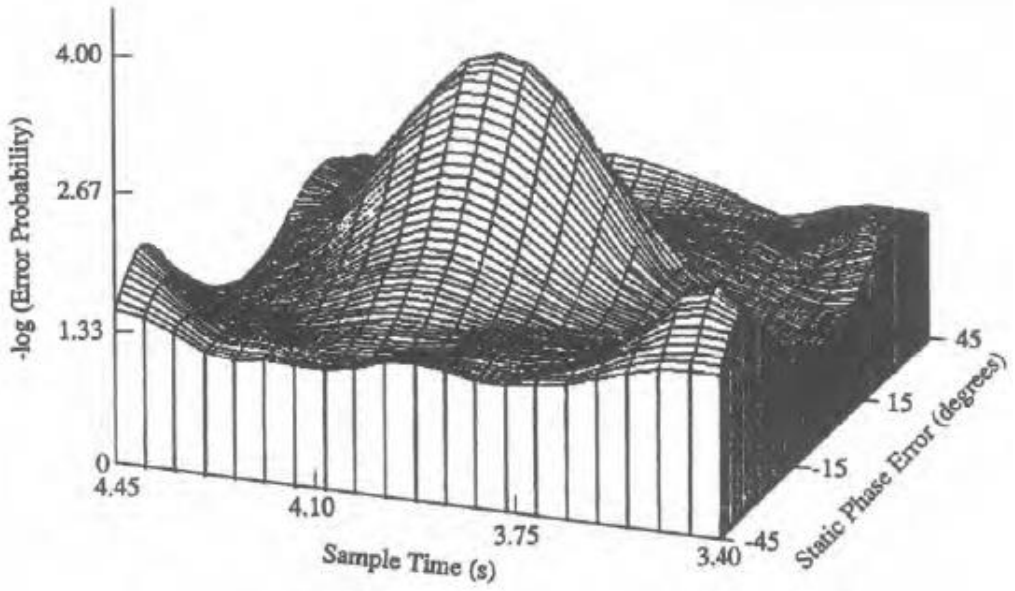
Như vậy ta có các kiểu tiếp cận mô phỏng khác nhau phụ thuộc vào cách mà $(\hat{\theta}, \hat{\tau})$ được mô tả và cách cụ thể để thu được chúng. Một số cách tiếp cận thuộc dạng "cấu trúc" xác định các hoạt động được thực hiện để rút ra $(\hat{\theta}, \hat{\tau})$. Các tiếp cận khác được dựa trên mô tả thống kê $(\hat{\theta}, \hat{\tau})$ hoặc dựa trên giải thuật trên một khuôn khổ thống kê.

Một trong những cách đơn giản đó là sử dụng cách tiếp cận đồng bộ đấu cứng. Dạng sóng $d(t)$ được lọc phối hợp và được lấy mẫu tại thời điểm $\hat{\tau}$ đưa vào bộ quyết định để xác định xác suất lỗi $P_e(\hat{\theta}, \hat{\tau})$. Trong phương pháp này, tại mỗi giá trị $\hat{\tau}$ cố định một hàm BER điều kiện $P_e\left(\hat{\theta}, \hat{\tau} \middle| \hat{\tau}\right)$ sẽ thu được bằng cách biến đổi $\hat{\theta}$

trên một tập giá trị rời rạc. Sẽ có một $\hat{\theta} = \hat{\theta}_{opt}$ mà BER là nhỏ nhất như cho thấy ở hình 4-36a và đặc tính về độ lệch hay lỗi pha tĩnh được xác định. Cũng một cách tương tự tại mỗi giá trị $\hat{\theta}$ một hàm BER điều kiện $P_e\left(\hat{\theta}, \hat{\tau} \middle| \hat{\theta}\right)$ sẽ thu được trên một tập giá trị $\hat{\tau}$ rời rạc. Hàm BER theo độ lệch định thời $\hat{\tau} - \hat{\tau}_{opt}$ sẽ được xác định như cho thấy ở hình 4-36b. Như vậy, BER là một hàm của cả hai $\hat{\theta}$ và $\hat{\tau}$, do đó một sự biểu diễn sự phụ thuộc trên một bề mặt không gian 3 chiều có thể được thực hiện như ví dụ trong hình 4-37. Trong thực tế các đại lượng $\hat{\theta}$ và $\hat{\tau}$ là các quá trình biến đổi chậm và có thể được mô tả theo một phân bố nào đó và Pe được tính trung bình trên các phân bố đó.



Hình 4-36 Sự phụ thuộc BER vào (a) độ lệch pha và (b) độ lệch định thời.



Hình 4-37 Sự phụ thuộc BER vào đồng thời cả hai tham số độ lệch pha và độ lệch định thời trong biểu diễn 3D.

Trong mô phỏng mức hệ thống, nó không cần thiết phải mô phỏng chi tiết ($\hat{\theta}$,$\hat{\tau}$) được sinh ra thế nào. Do vậy một quá trình ngẫu nhiên tương đương được sử dụng để phỏng tạo lại quá trình đồng bộ thực tế ảnh hưởng lên BER. Xét dạng sóng thu được có dạng

$$r(t) = s(t) + n(t) \tag{4.72}$$

trong đó $n(t)$ là nhiễu Gauss trắng cộng và tín hiệu

$$s(t) = \rho(t)\cos\left[\omega_c t + \phi(t) + \delta(t) + \theta\right] \tag{4.73}$$

bao gồm cả thành phần pha mong muốn $\phi(t)$ cũng như thành phần pha không mong muốn $\delta(t)$ đặc trưng cho nhiễu pha bộ dao động và méo dạng. Từ (4.72) $r(t)$ có thể được biểu diễn như

$$r(t) = \rho'(t)\cos\left[\omega_c t + \phi(t) + \alpha(t) + \delta(t) + \theta\right] \tag{4.74}$$

phản ánh sự có mặt của nhiễu. Tín hiệu tham chiếu dao động nội có thể được mô tả như

$$c(t) = 2\cos\left[\omega_c t + \hat{\delta}(t) + \hat{\theta} + \varepsilon(t)\right] \tag{4.75}$$

trong đó $\varepsilon(t)$ sinh ra từ nhiễu nhiệt và các thành phần pha khác là các ước tính của các đại lượng tương ứng trong (4.73). Dạng sóng giải điều chế thu được bằng việc nhân $r(t)$ với $c(t)$ và lấy thành phần thông thấp để có được

$$d(t) = \rho(t)\cos\left[\phi(t) + \alpha(t) + \mu(t)\right] \tag{4.76}$$

trong đó $\mu(t) = \delta(t) - \hat{\delta}(t) - \varepsilon(t) + \theta - \hat{\theta}$. Như vậy dạng sóng giải điều chế phụ thuộc vào nhiễu pha không được theo dấu $\delta(t) - \hat{\delta}(t)$ và nhiễu nhiệt được theo dấu $\varepsilon(t)$. (Lỗi pha tĩnh $\theta - \hat{\theta}$ đơn giản là một hằng số có thể được giả sử như là một tham số giống như cách tiếp cận đấu cứng). Bỏ qua lỗi pha tĩnh có thể thấy rằng $\mu(t)$ chính là quá trình ngẫu nhiên tương đương. Do vậy hệ thống có thể được mô phỏng không có nhiễu pha và $\mu(t)$ có thể đơn giản được đưa trực tiếp vào tín hiệu giải điều chế như chỉ ra ở (4.76). Nói chung $\mu(t)$ là một quá trình chậm do vậy cần chú ý các giá trị của nó cách nhau bởi T_s có mức độ tương quan nhất định do vậy cần sử dụng kỹ thuật đa tốc độ trong mô phỏng. Thêm nữa thời gian chạy mô phỏng phải đủ lớn để quan sát được hết sự tiến triển của quá trình đặc trưng. Đối với đại lượng jitter thay cho biểu thị theo độ thì sẽ được biểu diễn trong miền thời giant ham chiếu với một chu kỳ kí hiệu. Do vậy ảnh hưởng của jitter đồng hồ có thể được mô phỏng bằng cách truyền một luồng dữ liệu đồng bộ hoàn toàn và lấy mẫu đầu ra bộ lọc phối hợp bằng một tín hiệu đồng hồ hay lấy mẫu bị nhiễu. Tính nhiễu của quá trình này được mô tả bởi một quá trình ngẫu nhiên tương đương thường là quá trình ngẫu nhiên Gauss với một PSD xác định.

4.6.2 Mô phỏng mạch vòng khóa pha (PLL)

Phần tử trung tâm của nhiều cấu trúc đồng bộ chính là mạch vòng khóa pha (PLL) thường được mô tả bởi một phương trình vi phân phi tuyến. Có nhiều mô hình mô tả PLL phụ thuộc vào cấu trúc hay bậc của mạch vòng.

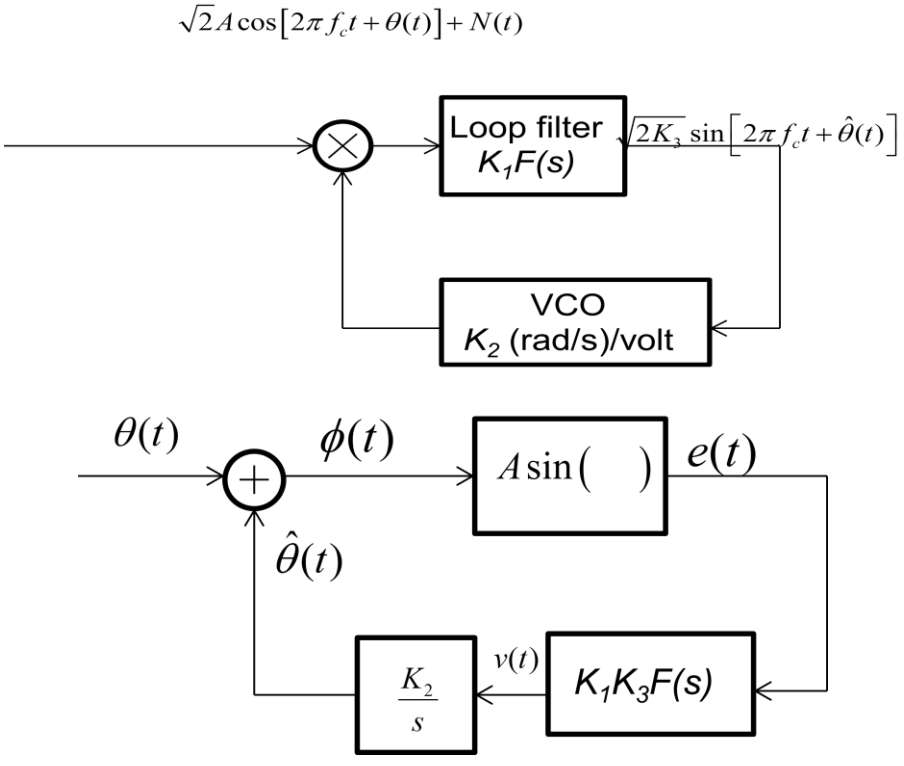
Sơ đồ khối cơ bản của PLL dạng mạch vòng bậc hai và mô hình tương đương trong miền pha của nó được cho thấy trong hình 4-38. Trong hình này, tín hiệu sóng mang đi vào sẽ được so pha với sóng mang được tạo ra từ bộ dao động. Độ lệch pha đầu ra được cho qua một bộ lọc vòng để chuyển đổi thành sự biến đổi dạng điện áp điều khiển bộ tạo dao động VCO. Hàm truyền bộ lọc vòng F(s) cho mạch vòng bậc 2 có thể được biểu diễn

$$F(s) = \frac{s\tau_2 + 1}{s\tau_1 + \alpha_1} \tag{4.77}$$

trong đó một trong hai trường hợp xác định α_1 như sau

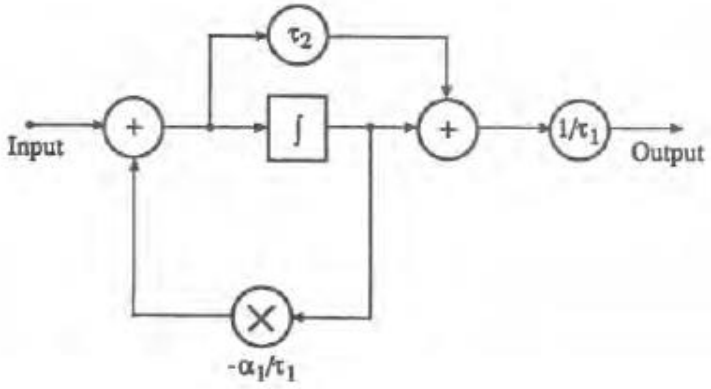
$$\alpha_1 = \begin{cases} 1, & \text{Bộ lọc thụ động} \\ 0, & \text{Bộ lọc tích cực} \end{cases} \tag{4.78}$$

và τ_1 và τ_2 là các tham số thời gian liên quan đến tần số tự nhiên và hệ số tắt dần của PLL.

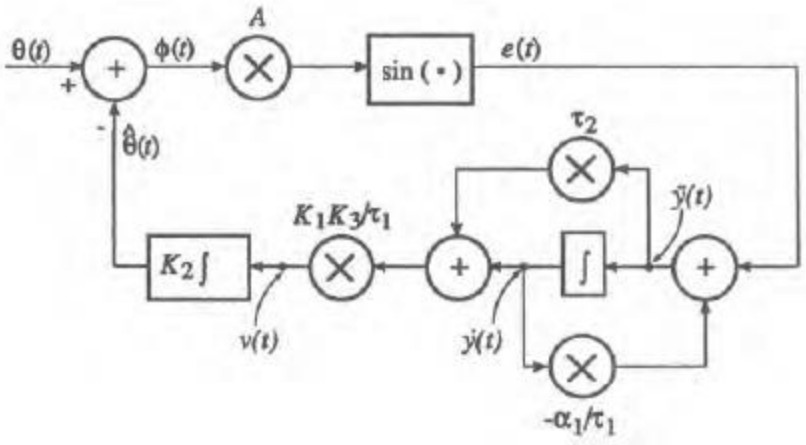


Hình 4-38 Mô tả mạch vòng khóa pha

Có hai kiểu mô hình mô phỏng PLL đó là mô hình độc lập (stand-alone) và mô hình lắp ráp (assembled). Sự khác nhau giữa hai mô hình đó là mô hình lắp ráp được xây dựng bằng việc tập hợp mô hình phương trình vi phân từ các mô hình mức thấp hơn. Tuy nhiên mô hình độc lập cho độ ổn định và độ chính xác tốt hơn tại khoảng cách mẫu xác định.



Hình 4-39 Mô tả bộ lọc vòng F(s) theo dạng kinh điển



Hình 4-40 Sơ đồ khối PLL mở rộng cho mô hình độc lập

Trong mô hình độc lập, hàm truyền bộ lọc vòng (4.77) được viết lại

$$F(s)=\frac{1}{\tau_1}\frac{\tau_2+1/s}{1+\alpha_1/\tau_1 s} \tag{4.79}$$

và được biểu diễn theo hệ thống như hình 4-39. Thay biểu diễn này vào sơ đồ khối hình 4-38 ta có được sơ đồ khối mở rộng như trong hình 4-40. Ở đây các biến mới $\ddot{y}(t)$ và $\dot{y}(t)$ với $\ddot{y}(t)=d^2y/dt^2$, $\dot{y}(t)=dy/dt$ được đưa vào để xây dựng bài toán theo dạng mô hình yêu cầu. Từ hình 4-40, các mối quan hệ giữa các tham số của mạch vòng được xác định

$$v(t) = \left(K_1 K_3 / \tau_1\right)\left[\tau_2 \ddot{y}(t) + \dot{y}(t)\right] \tag{4.80}$$

$$\hat{\theta}(t) = \left(K_1 K_2 K_3 / \tau_1\right)\left[\tau_2 \dot{y}(t) + y(t)\right] \tag{4.81}$$

$$\phi(t) = \theta(t) - \hat{\theta}(t) = \theta(t) - \left(K_1 K_2 K_3 / \tau_1\right)\left[\tau_2 \dot{y}(t) + y(t)\right] \tag{4.82}$$

$$e(t) = A\sin\left\{\theta(t) - \left(K_1 K_2 K_3 / \tau_1\right)\left[\tau_2 \dot{y}(t) + y(t)\right]\right\} \tag{4.83}$$

$$\ddot{y}(t) = e(t) - \left(\alpha_1 / \tau_1\right)\dot{y}(t) \tag{4.84}$$

Đặt

$$c_1 = \frac{-K_1 K_2 K_3 \tau_2}{\tau_1},\ c_2 = \frac{-K_1 K_2 K_3}{\tau_1},\ \ c_3 = -\frac{\alpha_1}{\tau_1} \tag{4.85}$$

và thay (4.83) vào (4.84) thu được

$$\ddot{y}(t) = A\sin\left[\theta(t) + c_1 \dot{y}(t) + c_2 y(t)\right] + c_3 \dot{y}(t) \tag{4.86}$$

Đây chính phương trình vi phân phi tuyến bậc hai theo biến y và đầu vào θ(t). Như vậy mô phỏng PLL quay trở lại bài toán giải phương trình vi phân. Các phương pháp giải phương trình vi phân ODE đã được đề cập trong Chương 3. Thêm nữa dựa trên sơ đồ khối mô tả PLL trong hình 4-40, mô hình Simulink dễ dàng được xây dựng để mô phỏng PLL.

4.7 Tổng kết chương

Nội dung chương này đã đề cập đến những vấn đề cơ bản nhất trong mô phỏng các hệ thống truyền thông trong đó phân biệt hai loại mô hình là mô hình thông dải và mô hình tương đương băng gốc. Hầu hết các mô phỏng hệ thống truyền thông hiện nay sử dụng mô hình tương đương băng gốc do giảm được tốc độ lấy mẫu để cho phép giảm tải tính toán trong mô phỏng.

Dựa trên các quá trình cơ bản xảy ra trong hệ thống truyền thông, phương pháp luận và các kỹ thuật mô phỏng các quá trình thu phát đã được trình bày cụ thể bao gồm quá trình tạo tín hiệu, quá trình mã hóa, quá trình lọc và quá trình điều chế và giải điều chế. Cùng với các ví dụ được xây dựng từ các mô hình đặc trưng cho mỗi quá trình, các hàm cơ bản sẵn có trong thư viện toolbox của MATLAB cũng được giới thiệu.

Vai trò của quá trình đồng bộ trong mô phỏng cũng được đề cập đến trong chương này. Phương pháp luận để mô phỏng ảnh hưởng của các tham số đồng bộ

đến hiệu năng hệ thống và cách thức mô phỏng đặc tính mạch vòng khóa pha đã được trình bày.

Câu hỏi/ bài tập chương 4

4-1/ Viết chương trình MATLAB tạo chuỗi bit ngẫu nhiên phân bố đều có độ dài 128 bit. Sau đó thực hiện chuyển đổi chuỗi bit này thành các giá trị thập phân trong dải từ [0,15].

- Gợi ý: Chuyển đổi vector hàng thành ma trận mx4, sau đó dùng hàm *bi2de* để chuyển đổi sang dạng thập phân.

4-2/ Xây dựng các *function* để nén và giải nén tín hiệu theo luật A có dạng cấu trúc sau:

function [y,amax] = alaw(x,A) và function x = invalaw(y,A)

4-3/ Viết chương trình mã hóa tín hiệu x = 2cos(4πt) tại tần số lấy mẫu f_s = 20 Hz sử dụng quá trình PCM theo luật A với 8 mức lượng tử. Hãy xác đinh từ mã đầu ra của 5 mẫu đầu tiên. Vẽ biểu diễn tín hiệu gốc ban đầu, tín hiệu được lấy mẫu và tín hiệu được lượng tử hóa trên cùng một hình. (Sử dụng các function đã xây dựng của bài tập trên)

4-4/ Xây dựng function để chuyển đổi chuỗi bit dữ liệu đầu vào thành mã đường RZ đơn cực.

Sử dụng function này để mã hóa RZ và biểu diễn dạng sóng của chuỗi bit [0 1 1 0 1 0 1 0] tại tốc độ 1 Mbit/s.

4-5/ Viết function để tạo chuỗi xung tam giác đầu ra. Biết trong một chu kỳ dạng xung được biểu diễn bởi hàm sau:

$$p(t)=\begin{cases}1-\left|(t-T_w)/T_w\right|, & 0\le t\le T_p \\ 0, & t\ khác\end{cases}$$

Với $T_w = T_p/2$
T_p – chu kì xung

4-6/ Cho tín hiệu tương tự được mô tả bởi công thức sau:

$$s(t)=2\cos(20\pi t+\pi/4)+\cos(30\pi t)$$

Viết chương trình thực hiện điều chế biên độ tín hiệu bằng sóng mang f_c = 300 Hz. Vẽ dạng sóng tín hiệu bản tin ban đầu và tín hiệu được điều chế.

Giải điều chế tín hiệu trên bằng kỹ thuật phù hợp và vẽ dạng sóng tín hiệu sau khi được giải điều chế.

4-7/ Tạo chuỗi ký tự ngẫu nhiên có độ dài 100 ký tự để thực hiện điều biến BPSK. Hãy viết chương trình biểu diễn dạng sóng đường bao phức của tín hiệu điều biến BPSK tại tốc độ 10 Mb/s bởi các xung raised cosine được mô tả bởi hàm sau:

$$p(t) = \left(1 - \cos\left(\frac{2\pi t}{T}\right)\right) \qquad 0 \le t \le T$$

Vẽ dạng phổ của tín hiệu được điều biến.

Chương 5 Mô phỏng kênh thông tin

Kênh thông tin là thành phần quan trọng nhất trong hệ thống truyền thông, xác định đặc tính truyền dẫn của hệ thống. Chương này sẽ đề cập đến phương pháp luận cơ bản trong việc mô phỏng kênh thông tin.

5.1 Giới thiệu chung

Các hệ thống thông tin hiện đại hoạt động trên một dải rộng các kênh thông tin bao gồm đôi dây xoắn, cáp đồng trục, sợi quang và các kênh vô tuyến. Tất cả các kênh thực tế đều gây ra sự méo dạng, nhiễu và giao thoa. Các yếu tố này ảnh hưởng đến các quá trình điều chế, mã hóa và một số chức năng xử lý khác như quá trình cân bằng để loại bỏ sự suy giảm chất lượng gây ra bởi kênh thông tin nhằm tạo ra một hệ thống thỏa mãn các mục tiêu về dung lượng và chất lượng dịch vụ trong điều kiện ràng buộc về công suất, băng thông, độ phức tạp và chi phí.

Trong mô phỏng hệ thống truyền thông, mô hình kênh thông tin mô tả sự suy giảm tín hiệu phát trải qua trên đường truyền tới bộ thu. Nói cách khác mô hình kênh thông tin mô tả mối quan hệ đầu vào – đầu ra của kênh dưới dạng toán học hay giải thuật. Mô hình này có thể thu được từ các phép đo kiểm, hoặc dựa trên lý thuyết truyền sóng trong môi trường vật lý. Các mô hình dựa trên đo kiểm đưa tới một mô tả thực nghiệm của kênh thông tin trong miền thời gian hoặc tần số và thường ở dạng các mô tả thống kê của các biến hay quá trình ngẫu nhiên. Các tham số của hàm phân bố và mật độ phổ công suất luôn được ước tính từ dữ liệu đo kiểm. Trong khi các mô hình dựa trên đo kiểm cho thấy độ tin cậy cao và thường là các mô hình hữu dụng nhất cho thiết kế, thì các mô hình thực nghiệm thường chứng minh n và khó để tổng quát hóa trừ phi một lượng lớn các phép đo được thực hiện trên các môi trường thích hợp. Ví dụ, nó rất khó để sử dụng các phép đo thu được ở khu vực đô thị này để đưa ra một mô hình cho một khu vực đô thị khác trừ phi một lượng lớn dữ liệu được thu thập trên một dải rộng các khu vực đô thị khác nhau và sẵn có lý thuyết liên quan cần thiết để hiệu chỉnh quá trình ngoại suy mô hình cho khu vực mới.

Việc phát triển các mô hình toán học cho quá trình lan truyền tín hiệu trên một môi trường truyền dẫn đòi hỏi một sự hiểu biết tốt các hiện tượng vật lý cơ bản. Ví dụ, để phát triển một mô hình cho kênh vô tuyến tầng điện ly, ta phải hiểu cơ sở vật lý của quá trình truyền sóng vô tuyến. Tương tự, một sự hiểu biết cơ bản về quang học là cần thiết để phát triển các mô hình cho các sợi quang đơn mode và đa mode. Các kỹ sư thông tin dựa vào các chuyên gia trong vật lý để cung cấp các mô hình cơ bản cho các kiểu kênh thông tin khác nhau.

Một trong những thách thức trong mô hình hóa kênh thông tinh là sự chuyển đổi một mô hình truyền sóng chi tiết thành một dạng phù hợp cho việc mô phỏng. Các mô hình toán học, từ quan điểm vật lý, có thể thường rất chi tiết và không phù hợp cho mô phỏng. Ví dụ, mô hình toán học cho một kênh vô tuyến có thể ở dạng các phương trình Maxwell. Trong khi nó rất chính xác, thì mô hình này phải được đơn giản hóa và được chuyển đổi thành một dạng phù hợp như hàm truyền đạt hoặc đáp ứng xung trước khi sử dụng nó cho mô phỏng. Rất may, quá trình này dễ dàng hơn việc thu được các mô hình vật lý cơ bản và xác định các tham số của các mô hình như vậy.

Các kênh thông tin vật lý như đôi dây xoắn, ống dẫn sóng, không gian tự do và sợi quang thường có đặc tính tuyến tính. Một số kênh như kênh vô tuyến di động, trong khi là tuyến tính có thể cư xử theo một kiểu biến đổi theo thời gian một cách ngẫu nhiên. Mô hình mô phỏng của các kênh này nằm trong hai loại chính sau:

- Các mô hình hàm truyền đạt đối với các kênh bất biến theo thời gian: Trong các mô hình này, kênh thông tin được giả sử có bản chất tĩnh (tức kênh có đáp ứng xung bất biến theo thời gian) cung cấp một đáp ứng tần xác định vì các trễ cố định trong kênh thông tin. Hàm truyền của kênh bất biến theo thời gian được cho là phẳng nếu nguồn tin có độ rộng băng tần mà trên đó kênh thông tin có đáp ứng khuyếch đại không đổi. Kênh được cho là có tính chọn lọc tần số nếu nguồn tin có độ rộng băng tần mà trên đó kênh thông tin có sự thay đổi đáng kể hệ số khuyếch đại.
- Các mô hình đường trễ nhánh (TDL) cho các kênh biến đổi theo thời gian. Đối với các mô hình kênh này, kênh thông tin được giả sử để biến đổi theo thời gian. Nếu kênh thay đổi trong khoảng thời gian quan tâm nhỏ nhất đối với một tín hiệu, thì kênh được cho là kênh fading nhanh. Nếu kênh duy trì đặc tính tĩnh trên một số lượng lớn các symbol liên tiếp

của nguồn tin, thì kênh được cho là kênh fading chậm và kênh có thể được xem giống như mô hình trên trong một khoảng thời gian xác định.

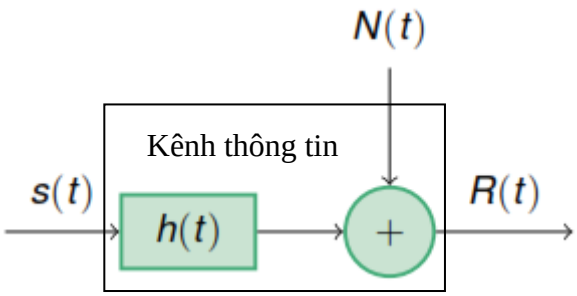
Các mô hình hàm truyền đạt có thể được mô phỏng hoặc trong miền thời gian hoặc miền tần số sử dụng các bộ lọc đáp ứng xung hữu hạn (FIR) hoặc đáp ứng xung vô hạn (IIR). Các mô hình thực nghiệm ở dạng đáp ứng xung hoặc đáp ứng tần đo được hoặc tổng hợp được thường được mô phỏng sử dụng kỹ thuật FIR. Các biểu thức giải tích cho hàm truyền đạt dễ dàng hơn để mô phỏng khi dùng kỹ thuật IIR.

Trong mô phỏng hệ thống truyền thông, các mô hình kênh dạng sóng được sử dụng để đặc trưng cho các tương tác vật lý giữa một dạng sóng được phát và kênh thông tin. Các mô hình kênh dạng sóng được lấy mẫu tại một tần số lấy mẫu thích hợp. Các mẫu được xử lý thông qua mô hình mô phỏng. Một kỹ thuật khác hiệu quả hơn trong một số ứng dụng đặc trưng kênh thông tin bởi một số trạng thái hữu hạn. Khi thời giant hay đổi, trạng thái kênh thay đổi theo tương ứng bởi một tập xác suất chuyển tiếp. Kênh thông tin khi đó có thể được định nghĩa bởi một chuỗi Markov. Mô hình kênh sinh ra thường có dạng một mô hình Markov ẩn (HMM) thuộc về các mô hình kênh rời rạc được sử dụng trong mô phỏng với lượng tính toán nhỏ nhất.

Đối với mô hình kênh dạng sóng, một mô hình tổng quát có thể được mô tả trong hình 5-1 trong đó mô tả hai ảnh hưởng cơ bản của kênh truyền đó là:

- Méo dạng sóng tín hiệu do giới hạn băng tần của kênh được đặc trưng bởi đáp ứng xung của kênh $h(t)$.
- Nhiễu cộng được đặc trưng bởi $n(t)$.

Tùy thuộc vào mô hình mô phỏng như đã đề cập ở chương 4 mà các hàm đáp ứng xung và nhiễu cộng cũng có giá trị phức hoặc giá trị thực. Việc mô phỏng các mô hình kênh có thể được thực hiện trong miền thời gian và miền tần số tương ứng.



Hình 5-1 Mô hình kênh tổng quát trong mô phỏng hệ thống truyền thông

5.2 Mô hình kênh AWGN

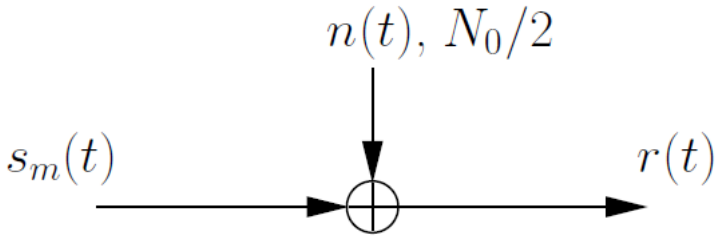
Nhiễu Gauss trắng cộng (AWGN) là một mô hình nhiễu cơ bản trong hệ thống thông tin để mô tả các quá trình ngẫu nhiên xảy ra trong kênh truyền. Các đặc tính cơ bản của nhiễu AWGN bao gồm:

- Nhiễu có tính cộng: nghĩa là nhiễu của kênh truyền được cộng thêm vào tín hiệu được phát đi, ở đây nhiễu có tính độc lập thống kê với tín hiệu.
- Nhiễu là trắng: tức mật độ phổ công suất nhiễu là phẳng, do vậy tự tương quan của nhiễu trong miền thời gian là bằng không cho bất kỳ độ lệch thời gian khác không nào.
- Nhiễu có phân bố Gauss hoặc phân bố chuẩn.

Do hầu hết các nhiễu sinh ra trong hệ thống có các đặc tính này nên mô hình kênh AWGN được coi là mô hình kênh cơ bản nhất khi mô tả ảnh hưởng của nhiễu trong hệ thống. Mối quan hệ giữa đầu vào và đầu ra của kênh AWGN có thể được biểu diễn đơn giản như sau:

$$r(t) = s(t) + n(t) \tag{5.1}$$

trong đó $n(t)$ là thành phần nhiễu trắng có phân bố Gauss với trung bình bằng 0 được cộng thêm vào tín hiệu đầu vào kênh truyền $s(t)$ như mô tả trong hình 5-2.



Hình 5-2 Mô hình kênh AWGN

Trong mô phỏng hệ thống thông tin, tùy thuộc vào loại mô hình mô phỏng hay tín hiệu đầu vào mà thành phần nhiễu $n(t)$ cũng có tính chất phù hợp tương ứng. Đối với trường hợp mô hình dải thông, tín hiệu đầu vào $s(t)$ là tín hiệu thực nên thành phần nhiễu AWGN $n(t)$ được tạo ra cũng là tín hiệu thực và có hàm tự tương quan tương ứng là:

$$\rho_n(\tau) = E\left[n(t).n^*(t+\tau)\right] = N_0\delta(\tau) \tag{5.2}$$

trong đó $E[]$ là phép tính kỳ vọng trong thông kê. Do đó mật độ phổ công suất của nhiễu $n(t)$ có thể thu được qua khai triển Fourier từ (5.1) và có được:

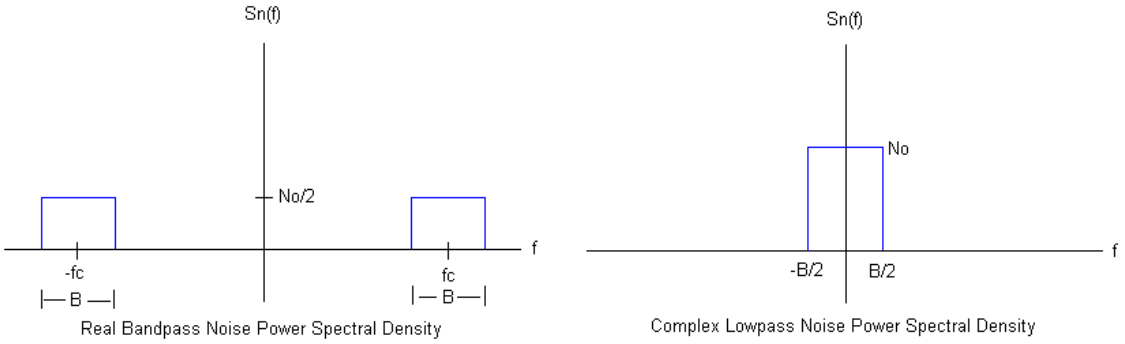
$$S_n(f) = N_0 \tag{5.3}$$

cho tất cả các thành phần tần số trong băng tần nhiễu.

Đối với trường hợp mô phỏng tương đương băng gốc, tín hiệu đầu vào $s(t)$ là tín hiệu phức nên thành phần nhiễu AWGN $n(t)$ được tạo ra cũng là tín hiệu phức gồm 2 thành phần thực và ảo (hay thành phần đồng pha và vuông pha) và mỗi thành phần có hàm tự tương quan là $N_0\delta(\tau)/2$. Các thành phần nhiễu đồng pha và vuông pha đều có phân bố Gauss và là các quá trình độc lập nhau. Như vậy đại lượng mật độ phổ công suất N_0 xác định mức công suất nhiễu AWGN hay phương sai của hàm phân bố nhiễu Gauss. Đây cũng chính là tham số đặc trưng cho kênh AWGN và xác định tỉ số tín hiệu trên nhiễu (SNR) của kênh truyền:

$$SNR = \frac{P_s}{N_0} \quad \text{hay} \quad SNR_{dB} = 10\log_{10}\left(\frac{P_s}{N_0}\right) \tag{5.4}$$

trong đó P_s là mức công suất tín hiệu đầu vào kênh AWGN. Đối với tín hiệu đầu vào là tín hiệu có tính ngẫu nhiên thì công suất tín hiệu đầu vào có thể được ước tính qua phép tính phương sai của tín hiệu. Tham số SNR cũng được sử dụng đặc trưng cho kênh AWGN thay cho mức công suất nhiễu. Khi đó, mức công suất hay phương sai nhiễu sẽ được rút ra từ giá trị SNR được thiết lập cho kênh và mức công suất tín hiệu đầu vào.



Hình 5-3 Mật độ phổ công suất kênh AWGN trong hai mô hình thông dải và mô hình tương đương băng gốc.

Kênh AWGN có thể được mô phỏng trong MATLAB bằng việc sử dụng hàm `randn` để tạo ra các mẫu nhiễu theo phân bố Gauss cộng vào tín hiệu đầu vào. Một đoạn mã MATLAB tạo ra N mẫu nhiễu Gauss phức độc lập nhau có phương sai là `VarN` được cho dưới đây:

```
% Tạo nhiễu Gauss phức có phương sai là VarN.

Noise = sqrt(VarN/2) * (randn(1,N) + j * randn(1,N));
```

Như vậy một hàm MATLAB mô phỏng kênh AWGN cho cả tín hiệu thực và tín hiệu phức có thể được xây dựng như ví dụ cho dưới đây:

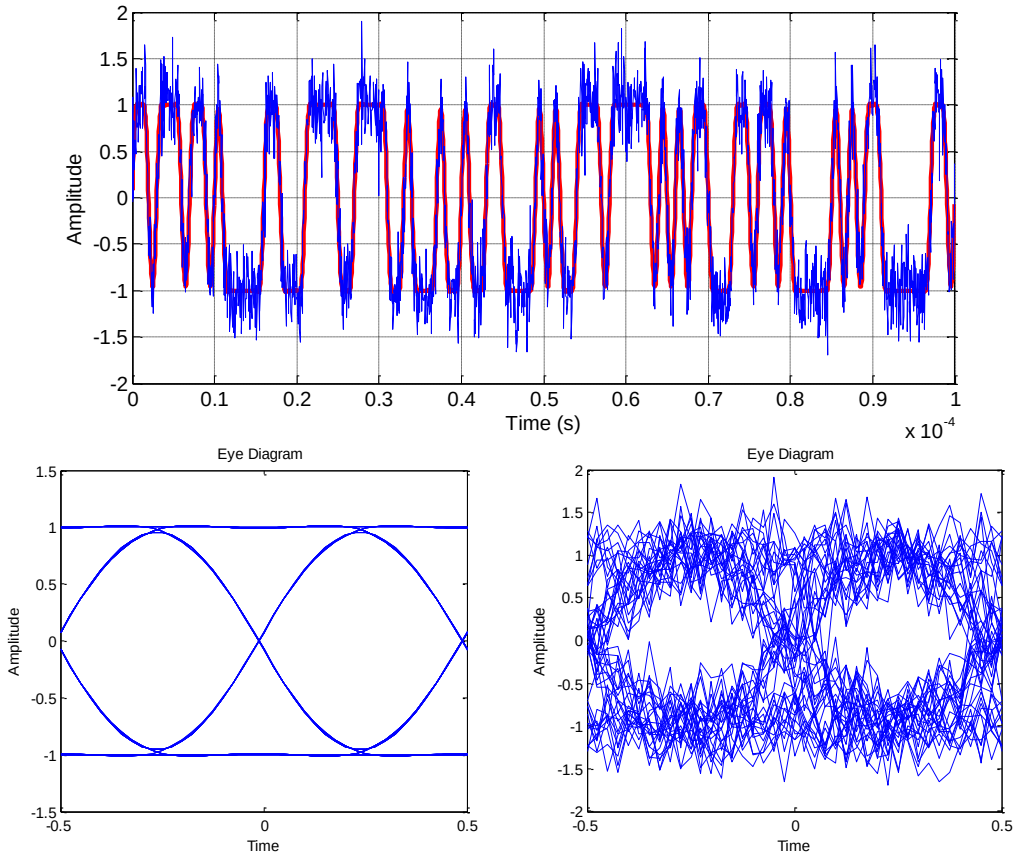
```
% Ví dụ xây dựng hàm kênh AWGN
function yNoise = addnoise(yClean,VarN)
% Hàm này thực hiện cộng nhiễu Gauss vào tín hiệu đầu vào
% yClean - Tín hiệu đầu vào
% VarN - Phương sai nhiễu
% yNoise - Tín hiệu đầu ra

if (isreal(yClean))
    yNoise = yClean + sqrt(VarN)*randn(size(yClean));
else
    yNoise = yClean + sqrt(VarN/2) ...
            *(randn(size(yClean))+j*randn(size(yClean)));
end
```

Đoạn mã MATLAB sau cho một ví dụ về sử dụng hàm addnoise đối với một tín hiệu NRZ đầu vào giả sử là tín hiệu thực. Hình 5-4 cho kết quả ảnh hưởng của nhiễu AWGN lên dạng sóng của tín hiệu tại tỉ số SNR được thiết lập tại 10 dB. Một ví dụ khác đối với tín hiệu phức sử dụng điều chế QPSK khi truyền qua kênh AWGN được cho trong hình 5-5. Ảnh hưởng của nhiễu AWGN được thấy qua biểu đồ chòm sao của tín hiệu QPSK tại 2 giá trị SNR khác nhau là 10 và 20 dB.

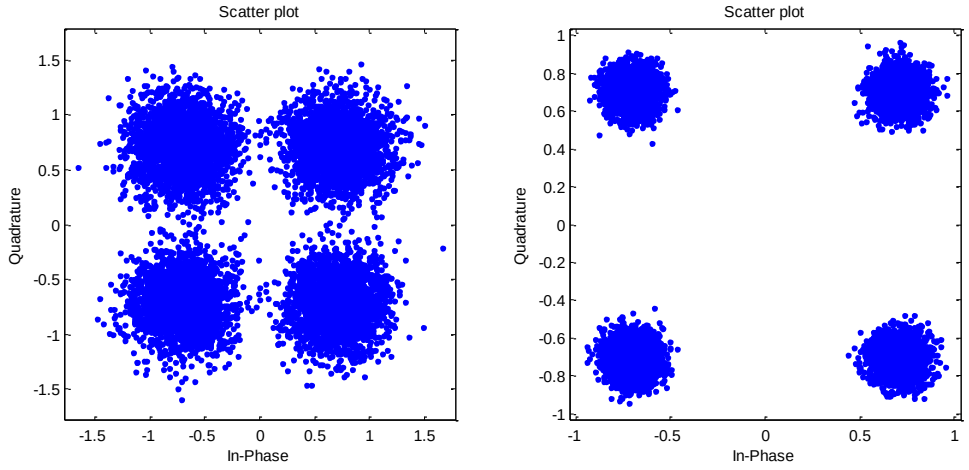
```
% Vi du ve kenh AWGN
d = randint(1,100); % Tạo chuỗi bản tin
[t,x] = nrzcode(d,1e6,2000,'pol');  % Mã hóa dạng sóng NRZ
Ts = t(2)-t(1);  % Thời gian lấy mẫu
xp = raisedcosflt(x,1.5e6,Ts,0.5);  % Tạo dạng xung
% Truyền dẫn qua kênh AWGN
SNRdB = 10;  % dB
SNR = 10^(SNRdB/10);
varn = var(xp)/SNR;  % Xác định phương sai nhiễu
xnoise = addnoise(xp,varn);
% Hiển thị kết quả
plot(t,xp,'rx',t,xnoise);grid;
xlabel('Time (s)');
ylabel('Amplitude');
eyediagram(xp,40);      % Biểu diễn biểu đồ mắt tín hiệu đầu vào
eyediagram(xnoise,40);  % Biểu diễn biểu đồ mắt tín hiệu đầu ra
```

Trong ví dụ trên hàm `eyediagram` được sử dụng để vẽ biểu đồ mắt của một tín hiệu dạng sóng. Cú pháp cơ bản sử dụng hàm `eyediagram` như sau:

- `eyediagram(x,n);` vẽ biểu đồ mắt của tín hiệu x với cửa sổ biểu diễn gồm n mẫu. n phải là một số nguyên lớn hơn 1. (Thường n là số nguyên lần số mẫu/1 chu kỳ symbol).



Hình 5-4 Dạng sóng tín hiệu NRZ trước và sau khi truyền qua kênh AWGN và biểu đồ mẫu mắt tương ứng.



Hình 5-5 Biểu đồ chòm sao của tín hiệu QPSK khi truyền qua kênh AWGN tại mức SNR bằng 10 dB và 20 dB tương ứng.

Trong thư viện `Communications Toolbox` của MATLAB có các hàm sử dụng để mô phỏng kênh AWGN đó là `awgn` và `wgn`:

- `y = awgn(x,snr) % cộng nhiễu Gauss trắng vào vector tín hiệu x`

5.3 Các mô hình kênh thông tin cụ thể

Có nhiều loại kênh thông tin khác nhau và thường phân ra thành hai loại chính là kênh hữu tuyến và kênh vô tuyến.

5.3.1 Kênh hữu tuyến và ống dẫn sóng

Các hệ thống thông tin sử dụng các loại môi trường dẫn khác nhau như đôi dây xoắn, cáp đồng trục. Các kênh thông tin này có thể được đặc trưng phù hợp bởi mô hình mạch RLC, và các đặc tính truyền tín hiệu đầu vào – đầu ra có thể được mô hình hóa bởi một hàm truyền đạt. Các nhà sản xuất cáp thường cung cấp các đặc tính trở kháng của mô hình đường truyền dẫn cho cáp dẫn và nó dễ dàng để thu được các mô hình hàm truyền đạt từ dữ liệu này. Hàm truyền đạt sau đó được sử dụng như một mô hình mô phỏng. Nó cũng dễ dàng đo được đáp ứng tần của các độ dài cáp khác nhau và thu được mô hình hàm truyền đạt dựa trên các phép đo trên. Trong một mạng cáp lớn nó có thể cần thiết để xác định kênh bằng việc sử dụng một số biến ngẫu nhiên đặc trưng cho các tham số của một hàm truyền hoặc đáp ứng xung cần tìm. Kênh thông tin, trong trường hợp này, có thể được xem như bất biến theo thời gian và do đó không cần đến mô hình biến đổi theo thời gian. Như vậy việc mô phỏng kênh hữu tuyến lúc này tương tự như quá trình lọc như đã đề cập trong Chương 4. Một hàm truyền đạt cáp đồng đơn giản dựa trên mô hình mạch RLC có dạng

$$H(\omega) = \frac{1}{1 - \omega^2 LC + j\omega RC} \tag{5.5}$$

trong đó R, L, C là điện trở, độ cảm và điện dung của cáp tương ứng, các tham số này phụ thuộc vào đặc điểm cụ thể về kích thước và vật liệu của cáp.

Các ống dẫn sóng và sợi quang có thể cũng được bao gồm trong loại môi trường truyền dẫn có dẫn sóng (hữu tuyến). Trong khi mode truyền dẫn có thể thay đổi, các kênh thông tin trong loại này có thể được mô hình hóa như các hệ thống tuyến tính bất biến theo thời gian được mô tả bởi hàm truyền đạt. Tuy nhiên để mô tả chính xác quá trình lan truyền của các mode dẫn trong ống dẫn sóng

Các hệ thống thông tin quang hữu tuyến sử dụng sợi quang, trong khi các hệ thống thông tin quang không dây truyền ánh sáng qua không gian tự do. Kiểu hệ thống thông tin quang phổ biến nhất sử dụng sợi đơn mode hoặc đa mode như kênh truyền, và có nguồn tin số nhị phân và một bộ thu thực hiện quyết định dựa trên năng lượng thu được trong mỗi chu kỳ bit. Ngoài việc gây suy hao các xung quang

phát đi, sợi quang còn gây méo hoặc dãn xung quang. Có hai cơ chế gây méo chính khác nhau: tán sắc sắc thể và tán sắc mode. Tán sắc sắc thể là kết quả của sự khác biệt về vận tốc lan truyền của các thành phần phổ khác nhau được phát đi và là nguyên nhân méo dạng chính trong các sợi quang đơn mode sử dụng phổ biến hiện nay. Trong điều kiện hoạt động tuyến tính, sợi quang cũng có thể được mô hình hóa bằng một hàm truyền đạt trong đó băng thông truyền dẫn phụ thuộc vào tán sắc như sau:

$$H_f(z,\omega) = e^{\left(j\frac{\beta_2\omega^2 z}{2} + j\frac{\beta_3\omega^3 z}{6}\right)} \tag{5.6}$$

trong đó z là khoảng cách truyền dẫn, β_2 và β_3 là các hệ số lan truyền bậc 2 và bậc 3 tương ứng của sợi quang đơn mode, các hệ số này quan hệ với tham số tán sắc của sợi qua các hệ thức sau:

$$\beta_2 = -\frac{\lambda_0^2 D}{2\pi c} \quad \text{và} \quad \beta_3 = -\left(\frac{\lambda_0^2}{2\pi c}\right)^2\left(\frac{2D}{\lambda_0} + S\right) \tag{5.7}$$

ở đây λ_0 là bước sóng tín hiệu, D và S là hệ số tán sắc và độ dốc tán sắc của sợi quang đơn mode. Tuy nhiên trong nhiều hệ thống thông tin sợi quang hiện nay, hiệu ứng phi tuyến cũng là ảnh hưởng cần xem xét, do vậy mô hình truyền dẫn trong sợi quang đơn mode phải được mô tả bởi phương trình Schrodinger phi tuyến. Việc giải phương trình này đòi hỏi một số kỹ thuật đặc thù sẽ không được đề cập đến trong nội dung bài giảng này.

5.3.2 Kênh vô tuyến

Các kênh vô tuyến đã được sử dụng từ rất sớm cho các hệ thống truyền thông đường dài, bắt đầu từ thử nghiệm của Marconi trong điện báo vô tuyến. Sự truyền lan sóng vô tuyến qua khí quyển gồm cả tầng điện ly-kéo dài vài trăm km trên bề mặt trái đất là một hiện tượng rất phức tạp. Truyền lan qua khí quyển phụ thuộc vào rất nhiều nhân tố như tần số, độ rộng băng tần tín hiệu, kiểu ăng ten, địa thế giữa ăng ten phát và ăng ten thu(nông thôn, thành phố, trong nhà, ngoài trời,v.v….)và các điều kiện thời tiết (khí quyển, mưa, sương mù,v.v.). Các nhà khoa học về khí quyển đã có cố gắng đáng kể để hiểu và triển khai các mô hình nhằm mô tả quá trình truyền lan sóng vô tuyến qua khí quyển. Ngoài ra, rất nhiều chương trình đo đạc đã thu được các dữ liệu truyền dẫn thực nghiệm từ tần số cao HF tới sóng viba. Tất cả những nỗ lực đó đã cung cấp cách để mô hình hóa quá trình truyền lan sóng

vô tuyến qua khí quyển và cách sử dụng những mô hình đó để phân tích, thiết kế và mô phỏng các hệ thống truyền thông hiện đại.

Từ quan điểm thiết kế truyền thống, ta phân các mô hình lan truyền thành hai loại sau: (i) mô hình ảnh hưởng phạm vi rộng (được dùng để tính toàn tổn hao truyền sóng); (ii) mô hình ảnh hưởng phạm vi hẹp (được dùng để mô hình hóa méo thông tin do hiệu ứng đa đường, hoặc do sự thay đổi ngẫu nhiên đặc tính truyền lan của kênh). Trong khi, giai đoạn 1 được dùng để thiết lập độ dự trữ công suất tuyến và phân tích vùng phủ sóng trong giai đoạn thiết kế khởi đầu, thì loại thứ hai được dùng để thiết kế chi tiết cho hệ thống.

Một kênh "gần như" là không gian tự do được giả định rằng coi khí quyển là môi trường hoàn toàn đồng nhất, không hấp thụ để khảo sát vùng truyền lan sóng giữa ăng ten phát/thu (không xảy ra các hiện tượng hấp thụ hoặc phản xạ sóng vô tuyến RF). Trong mô hình được lý tưởng hóa này, kênh chỉ làm suy hao tín hiệu và không gây méo dạng sóng. Suy hao được tính theo mô hình truyền lan không gian tự do được định nghĩa là:

$$L_f = \left(\frac{4\pi d}{\lambda}\right)^2 \tag{5.8}$$

trong đó λ là bước sóng của tín hiệu phát, d là khoảng cách ăng ten phát/thu vô hướng. Khuếch đại ăng ten phát và ăng ten thu được xem xét khi tính toán công suất thu thực tế.

Đối với đa số các kênh thực tế trong đó tín hiệu truyền lan qua khí quyển và gần mặt đất, việc giả định kênh truyền lan không gian tự do không còn phù hợp nữa. Ảnh hưởng đầu tiên phải xét đến là khí quyển, nó gây ra hấp thụ, khúc xạ và tán xạ. Sự hấp thụ do khí quyển khi xem xét trên băng tần hẹp sinh ra suy hao bổ sung. Tuy nhiên, với băng rộng tổn hao lại phụ thuộc vào tần số và thường được mô hình hóa bởi một hàm truyền đạt. Hiệu ứng lọc này có thể được coi như bất biến theo thời gian hoặc giả tĩnh (vì kênh thay đổi rất chậm so với tín hiệu). Một hiện tượng thuộc khí quyển khác nữa là méo pha do tầng điện ly, có thể được mô hình hóa bởi một đáp ứng pha thay đổi chậm theo thời gian hoặc bất biến theo thời gian. Một số ví dụ về các mô hình hàm truyền đạt được dùng để đặc tính hóa các loại kênh khí quyển cụ thể được mô tả dưới đây.

Các ảnh hưởng của khí quyển khác, mặt đất, những vật thể có trên mặt đất gần với đường truyền sóng thường tạo ra truyền dẫn đa đường. Truyền dẫn đa đường là tín hiệu đến qua hiều đường phản xạ và/hoặc khúc xạ từ máy phát đến

máy thu. Các ảnh hưởng này có thể biến đổi theo thời gian do sự thay đổi trong các điều kiện khí quyển hoặc sự chuyển động tương đối của ăng ten phát và ăng ten thu như trong thông tin di động. Thuật ngữ lấp lánh được bắt nguồn từ thiên văn học vô tuyến, được sử dụng để mô tả tính thay đổi theo thời gian trong các đặc tính kênh do các thay đổi về vật lí trong môi trường truyền lan như các thay đổi mật độ điện tử trong tầng điện ly gây phản xạ sóng vô tuyến tần số cao. Thuật nhữ pha đinh đa đường được dùng trong truyền thông di động để mô tả các thay đổi trong các điều kiện kênh và trong các đặc tính tín hiệu thu.

Kênh tầng đối lưu: Truyền thông tầng đối lưu (không phải điện ly) sử dụng băng tần VHF (30 đến 300 MHz) và UHF (300 MHz tới 3 GHz) để truyền tin trên cự ly vài nghìn km. Trong những băng tần này, ôxy và hơi nước có trong khí quyển sẽ hấp thụ năng lượng RF. Tổn hao do hấp thụ phụ thuộc vào tần số của sóng RF cũng như điều kiện khí quyển, độ ẩm thực tế. Tập các đặc tính điển hình về tổn hao truyền lan do sự hấp thụ khí quyển thường được thấy ở nhiều tài liệu.

Các đặc tính hấp thụ mang tính chọn lọc tần số của khí quyển có thể được xấp xỉ bằng một hàm truyền đạt ở dạng:

$$H(f) = H_0 \exp\left\{j0.02096f\left[10^6 + N(f)\right]l\right\} \qquad (5.9)$$

trong đó $N(f)$ là khúc xạ phức của khí quyển và được cho bởi:

$$N(f) = N_0 + D(f) + jN''(f) \qquad (5.10)$$

Trong các biểu thức (5.9) và (5.10) thì H_0 là hằng số, N_0 là khúc xạ phụ thuộc tần số, $D(f)$ là hấp thụ khúc xạ, $N''(f)$ là độ hấp thụ, l là khoảng cách (km). Các tham số này phụ thuộc vào tần số và các điều kiện khí quyển như nhiệt độ, áp suất, độ ẩm. Các giá trị điển hình thường được trình bày ở dạng bảng dữ liệu [1].

Một khi biết trước các điều kiện khí quyển và độ rộng băng tần của tín hiệu, thì có thể tính được hàm truyền đạt theo thực nghiệm tại các giá trị tần số khác nhau theo (5.9) và (5.10). Có thể tìm được hàm truyền đạt tương đương băng gốc bằng cách dịch tần số và dùng kỹ thuật FIR để mô phỏng mô hình kênh này.

Ảnh hưởng của mưa lên kênh vô tuyến: Mưa có ảnh hưởng đáng kể lên truyền lan sóng viba tại các tần số cao (lớn hơn 10 GHz). Nhiều kỹ thuật đã được đề xuất trong các tài liệu để mô hình hóa các ảnh hưởng của mưa [1]. Suy hao do mưa rào là một hàm của tốc độ mưa và tần số. Khi tốc độ mưa và tần số tín hiệu càng lớn thì suy hao càng lớn. Theo đó, suy hao tăng khi tốc độ mưa và tần số tăng. Ngoài ra, xuất hiện các đỉnh cộng hưởng trong đặc tính suy hao. Các đỉnh cộng hưởng thường

xảy ra tại 22 GHz và 60 GHz. Đỉnh tại 22 GHz xảy ra do hơi nước và đỉnh tại 60 GHz xảy ra do các phân tử ôxy.

Suy hao do mưa rào thường được tính toán cho các vị trí địa lý cụ thể sử dụng tính thống kê về tỉ lệ mưa vùng đó. Với thông tin vệ tinh, tổn hao được tính toán là một hàm của góc ngẩng ăng-ten trạm mặt đất (theo chiều ngang) và tần số. Độ ngẩng càng thấp có nghĩa là có nhiều nước mưa hơn và tổn hao càng cao. Ảnh hưởng của mưa rào được tính toán cụ thể với xác suất ngừng hoạt động cho trước, là phân số của thời gian mà BER của tuyến vượt giá trị ngưỡng (thường là 10^{-3} cho thông tin thoại và 10^{-6} cho các tuyến số liệu).

Với các độ rộng băng thông (dải thông) tương đối nhỏ, các ảnh hưởng của mưa có thể được tính đến bằng cách đưa thêm thành phần tổn hao phụ vào mô hình kênh. Tuy nhiên, khi dải thông của tín hiệu lớn, sẽ thay đổi suy hao trên băng thông và cần có một mô hình kiểu hàm truyến đạt. Đáp ứng biên độ của hàm truyền đạt có một độ nghiêng tuyến tính (trên thang đo dB) và đáp ứng pha được giả sử là tuyến tính.

Các kênh truyền lan không gian tự do ở tần số cao sử dụng ăng ten định hướng cao, có đặc tính phân cực cụ thể. Khi bước sóng (tương ứng tần số sóng mang) lớn hơn rất nhiều so với kích thước phần tử khí quyển và khi không có các vật cản vật lý đối với đa đường, thì có thể sử dụng phân cực ăng ten để cách ly (phân lập) các kênh. Trong các hệ thống truyền thông sử dụng nhiều phân cực trực giao cho các tín hiệu khác nhau, thì phải xét đến hiệu ứng khử cực của mưa. Khử phân cực có nghĩa là năng lượng trong một phân cực rò ra hoặc kết hợp với năng lượng trong thành phần phân cực trực giao. Điều này gây ra xuyên âm. Nếu có hai tín hiệu được truyền trên các thành phần phân cực trực giao là:

$$\tilde{s}_i(t) = A_i(t)\exp[j\phi_i(t)], \qquad i = 1,2 \tag{5.11}$$

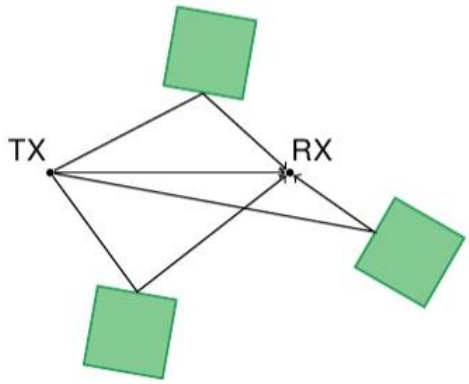
mô hình đơn giản nhất cho hai tín hiệu thu với khử phân cực là:

$$\begin{aligned} \tilde{r}_1(t) &= \alpha_{11}\tilde{s}_1(t) + \alpha_{12}\tilde{s}_2(t) \\ \tilde{r}_2(t) &= \alpha_{21}\tilde{s}_1(t) + \alpha_{22}\tilde{s}_2(t) \end{aligned} \tag{5.12}$$

trong đó tỉ số $20\log(\alpha_{11}/\alpha_{21})$ là số đo nhiễu phân cực chéo (XPI) lên tín hiệu 1 từ tín hiệu 2. Trong khi rất nhiều phép tính gần đúng khả dụng trong nhiều tài liệu để phân tích ảnh hưởng của XPI lên hệ thống số hoặc tương tự, thì nhu cầu mô phỏng tăng lên khi hệ thống lệch khỏi hệ thống lý tưởng.

5.3.3 Kênh pha đinh đa đường

Trong thông tin vô tuyến, đa đường là hiện tượng lan truyền của tín hiệu vô tuyến tới ăng ten thu bằng hai hoặc nhiều hơn hai đường truyền như mô tả trong hình 5-6. Các nguyên nhân của đa đường bao gồm hiệu ứng dẫn trong khí quyển, quá trình phản xạ và khúc xạ ở tầng điện ly, sự phản xạ từ mặt nước và các vật chướng ngại trên mặt đất như các ngọn núi và các tòa nhà. Các hiệu ứng đa đường gây ra sự giao thoa cộng hưởng và triệt tiêu, và sự dịch pha của tín hiệu.



Hình 5-6 Kênh đa đường

Các tín hiệu đa đường thu được ở các dạng lan truyền khác nhau khi tín hiệu tới bộ thu từ bộ phát thông qua các đường truyền khác nhau. Do đó sẽ có sự giao thoa đa đường dẫn đến pha đinh đa đường. Việc thêm vào sự chuyển động của hoặc bộ phát hoặc bộ thu hoặc các đối tượng xung quanh làm cho biên độ hoặc pha của toàn bộ tín hiệu thu được thay đổi trên một khoảng thời gian xác định gây ra pha đinh.

Một mô hình kênh pha đinh đa đường có thể được đặc trưng một cách tổng quát như một bộ lọc có đáp ứng xung

$$h(t) = \sum_{k=1}^{K} a_k . e^{j\phi_k} . e^{-j2\pi f_c \tau_k} . \delta(t - \tau_k) \qquad (5.13)$$

trong đó K đặc trưng cho số đường truyền tín hiệu từ bộ phát đến bộ thu trong đó mỗi đường truyền thứ k được đặc trưng bởi hệ số suy hao a_k, độ trễ τ_k và độ dịch pha ϕ_k tại thời điểm t, f_c là tần số sóng mang và δ là hàm xung Dirac. Từ phương trình (5.13) ta có thể thấy độ dịch pha bao gồm sự dịch pha vì quá trình lan truyền trong không gian tự do của thành phần đa đường thứ k và bất kì độ dịch pha bổ sung gặp phải trong kênh truyền. Tùy thuộc vào đặc tính của các tham số trong ptr. (5.13) mà ta có các mô hình kênh pha đinh đa đường khác nhau. Như vậy, khi tín hiệu đầu

vào là tín hiệu băng gốc $s(t)$ với tần số sóng mang f_c được truyền qua kênh pha đinh đa đường, tín hiệu thu được tương đương băng gốc có thể được biểu diễn bởi

$$r(t) = s(t) * h(t) = \sum_{k=1}^{K} a_k . e^{j\phi_k} . e^{-j2\pi f_c \tau_k} . s(t - \tau_k) \qquad (5.14)$$

Từ (5.14) ta thấy được tín hiệu thu được bao gồm tập hợp các bản sao của tín hiệu phát mà mỗi bản sao được định cỡ bởi $a_k . e^{j\phi_k} . e^{-j2\pi f_c \tau_k}$ và bị trễ bởi một lượng τ_k. Nói cách khác, có K đường truyền khác nhau tức là có K bản sao của cùng tín hiệu tới bộ thu tại thời điểm t. Mỗi bản sao trải qua các tổn hao khác nhau dẫn tới biên độ khác nhau, và quãng đường khác nhau sinh ra sự dịch pha và trễ khác nhau.

Đáp ứng tần số hay hàm truyền đạt của kênh có thể thu được từ (5.14) qua khai triển Fourier và thu được

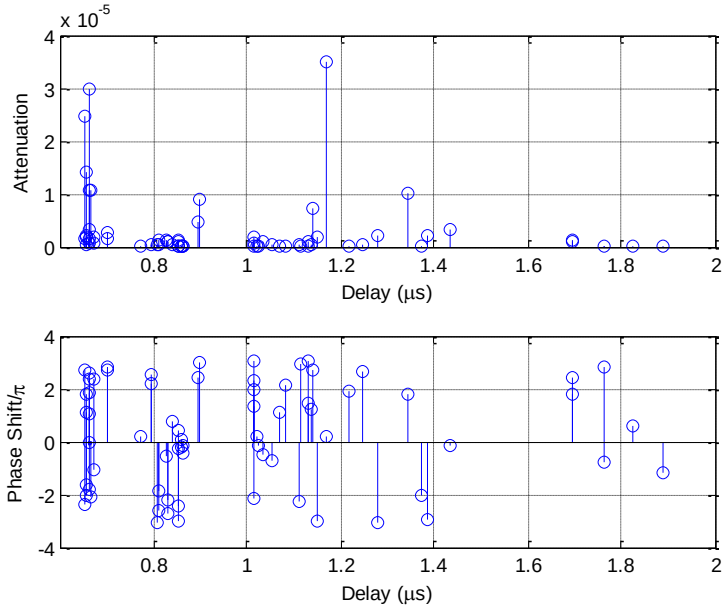
$$H(f) = \sum_{k=1}^{K} a_k . e^{j\phi_k} . e^{-j2\pi f_c \tau_k} . e^{-j2\pi f \tau_k} \qquad (5.15)$$

Tại một tần số f xác định, đáp ứng tần cũng là tổng của các số phức. Như vậy khi các số hạng này cộng triệt tiêu nhau thì đáp ứng tần là rất nhỏ hoặc thậm chí bằng 0 tại tần số đó. Các điểm null trong đáp ứng tần của kênh pha đinh đa đường là điển hình trong hệ thống thông tin vô tuyến và được xem như là pha định chọn lọc tần số.

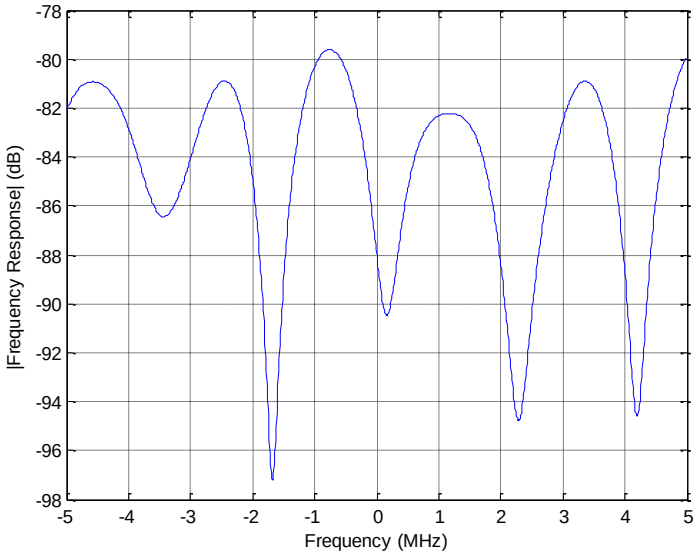
Trong trường hợp các kênh bất biến theo thời gian, các tham số trong đáp ứng xung hay đáp ứng tần không phụ thuộc vào biến thời gian t. Các tham số này có thể được xác định qua các phép đo kiểm kênh truyền. Khi các tham số được xác định, đáp ứng tần của kênh pha đinh đa đường dễ dàng tính được qua biểu thức (5.15). Một đoạn mã MATLAB tính toán đáp ứng tần của kênh pha đinh đa đường được cho thấy dưới đây:

```
% HH – Hàm đáp ứng tần số
% fc – tần số sóng mang
% amp – vector hệ số biên độ (độ lợi)
% tau – vector độ trễ
% ff – vector các thành phần tần số trong băng tần mô phỏng
% Lưu ý: Các vector tham số biên độ và độ trễ gồm K phần tử đặc trưng
% cho K đường truyền khác nhau của kênh.

HH = amp.*exp(-j*2*pi*fc*tau) * exp(-j*2*pi*tau'*ff);

% Chú ý: tau'*ff là phép nhân ma trận và tích của hai hàm mũ exp cũng
% là nhân ma trận → tạo ra phép tính tổng trong biểu thức đáp ứng
tần.
```

Một ví dụ về mô hình kênh pha đinh đa đường được cho thấy trong các hình 5-7 và 5-8. Trong ví dụ này, kênh đa đường được đặc trưng bởi 69 đường truyền khác nhau với các tham số được cho thấy trong hình 5-7. Hình 5-8 cho thấy đáp ứng tần tương ứng của kênh với các điểm notch đặc trưng cho tính chọn lọc tần số của kênh vô tuyến.

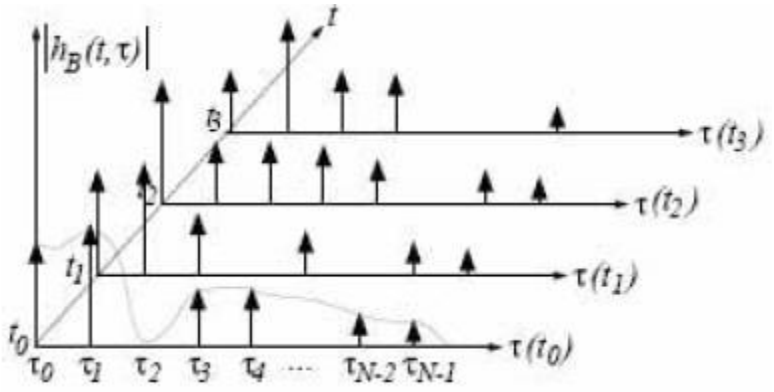


Hình 5-7 Ví dụ đáp ứng xung của kênh pha đinh đa đường cho thấy suy hao, trễ và pha của mỗi đường giữa bộ thu và bộ phát.



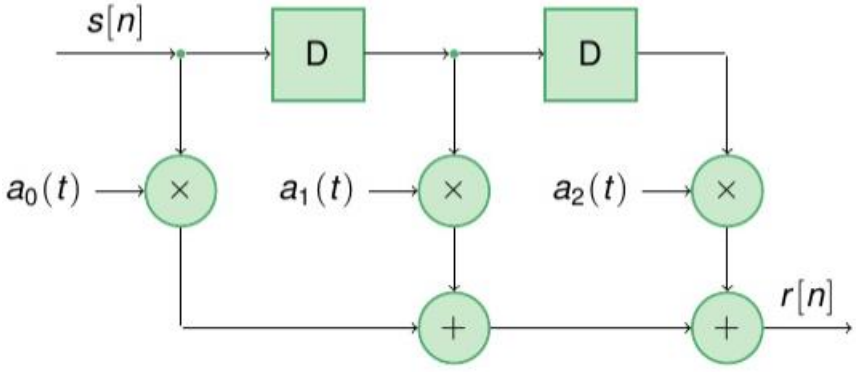
Hình 5-8 Đáp ứng tần của kênh pha đinh đa đường tương ứng với đáp ứng xung trong hình 5-7 được đặc trưng bởi các điểm notch.

Trong mô hình kênh biến đổi theo thời gian mà phần lớn các hệ thống vô tuyến hay gặp, các tham số biến đổi theo biến thời gian t hay $a_k = a_k(t)$, $\phi_k = \phi_k(t)$ và $\tau_k = \tau_k(t)$. Sự biến đổi của tất cả ba tham số này đều là các quá trình ngẫu nhiên. Độ dịch pha có thể được giả sử theo phân bố đều, còn hàm phân bố tham số biên độ phụ thuộc vào kiểu pha đinh cùng với khoảng cách lan truyền (tổn hao tuyến). Độ trễ thông thường được mô hình hóa theo phân bố Poisson. Một mô tả ví dụ về đáp ứng xung của một kênh đa đường biến đổi theo thời gian cho thấy trong hình 5-9, trong đó đối với một thời điểm t_i xác định, mỗi thành phần đường truyền có một độ trễ, biên độ và pha khác nhau.



Hình 5-9 Mô tả đáp ứng xung của kênh đa đường biến đổi theo thời gian

Phụ thuộc vào hàm phân bố biên độ các thành phần ta có các kiểu mô hình kênh pha đinh khác nhau. Hai hàm phân bố hay được sử dụng là hàm phân bố Rayleigh và hàm phân bố Rice tương ứng với các mô hình kênh pha đinh Rayleigh và mô hình kênh pha đinh Rice.



Hình 5-10 Mô hình kênh pha đinh đa đường biến đổi theo thời gian

Mô phỏng kênh pha đinh đa đường biến đổi theo thời gian cũng có thể được thực hiện như mô phỏng bộ lọc FIR hoạt động theo dạng rời rạc theo thời gian. Hình 5-10 mô tả sơ đồ khối mô hình kênh pha đinh đa đường theo mô hình bộ lọc FIR. Số lượng nhánh của bộ lọc được xác định bởi tích độ trải trễ và tốc độ lấy mẫu.

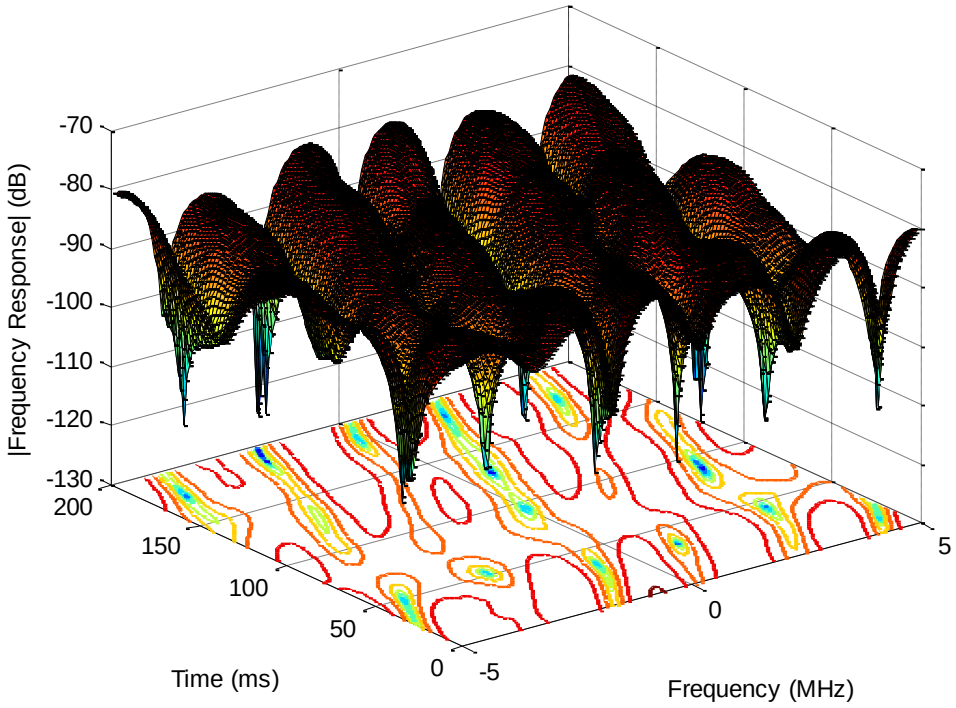
Các nhánh có trọng số phức là ngẫu nhiên tuân theo hàm phân bố Gauss. Biên độ trọng số của mỗi nhánh được phản ánh qua mặt cắt trễ công suất. Sự biến đổi theo thời gian của các nhánh có thể được đặc trưng bởi tham số dịch tần Doppler tức thời liên hệ với thành phần dịch pha mỗi nhánh có thể được biểu diễn như sau:

$$f_{di} = f_d \cos\alpha_i \tag{5.16}$$

trong đó α_i là góc tới của thành phần tín hiệu đường truyền thứ i, f_d là tần số dịch Doppler cực đại được xác định bởi

$$f_d = \frac{v}{c} f_c \tag{5.17}$$

với v là tốc độ chuyển động tương đối giữa bộ phát và bộ thu. Ví dụ về đáp ứng tần biến đổi theo thời gian được mô phỏng cho thấy trong hình 5-11.



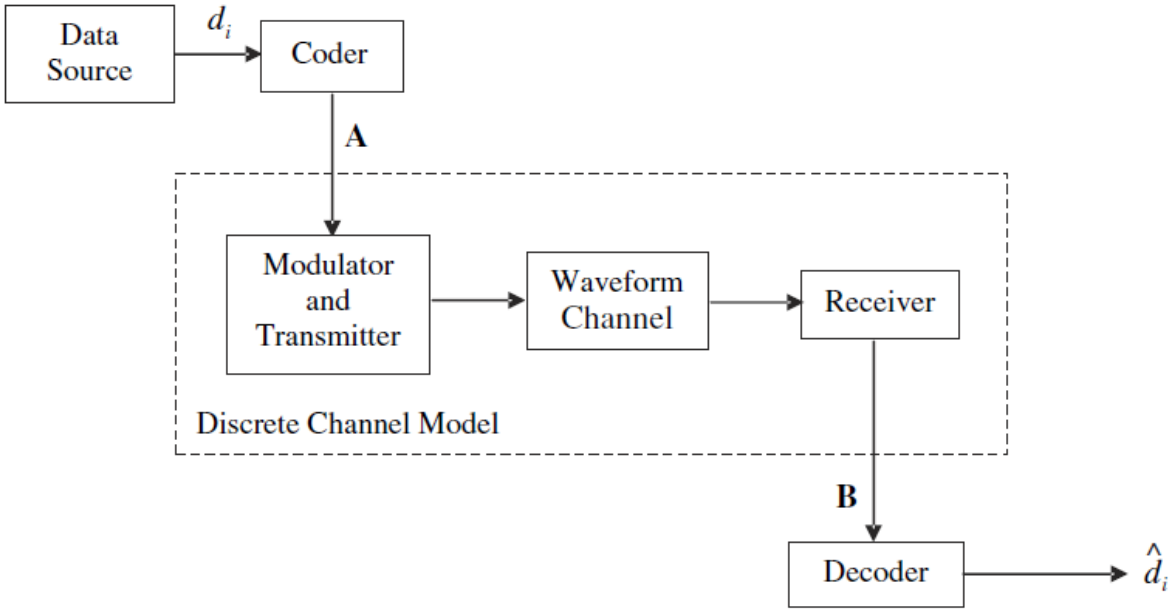
Hình 5-11 Sự phụ thuộc đáp ứng tần theo thời gian tại f_c = 1GHz, vận tốc v = 10m/s và tần số Doppler cực đại ≈ 33Hz.

Quá trình mô phỏng kênh pha đinh đa đường có thể được thực hiện trong miền thời gian thông qua tích chập như mô tả bởi ptr. (5.15) hay như hình 5-10. Cần lưu ý khi thực hiện phép tích chập trong MATLAB qua hàm `conv`, độ dài vector kết quả lớn hơn độ dài vector tín hiệu đầu vào, do vậy cần hiệu chỉnh độ dài vector nếu cần để đồng bộ với vector thời gian khi biểu diễn kết quả.

5.3.4 Kênh rời rạc

Mô hình cơ bản của một hệ thống thông tin có thể mô tả trong hình 5-12, bao gồm một nguồn dữ liệu rời rạc, một bộ mã hóa kênh để điều khiển lỗi, một bộ điều chế và bộ phát, một kênh thông tin, một bộ thu và một bộ giải mã. Phụ thuộc vào ứng dụng cho hệ thống và chi tiết của mô phỏng, các phần tử khác như bộ cân bằng, các bộ ghép/tách xen, bộ đồng bộ sóng mang và ký hiệu có thể là một phần của mô hình hệ thống. Như đã đề cập ở Chương 4, bộ điều chế sắp xếp một ký hiệu hoặc một chuỗi ký hiệu tại đầu vào bộ điều chế thành một dạng sóng tại đầu ra bộ điều chế. Dạng sóng sẽ bị tác động bởi một số hiệu ứng suy giảm trong kênh thông tin như là nhiễu, giới hạn băng tần, giao thoa và pha đinh. Tất cả các hiệu ứng này có thể được đặc trưng bởi dạng sóng. Đầu vào một bộ thu cũng là một dạng sóng bị méo dạng bởi kênh truyền. Vai trò của bộ thu là để quan sát dạng sóng thu được trên một chu kì của ký hiệu hoặc chuỗi ký hiệu để xác định các ký hiệu được phát đi. Kênh truyền trong trường hợp này được xem như là kênh dạng sóng vì cả hai đầu vào và đầu ra của kênh được đặc trưng như dạng sóng. Nó rất quan trọng để lưu ý rằng tất cả các phần tử hệ thống với ngoại trừ kênh thông tin đều được mô tả bởi các sắp xếp xác định. Kênh thông tin thực hiện một sự sắp xếp ngẫu nhiên hay thông kê của tín hiệu đầu vào tới đầu ra của kênh.

Thuật ngữ mô hình kênh rời rạc được sử dụng để biểu thị tất cả các phần tử của một hệ thống thông tin nằm giữa hai điểm A và B trong hệ thống, trong đó đầu vào tại điểm A là một vector các ký hiệu rời rạc (chuỗi đầu vào) ký hiệu là $\mathbf{X} = [x_1, x_2, \ldots, x_k, \ldots]$ và đầu ra tại B là một vector các ký hiệu rời rạc khác (chuỗi đầu ra) ký hiệu là $\mathbf{Y} = [y_1, y_2, \ldots, y_k, \ldots]$. Thông thường điểm A sẽ là đầu ra bộ mã hóa kênh hoặc tương đương đầu vào bộ điều chế, và điểm B sẽ là đầu vào bộ giải mã như cho thấy trong hình 5-12. Chú ý rằng, với sự phân chia cho thấy trong hình 5-12, bộ điều chế phát, kênh dạng sóng và bộ thu là phần của mô hình kênh rời rạc. Mối quan hệ giữa vector đầu vào kênh rời rạc **X** và vector đầu ra **Y** sẽ bị tác động bởi các thành phần hệ thống như các bộ lọc và bởi các nhiễu loạn ngẫu nhiên xảy ra trong kênh vật lý dạng sóng. Trong một hệ thống thông tin nhị phân với các bộ thu quyết định cứng, các phần tử của cả **X** và **Y** sẽ là các chuỗi nhị phân. Trong trường hợp các bộ thu quyết định mềm, các phần tử của **Y** sẽ là từ một danh sách ký tự M mức. Các lỗi truyền dẫn sinh ra từ những khiếm khuyết trong các phần tử hệ thống giữa các điểm A và B bao gồm cả kênh vật lý sẽ làm cho các phần tử của **X** và **Y** đôi lúc khác nhau.



Hình 5-12 Mô hình kênh rời rạc trong hệ thống thông tin.

Các mô hình kênh rời rạc mô tả cơ chế tạo lỗi theo xác suất. Các mô hình này thuộc trong hai loại sau:

- Loại mô hình đầu tiên được xem là các mô hình kênh không nhớ, được sử dụng để mô hình hóa các lỗi truyền dẫn hoặc các chuyển tiếp từ đầu vào tới đầu ra của kênh dưới điều kiện giả sử rằng không có sự tương quan về thời gian trong cơ chế chuyển tiếp. Đó là xác suất chuyển tiếp đầu vào – đầu ra đối với ký hiệu đầu vào kênh thứ n không bị tác động bởi những gì xảy ra với bất kỳ ký hiệu đầu vào khác. Các mô hình như vậy có thể áp dụng đối với các kênh thông tin mà không có sự giao thoa giữa các ký tự (ISI) hay pha đinh và nhiễu là AWGN. Trong một hệ thống nhị phân quyết định cứng, sự giả sử này bao hàm rằng các lỗi bit không tương quan với nhau. Các mô hình kênh rời rạc cho các kênh không nhớ luôn thu được nghiệm tầm thường.
- Loại mô hình kênh rời rạc thứ hai áp dụng cho các trường hợp trong đó các chuyển tiếp từ các ký hiệu đầu vào đến các ký hiệu đầu ra là tương quan về thời gian. Trong trường hợp này, xác suất lỗi cho ký hiệu thứ n phụ thuộc vào liệu một lỗi có xảy ra trong truyền dẫn các ký hiệu trước hay không. Kênh pha đinh thường gặp phải trong thông tin vô tuyến là một ví dụ tốt của kênh thông tin có các lỗi tương quan. Các lỗi sẽ xu hướng xảy ra theo cụm khi kênh vô tuyến đang ở trạng thái pha đinh sâu,

và những kênh này được xem như là các kênh lỗi cụm hoặc các kênh có nhớ.

Các mô hình kênh rời rạc là các mô hình xác suất có hiệu quả về tính toán hơn so với các mô hình kênh dạng sóng. Hiệu quả được cải thiện tới từ hai yếu tố. Các mô hình rời rạc được mô phỏng tại tốc độ ký hiệu, trái lại các mô hình kênh dạng sóng thường được mô phỏng tại tốc độ từ 8 đến 16 lần tốc độ ký hiệu. Chính điều này đã giảm tải trong tính toán ít nhất một bậc độ lớn. Trong khi mỗi khối đơn được mô phỏng chi tiết theo một mô hình dạng sóng, thì có một sự rút gọn mức cao trong mô hình kênh rời rạc. Mức rút gọn này sẽ giảm thêm khối lượng tính toán. Hai yếu tố này có thể giúp tiết kiệm thời gian tính toán cỡ vài bậc khi thực hiện mô phỏng.

Trong các trường hợp đơn giản, các mô hình kênh rời rạc có thể thu được bằng giải tích từ các mô hình của các thành phần nằm giữa đầu vào kênh tại A và đầu ra kênh rời rạc tại B. Tuy nhiên trong hầu hết trường hợp, các mô hình kênh rời rạc thu được từ các mẫu lỗi được đo kiểm hoặc mô phỏng giữa các điểm A và B. Các mô hình kênh rời rạc được sử dụng để thiết kế và phân tích tác động của các thành phần bên ngoài phần hệ thống giữa điểm A và B, ví dụ như các bộ mã hóa điều khiển lỗi, các bộ mã hóa nguồn và các bộ đan xen.

Mô hình hóa các kênh rời rạc không nhớ là một quá trình rõ ràng. Ví dụ, trong trường hợp của một kênh đối xứng rời rạc không nhớ với các đầu vào và đầu ra là tín hiệu nhị phân, tất cả những gì ta cần biết để mô tả kênh là một giá trị xác suất lỗi bít. Mô phỏng kênh này đòi hỏi tạo một số ngẫu nhiên đơn và so sánh số đó với một ngưỡng để quyết định liệu một bít xác định sẽ gặp lỗi truyền dẫn hay không hay thu được mà không bị lỗi. Như vậy, để mô phỏng truyền dẫn một triệu bít qua một kênh xác định, tất cả những gì chúng ta phải làm là tạo ra một triệu số ngẫu nhiên phân bố đều và thực hiện các so sánh ngưỡng được yêu cầu. Do đó, toàn bộ hệ thống được đặc trưng bởi kênh này được mô phỏng hiệu quả. (So sánh trường hợp này với mô phỏng dạng sóng cái đòi hỏi tạo ra hàng triệu mẫu đặc trưng cho dạng sóng trong hệ thống được mô phỏng và xử lý các mẫu này xuyên suốt tất cả các khối chức năng trong hệ thống)

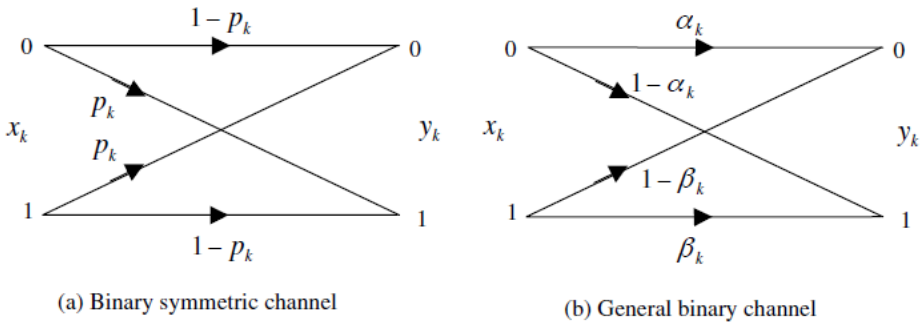
Các kênh rời rạc có nhớ khó để mô hình hóa hơn. Cơ chế tạo lỗi tương quan về thời gian thường được mô hình hóa bởi một chuỗi Markov rời rạc về thời gian trong đó một mô hình trạng thái được sử dụng để đặc trưng các trạng thái khác nhau của kênh thông tin và một tập xác suất chuyển tiếp trạng thái được sử dụng để xác

định tiến trình của các trạng thái kênh. Mỗi trạng thái cũng sẽ kết hợp với một tập xác suất chuyển tiếp ký hiệu từ đầu vào tới đầu ra. Như vậy, mô hình này phức tạp hơn và đòi hỏi nhiều tham số hơn kênh không tương quan. Cấu trúc mô hình và các giá trị tham số được ước tính từ hoặc các mẫu lỗi đo được hoặc mẫu lỗi mô phỏng. Mô phỏng kênh rời rạc bao gồm việc tạo ra một số ngẫu nhiên trước khi truyền dẫn mỗi ký hiệu để xác định trạng thái kênh, và sau đó rút ra một số ngẫu nhiên khác để xác định chuyển tiếp đầu vào – đầu ra. Trong khi mô hình và các thủ tục ước tính tham số là phức tạp, thì mô phỏng mô hình Markov cho thấy rất hiệu quả. …

Như vậy, mô phỏng kênh rời rạc đều đòi hỏi tạo mẫu lỗi ngẫu nhiên. Quá trình này có thể được thực hiện qua việc sử dụng các hàm tạo số ngẫu nhiên trong MATLAB. Ngoài các hàm tạo số ngẫu nhiên như mô tả trong Chương 4, ta có thể sử dụng hàm `randerr` trong MATLAB để tạo các mẫu lỗi bít. Đoạn mã dưới đây cho thấy ví dụ sử dụng hàm `randerr`:

```
% out = randerr(m) tạo ma trận nhị phân mxm với mỗi hàng có một vị
% trí lỗi ngẫu nhiên.
% out = randerr(m,n) tạo ma trận nhị phân mxn với mỗi hàng có một vị
% trí lỗi ngẫu nhiên.
% out = randerr(m,n,ne) tạo ma trận nhị phân mxn với mỗi hàng có
% số vị trí lỗi ngẫu nhiên được xác định bởi ne.

out = randerr(5,3,[0 2]) % Tạo mẫu lỗi bít với 2 kiểu lỗi gồm 0 và 2
vị trí lỗi ngẫu nhiên mỗi hàng

Kết quả:    out =

                 1     1     0
                 0     1     1
                 0     1     1
                 0     0     0
                 0     1     1

out2 = randerr(5,3,[0 2; .75 .25]) % Tạo mẫu lỗi bít với 2 kiểu lỗi
gồm 0 và 2 vị trí lỗi ngẫu nhiên mỗi hàng có xác suất lần lượt là 75%
và 25%.

Kết quả:    out2 =

                 0     0     0
                 1     0     1
                 0     0     0
                 0     0     0
                 0     0     0
```

```
% Ví dụ tạo lỗi cho một chuỗi bit đầu vào

dat = randint(10,3); % Tạo chuỗi từ mã nhị phân kích thước 10x3
err = randerr(10,3); % Tạo mẫu lỗi bít với 1 lỗi ngẫu nhiên mỗi hàng
dater = rem(dat+err,2); % Tạo chuỗi từ mã bị lỗi đầu ra
```



Hình 5-13 Mô hình kênh nhị phân

Mô hình kênh rời rạc không nhớ đơn giản nhất là kênh đối xứng nhị phân (BSC) như được mô tả trong hình 5-13a. Đầu vào kênh rời rạc này là một chuỗi nhị phân được ký hiệu bởi vector **X**. Thành phần thứ *k* của vector này, x_k, tương ứng với đầu vào kênh thứ *k*, và p_k tương ứng với xác suất lỗi lên truyền dẫn ký hiệu thứ *k*. Đối với một kênh không nhớ, p_k là độc lập với *k*, do vậy lỗi trên tất cả các ký hiệu bị tác động bởi kênh thông tin theo cùng một kiểu. (Nếu kênh là có nhớ, nhìn chung p_k sẽ là một hàm của p_{k-1}). Một chuỗi ký hiệu có độ dài *M* được xử lý qua kênh không nhớ bằng cách gọi mô hình kênh *M* lần liên tiếp. Đối với ký hiệu thứ *k*, một đầu vào nhị phân là 0 thu được chính xác là 0 với xác suất 1-p_k, và thu sai là 1 với xác suất p_k. Đầu ra kênh thứ *k* được kí hiệu là y_k và chuỗi *M* ký hiệu đầu ra kênh được ký hiệu bởi vector **Y**. Kênh là đối xứng khi các bít 0 và 1 bị ảnh hưởng bởi kênh truyền theo cùng một kiểu. Lưu ý rằng kết quả của tính đối xứng là xác suất lỗi độc lập với ký hiệu được phát đi, do đó nguồn lỗi có thể được mô phỏng tách biệt với nguồn tin (dữ liệu). Đối với một kênh nhị phân, quan hệ đầu vào-đầu ra có thể được biểu diễn như sau

$$\mathbf{Y} = \mathbf{X} \oplus \mathbf{E} \tag{5.18}$$

trong đó **X** và **Y** là các vector dữ liệu đầu vào và đầu ra tương ứng, $\oplus$ là phép toán XOR, và **E** là vector lỗi. Cụ thể, **E** = {e_1, e_2, e_3, …} là một vector nhị phân hoặc chuỗi có các phần tử {0, 1} trong đó e_k = 0 chỉ ra phần tử thứ *k* của **X**, x_k, thu được chính xác ($y_k = x_k$), và e_k = 1 chỉ ra phần tử thứ *k* của **X** thu được bị lỗi ($y_k \neq x_k$). Tham số định nghĩa hiệu năng của mô hình kênh nhị phân đối xứng là xác suất lỗi P_e cái có thể được ước tính dễ dàng từ các phép đo hoặc từ mô phỏng hệ thống ở mức dạng sóng. Chú ý rằng đối với một hệ thống nhị phân, ước tính xác suất lỗi theo phương pháp Monte Carlo là trọng số Hamming của vector lỗi **E** chia cho N là số phần tử trong vector **E**. Khi p_k được biết, chuỗi lỗi tương ứng với N ký hiệu phát đi có thể được tạo ra bằng việc gọi N lần bộ tạo số ngẫu nhiên phân bố đều và so sánh số ngẫu nhiên với ngưỡng *p*. Cụ thể:

$$e_k = \begin{cases} 1, & U_k \le p_k \\ 0, & U_k > p_k \end{cases} \qquad (5.19)$$

trong đó U_k là số thu được từ lần gọi bộ tạo số ngẫu nhiên thứ k. Mô hình phức tạp hơn với xác suất lỗi bít 1 và 0 khác nhau như mô tả trong hình 5-13b và các mô hình nhiều đầu vào và đầu ra đều có thể mở rộng ra từ mô hình BSC.

Trong MATLAB, mô hình kênh BSC dễ dàng được xây dựng từ hàm `randerr` như trong ví dụ trước. Tuy nhiên ta có thể sử dụng hàm kênh đối xứng nhị phân bsc trong thư viện `Communications Toolbox`. Ví dụ về hàm `bsc` được cho trong đoạn mã dưới đây:

```
% Ví dụ sử dụng mô hình kênh BSC

>> d = randint(100,100); % Tạo bản tin ngẫu nhiên
>> ze = bsc(d,.05); % Truyền qua kênh đối xứng nhị phân
>> [ne, pe] = biterr(d,ze); % Xác định số lỗi và xác suất lỗi đầu ra
Kết quả:
      ne  =
         492
      pe  =
          0.0492
```

Một ví dụ khác trong MATLAB về sử dụng mô hình kênh BSC trong khảo sát hiệu năng mã sửa lỗi BCH trong hệ thống viễn thông cũng được cho trong đoạn mã dưới đây:

```
% Ví dụ sử dụng kênh BSC trong mô phỏng hiệu năng mã sửa lỗi BCH

m = 4; n = 2^m-1; % Độ dài từ mã đầu ra
k = 5; % Độ dài từ mã bản tin
nwords = 10000; % Số lượng từ mã để mã hóa
msg = gf(randint(nwords,k)); % Tạo bản tin và tạo mảng trường Galois
% Xác định đa thức tạo mã và t số lỗi có thể sửa.
[genpoly,t] = bchgenpoly(n,k);
% Mã hóa bản tin.
code = bchenc(msg,n,k);
% Truyền qua kênh BSC
[ncode,err] = bsc(code,.05); % Tạo các lỗi trong mã đầu vào.
% Giải mã bản tin thu được.
[newmsg,errs,dcode] = bchdec(ncode,n,k);
% Xác định lỗi.
numchanerrs = sum(sum(newmsg~=msg));
ber = numchanerrs/(nwords*k); % Tỉ lệ lỗi bít khi sử dụng mã BCH
numerrs = sum(sum(err));
ber0 = numerrs/(nwords*n); % Tỉ lệ lỗi bít gây ra bởi kênh truyền
```

Kết quả sau khi chạy chương trình:

```
ber =

    0.0021

ber0 =

    0.0499
```

Như vậy qua mô hình kênh rời rạc hiệu năng của mã sửa lỗi dễ dàng được mô phỏng mà không phải thực hiện tải tính toán lớn qua xử lý các mẫu tín hiệu trong kênh dạng sóng.

5.4 Tổng kết chương

Chương này đã tập trung vào phương pháp luận để mô phỏng kênh thông tin, một thành phần quan trọng trong hệ thống truyền thông. Các kiểu kênh thông tin khác nhau từ kênh cơ bản nhất là AWGN đến các kênh thông tin cụ thể bao gồm cả kênh hữu tuyến và kênh vô tuyến đã được trình bày cách tiếp cận để mô phỏng cho các hệ thống truyền thông khác nhau. Mô hình kênh rời rạc cũng được giới thiệu để thấy rõ phạm vi ứng dụng và ý nghĩa của mô hình này trong mô phỏng hệ thống truyền thông.

Câu hỏi/bài tập chương 5

5-1/ Kênh truyền AWGN sử dụng cho hai mô hình thông dải và mô hình tương đương băng gốc có gì khác biệt nhau?

5-2/ Viết chương trình MATLAB hệ thống truyền dẫn tương tự sử dụng kỹ thuật điều chế AM có f_c = 60 Hz và tín hiệu đầu vào có dạng:

$$s = \cos(8\pi t) + \cos(12\pi t + \pi/4)$$

Biết tín hiệu được truyền qua kênh AWGN có SNR bằng 8 dB. Hãy so sánh kết quả giữa tín hiệu gốc ban đầu và tín hiệu thu được sau khi giải điều chế.

5-3/ Tương tự bài 2 nhưng sử dụng kỹ thuật điều chế FM với độ sâu điều chế m = 0,4.

5-4/ Cho tín hiệu điều chế được cho trong bài tập 4-7 truyền qua kênh AWGN có SNR bằng 10 dB. Hãy biểu diễn biểu đồ mắt của tín hiệu trước và sau khi truyền qua kênh.

5-5/ Mô phỏng ảnh hưởng của kênh truyền cáp đồng trục dài 1 km được mô tả bởi phương trình (5.5) lên tín hiệu băng gốc sử dụng mã đường NRZ tại tốc độ 100 Kb/s. Biết cáp đồng trục có trở kháng 50 Ω, độ cảm L = 235.7 nH/m và điện dung C = 94.2 pF/m. Giả sử không xét đến suy hao của kênh truyền. Tín hiệu sẽ bị ảnh hưởng như thế nào nếu sử dụng mã RZ 50%.

5-6/ Giả sử rằng một kênh BSC được xác định bởi xác suất lỗi $p = 10^{-2}$ với p_k độc lập với k. Xây dựng chương trình MATLAB sử dụng hàm `randerr` để mô phỏng truyền dẫn N ký hiệu qua kênh với N = 100, N = 1000, và N = 10000. Trong mỗi trường hợp đếm số lỗi thực xảy ra và tính BER thu được từ mô phỏng. Lặp lại thí nghiệm 10 lần cho mỗi giá trị của N. Hãy đưa ra kết luận sau khi thực hiện.

5-7/ Mô phỏng ảnh hưởng tán sắc trong sợi đơn mode lên xung Gauss có độ rộng xung là 10 ps. Biết sợi quang có hệ số tán sắc D = 17 ps/nm-km và S ≈ 0 với khoảng cách L = 200 m, 500 m và 1 km.

5-8/ Cho kênh pha đinh đa đường Rayleigh được mô tả bởi 2 tia có đặc điểm cho trong bảng sau:

Trường hợp	P_1	P_2	Trễ (mẫu)
1	1	0.2	0
2	1	0.2	$1/2T_{sym}$

Hãy mô phỏng ảnh hưởng của kênh lên tín hiệu QPSK tại mức SNR = 8 dB và so sánh với trường hợp kênh AWGN.

Chương 6 Ước tính các tham số và đánh giá hiệu năng

Do tính thống kê trong quá trình mô phỏng hệ thống truyền thông, do vậy kỹ thuật ước tính các tham số có ý nghĩa trong mô phỏng, đặc biệt là ước tính hiệu năng. Chương này sẽ đề cập đến một số vấn đề cơ bản trong ước tính tham số và đánh giá hiệu năng hệ thống được mô phỏng.

6.1 Ước tính các tham số

6.1.1 Ước tính mức sóng trung bình

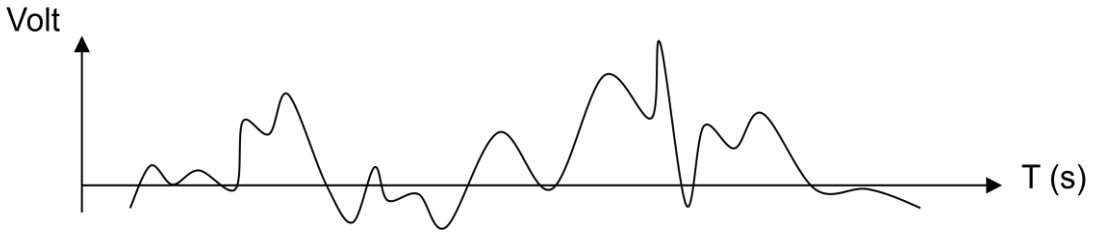
Ước tính mức sóng trung bình có thể được thực hiện trong hoặc trước khi chạy mô phỏng thường nhằm mục đích hiệu chỉnh. Gọi $x(t)$ là dạng sóng quan tâm như mô tả trong hình 6-1 cùng với các mẫu $x(kT_s) \equiv x_k$ cách nhau một khoảng lấy mẫu T_s. Bộ ước tính tự nhiên của mức trung bình là trung bình mẫu

$$\langle \mathbf{x} \rangle = \frac{1}{N} \sum_{k=1}^{N} x_k \tag{6.1}$$

trong đó N là số mẫu. Một bộ ước tính cũng là một biến ngẫu nhiên, do vậy giá trị kỳ vọng (hay trung bình) của nó được xác định

$$E\left(\langle \mathbf{x} \rangle\right) = E\left(\frac{1}{N} \sum_{k=1}^{N} x_k \right) = \frac{1}{N} \sum_{k=1}^{N} E\left(x_k\right) = \frac{1}{N} \sum_{k=1}^{N} E\left(x\right) = E\left(x\right) \tag{6.2}$$

trong đó tính phân bố của kỳ vọng và tính dừng được sử dụng. Như vậy giá trị trung bình của mẫu cũng là giá trị trung bình của tập phân bố.



Hình 6-1 Mô tả dạng sóng thống kê

Phương sai của bộ ước tính là trung bình cộng của bình phương độ lệch giữa giá trị đo và giá trị kì vọng

$$Var\left(\langle \mathbf{x} \rangle\right) = E\left(\langle \mathbf{x} \rangle = E\left(\langle \mathbf{x} \rangle\right)\right)^2 = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} C_{xx}(i,j) \tag{6.3}$$

trong đó C_{xx} là tự tương quan của quá trình

$$C_{xx}(i,j) = E\left(\left[x_i - E(x_i)\right]\left[x_j - E(x_j)\right]\right) \tag{6.4}$$

Từ tính dừng ta có $E(x_i) = E(x_j) = E(x)$ và

$$C_{xx}(i,j) = C_{xx}(i-j) \tag{6.5}$$

Như vậy, thay (6.5) vào (6.3) và đánh lại ký hiệu $i - j = k$, ta có

$$Var\left(\langle \mathbf{x} \rangle\right) = \frac{\sigma_x^2}{N} + \frac{2}{N^2}\sum_{k=1}^{N-1}(N-k)C_{xx}(kT_s) \tag{6.6}$$

trong đó $C_{xx}(0) = \sigma_x^2$ là phương sai của quá trình ngẫu nhiên.

6.1.2 Ước tính công suất trung bình

Công suất trung bình của một dạng sóng là một thuộc tính quan trọng trong cả truyền dẫn tương tự và truyền dẫn số. Dạng sóng quan tâm thường bao gồm tín hiệu và nhiễu. Công suất trung bình là một tham số quan tâm ở mức độ chi tiết cao hơn trong thiết kế hệ thống. Ví dụ nó thường xác định mức kích thích đầu vào một bộ khuyếch đại, do đó có thể kiểm tra điểm hoạt động cần thiết được thực hiện thực sự.

Công suất trung bình P_{av} là trung bình của công suất trung thời gian hữu hạn $P_k(T_0)$ hay năng lượng trên khoảng thứ k được chia cho chu kỳ T_0. Nếu năng lượng trong khoảng thứ k $\xi_k(T_0)$ được cho là độc lập với k như trong một tín hiệu đường bao không đổi, thì có một kết nối rõ ràng giữa công suất trung bình và năng lượng. Trong thực tế, năng lượng sẽ thăng giáng từ khoảng này sang khoảng tiếp theo vì sự biến đổi đường bao vì quá trình lọc và vì sự có mặt của nhiễu.

Trong một số hệ thống năng lượng $\xi_k(T_0)$ luôn là ngẫu nhiên. Ví dụ trong một bộ thu quang trực tiếp, đầu ra của diode thu quang thác trong một khoảng thời gian xác định là một biến ngẫu nhiên thậm chí khi cường độ đầu vào được cố định. Trong một hệ thống như vậy, trong khi đầu ra trung bình có thể vẫn là thông tin hữu

ích thì những gì quan tâm thực sự là phân bố của đầu ra xác định cho một kiểu kí hiệu.

Dạng sóng $x(t)$ được lấy mẫu tại các khoảng T_s để thu được **x** và bộ ước tính công suất trung bình có dạng sau

$$P_N(\mathbf{x}) = \frac{1}{N}\sum_{k=1}^{N} x_k^2 \tag{6.7}$$

Sử dụng một biến trung gian $y(t) = x^2(t)$, bài toán trở thành ước tính trung bình của y(t) và có thể sử dụng kết quả của phần trước

$$\langle \mathbf{y} \rangle = \frac{1}{N}\sum_{k=1}^{N} y_k \tag{6.8}$$

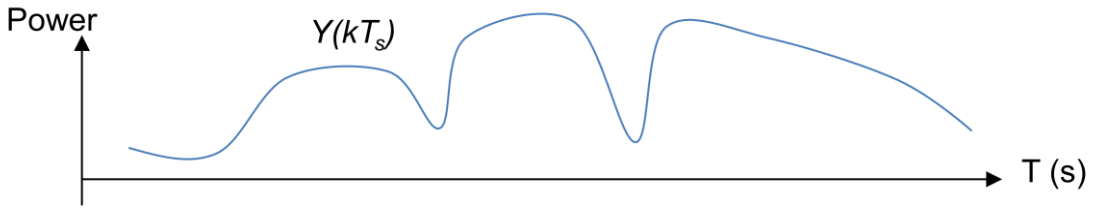
Nó rõ ràng rằng

$$E(\langle \mathbf{y} \rangle) = E(y) \tag{6.9}$$

cho một quá trình dừng. Do vậy

$$E(P_N(\mathbf{x})) = E(x^2) \tag{6.10}$$

hay giá trị kì vọng của bộ ước tính công suất trung bình mẫu hữu hạn là bằng với giá trị trung bình bình phương của quá trình, độc lập với N.



Hình 6-2 Mô tả ước tính công suất

Trong MATLAB một số hàm sẵn có cho phép sử dụng để ước tính các tham số thống kê có thể sử dụng đó là:

`mean(x)` – tính giá trị trung bình tập dữ liệu x

`var(x)` - tính phương sai tập dữ liệu x

`std(x)` - tính độ lệch chuẩn tập dữ liệu x

6.1.3 Ước tính phổ

Mật độ phổ công suất (PSD) thể hiện các tính chất miền tần số của một quá trình. Thuộc tính miền tần số quan trọng nhất là độ rộng băng tần cái có thể thu

được từ PSD. Có hai kỹ thuật cổ điển để ước tính phổ dựa trên định nghĩa mật độ phổ công suất: phương pháp gián tiếp và phương pháp trực tiếp.

Phương pháp gián tiếp: Theo định nghĩa PSD $S_{xx}(f)$ của quá trình $x(t)$ như là khai triển Fourier của hàm tự tương quan $R_{xx}(\tau)$

$$S_x(f) = \int_{-\infty}^{\infty} R_x(\tau)\exp(-j2\pi f\tau)d\tau, \qquad -\infty < f < \infty \tag{6.11}$$

Trong mô phỏng, tín hiệu $x(t)$ ở dạng rời rạc nên PSD trong miền rời rạc có dạng

$$P_x(f) = T_s \sum_{k=-\infty}^{\infty} R_x(k)\exp(-j2\pi fkT_s), \qquad |f| \le \tfrac{1}{2} f_s \tag{6.12}$$

trong đó $R_x(k)$ là hàm tự tương quan rời rạc thời gian

$$R_x(k) \equiv R_x(kT_s) = E[x(nT_s + kT_s)x(nT_s)] \equiv E[x(n+k)x(n)] \tag{6.13}$$

Tuy nhiên trong thực tế số lượng mẫu quan sát $\{x(n)\}$ là hữu hạn, nên bộ ước tính PSD sẽ được xác định như sau

$$\hat{P}_x(f) = T_s \sum_{k=-L}^{L} \hat{R}_x(k)\exp(-j2\pi fkT_s) \tag{6.14}$$

trong đó chỉ số k được như là trễ và trễ cực đại được sử dụng là L. Lưu ý rằng $\hat{P}_x$ là tuần hoàn theo f và vùng quan tâm là $|f| \le 1/2T_s$. Hàm tự tương quan lúc này sẽ được tính bởi

$$\hat{R}_x(k) = \frac{1}{N}\sum_{n=0}^{N-k-1} x(n+k)x^*(n) \tag{6.15}$$

trong đó $k = 0, 1, \ldots, N\text{-}1$ là số các mẫu quan sát, và x^* là liên hợp phức nếu x là số phức. $\hat{P}_x$ được xác định cho bất kỳ f nào trong khoảng $|f| \le 1/2T_s$.

Phương pháp trực tiếp: Hàm PSD được định nghĩa như sau

$$S_x(f) = \lim_{T\to\infty} E\left[\hat{S}_x(f,T)\right] \tag{6.16}$$

trong đó

$$\hat{S}_x(f,T) = \frac{|X_T(f)|^2}{T} \tag{6.17}$$

và

$$X_T(f) = \int_0^T x(t)\exp(-j2\pi ft)dt \tag{6.18}$$

là khai triển Fourier thời gian hữu hạn của một hàm mẫu của quá trình. Một bộ ước tính PSD dạng rời rạc thời gian $\hat{P}_x(f)$ được xác định như sau

$$\hat{P}_x(f) = \frac{1}{NT_s}\left|X_N(f)^2\right| \tag{6.19}$$

trong đó

$$X_N(f) = T_s\sum_{n=0}^{N-1} x(n)\exp(-j2\pi fnT_s) \tag{6.20}$$

là khai triển Fourier rời rạc của chuỗi dữ liệu quan sát.

Như vậy dựa trên giải thuật FFT, ước tính phổ sẽ được xác định hoặc bằng phương pháp gián tiếp hoặc bằng trực tiếp. Các ví dụ về tính toán PSD của tín hiệu sử dụng hàm `fft` đã được cho trong Chương 4 mục 4.1.

6.2 Ước tính tỉ số SNR

Phép đo hiệu năng tiêu chuẩn đối với một tín hiệu tương tự đó là tỉ số tín hiệu trên nhiễu (SNR) mặc dù SNR cũng là đại lượng đo chất lượng ý nghĩa cho bất kỳ tín hiệu nào. Xét một mô hình hệ thống như mô tả trong hình 6-3 với tín hiệu đầu ra $x(t)$ được xác định từ tín hiệu đầu vào $s(t)$ và hàm nhiễu $n(t)$. Để đơn giản tất cả các dạng sóng đầu vào và đầu ra được giả sử rằng đều là thực. Phân tích tín hiệu đầu ra thành các thành phần "tín hiệu" và "nhiễu" và có thể biểu diễn

$$\mathbf{x} = \mathbf{s}_0 + \mathbf{n}_0 \tag{6.21}$$

trong đó tín hiệu được biểu diễn như sau

$$\mathbf{s}_0 = A\mathbf{s}_\tau \tag{6.22}$$

với

$$s_\tau = \begin{cases} s(t-\tau) & 0 \le t \le T \text{ hoặc } s(iT_s - \tau),\ i = 1, 2, \ldots, N \\ 0, & \text{trường hợp khác} \end{cases} \tag{6.23}$$

Thành phần nhiễu được xem như là sai số trung bình bình phương nhỏ nhất trên tất cả các A và τ được định nghĩa như sau

$$\varepsilon^2 = \left\langle (\mathbf{x} - A\mathbf{s}_\tau)^2 \right\rangle \tag{6.24}$$

Nó có thể được chứng minh rằng (6.24) nhỏ nhất khi

$$u(\tau, A) = A^2 \left\langle \mathbf{s}_\tau^2 \right\rangle - 2AR_{xs}(\tau) \tag{6.25}$$

cũng là nhỏ nhất, trong đó

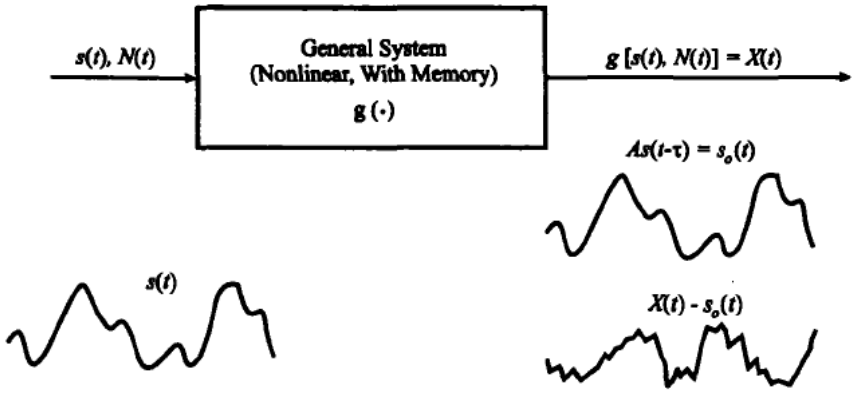
$$\left\langle \mathbf{s}_\tau^2 \right\rangle = \frac{1}{N} \sum_{k=1}^{N} s^2(kT_s - \tau) \tag{6.26}$$

là công suất trung bình N điểm, và

$$R_{xs}(\tau) = \frac{1}{N} \sum_{k=1}^{N} x(kT_s) s(kT_s - \tau) \tag{6.27}$$

được xem là hàm tương quan chéo thực nghiệm. Do vậy ước tính SNR $\hat{\rho}$ được xác định như sau:

$$\hat{\rho} = \frac{\left\langle \mathbf{s}_0^2 \right\rangle}{\varepsilon^2} \tag{6.28}$$



Hình 6-3 Định nghĩa tỉ số tín hiệu trên nhiễu

Có hai cách tiếp cận để tính toán ước tính SNR đó là tiếp cận miền thời gian và tiếp cận miền tần số. Nhiệm vụ chính của quá trình tính toán ước tính SNR là xác định độ trễ tốt nhất τ_* để (6.25) đạt cực tiểu.

Thủ tục mô phỏng miền thời gian để ước tính SNR:

- Chạy mô phỏng và thu được M mẫu đầu ra của hệ thống $\{x(k)\}$, k = 1, 2, …, M và $N = M + K$ mẫu tín hiệu đầu vào $\{s(k)\}$, k = -(K-1), -(K-2), …, 0, 1, …, M. K mẫu bổ sung của tín hiệu đầu vào cần được lựa chọn để trải trên khoảng trễ lớn nhất trong hệ thống τ_{max}, và hàm tương quan chéo thực nghiệm luôn chứa N điểm.
- Tính tương quan chéo thực nghiệm theo

$$R_{xs}(j) = \frac{1}{N}\sum_{k=1}^{N} x(k)s(k-j), \quad j = -(k-1),\ldots,0$$

và tìm giá trị của j là j_* để thu được giá trị cực đại của $R_{xs}(j)$.

- Tính các đại lượng

$$\left\langle \mathbf{X}^2 \right\rangle = \frac{1}{N}\sum_{k=1}^{N} x^2(k)$$

$$\left\langle s^2 \right\rangle = \frac{1}{N}\sum_{k=1}^{N} s^2(k-j_*)$$

- Tính toán $\hat{\rho}$, ước tính SNR từ

$$\hat{\rho} = \frac{R_{xs}(j_*)}{\sqrt{\left\langle \mathbf{X}^2 \right\rangle \left\langle s_*^2 \right\rangle - R_{xs}^2(j_*)}}$$

Thủ tục mô phỏng miền tần số để ước tính SNR:

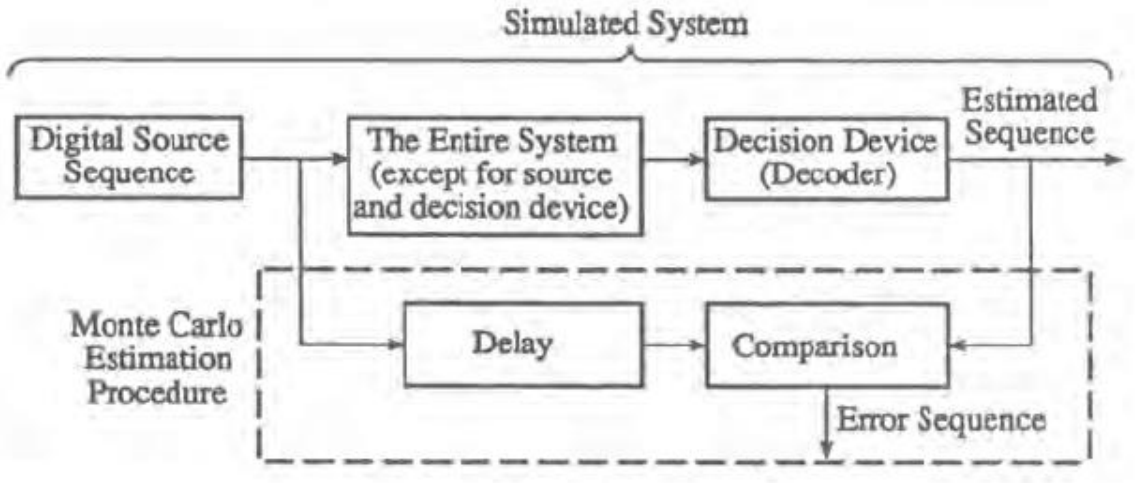
- Giống như thủ tục miền thời gian, chạy mô phỏng và thu được M mẫu đầu ra của hệ thống $\{x(k)\}$, $k = 1, 2, \ldots, M$ và $N = M + K$ mẫu tín hiệu đầu vào $\{s(k)\}$, $k = -(K-1), -(K-2), \ldots, 0, 1, \ldots, M$. K mẫu bổ sung của tín hiệu đầu vào trải trên khoảng trễ mong đợi lớn nhất τ_{max},
- Thực hiện khai triển Fourier rời rạc ngược (IDFT) đối với chuỗi $\{X_m^* S_m\}$, trong đó $\{X_m^*\}$ liên hợp phức của DFT của $\{x_m^*\}$ và $\{S_m\}$ là DFT của chuỗi $\{s_m\}$. Chú ý rằng chuỗi $\{x_m\}$ trước hết phải được chèn thêm K giá trị 0. Số hệ số k mà biên độ của nó là lớn nhất tương ứng với trễ được tìm thấy. Chú ý rằng chỉ $K + 1$ hệ số đầu tiên cần được tìm kiếm.
- Tính toán $\hat{\rho}$ theo biểu thức tương tự thủ tục miền thời gian với τ được đặt tại giá trị tìm được ở bước trên.

6.3 Đánh giá hiệu năng hệ thống

Trong các hệ thống truyền thông số, đại lượng đánh giá hiệu năng hệ thống quan trọng nhất là tỉ lệ lỗi ký hiệu (SER) hoặc tỉ lệ lỗi bit (BER) đối với tín hiệu nhị phân. Có một số phương pháp để ước tính các tham số trên bao gồm phương pháp Monte-Carlo, phương pháp bán giải tích, phương pháp lấy mẫu quan trọng và phương pháp ngoại suy đuôi phân bố.

6.3.1 Phương pháp Monte-Carlo

Phương pháp Monte-Carlo là một lớp rộng lớn các giải thuật tính toán dựa trên việc lấy mẫu ngẫu nhiên lặp lại để thu được các kết quả số (gần đúng); thông thường bằng việc chạy mô phỏng nhiều lần để thu được phân bố của một thực thể xác suất chưa biết.



Hình 6-4 Mô phỏng Monte-Carlo

Trong mô phỏng hệ thống truyền thông, phương pháp Monte-Carlo mô tả một mô phỏng mà một tham số của hệ thống như tỉ lệ lỗi bit (BER) hay tỉ lệ lỗi ký hiệu (SER) được ước tính bằng sử dụng kỹ thuật Monte-Carlo. Ước tính Monte-Carlo là quá trình ước tính giá trị của tham số bằng thực hiện các phép thử ngẫu nhiên hay thống kê. Ý tưởng thực hiện mô phỏng Monte-Carlo thực tế rất đơn giản như sau:

- Mô phỏng hệ thống lặp lại, việc lặp lại để đảm bảo đếm được đủ số lỗi xảy ra theo yêu cầu để đảm bảo độ tin cậy của kết quả ước tính.
- Mỗi lần chạy mô phỏng, đếm số lượng ký hiệu được phát đi và số lượng lỗi ký hiệu như mô tả theo sơ đồ khối trong hình 6-4. Các ký hiệu tại đầu ra bộ quyết định sẽ được so sánh với các ký hiệu được phát đi để phát hiện lỗi xảy ra. Khối *delay* được sử dụng để đồng bộ các ký hiệu tương ứng trước khi so sánh.
- Ước tính tốc độ lỗi ký hiệu SER như là tỉ lệ giữa tổng số lỗi quan sát được trên tổng số ký hiệu truyền đi.

$$\hat{P}_e = \frac{N_e}{N} \tag{6.29}$$

trong đó N_e và N là tổng số lỗi đếm được và tổng số ký hiệu truyền đi sau M lần chạy mô phỏng lặp lại.

Một điều cần lưu ý là các lỗi xảy ra ngẫu nhiên trong quá trình mô phỏng, do vậy cần xem xét đặc tính thống kê của bộ ước tính $\hat{P}_e$. Khi các sự kiện lỗi độc lập xảy ra, xác suất để N_e lỗi xảy ra trong N ký hiệu truyền dẫn được xác định bởi phân bố nhị thức

$$p_N(N_e) = \binom{N}{N_e} P_e^{N_e} (1-P_e)^{N-N_e} \tag{6.30}$$

trong đó

$$\binom{N}{k} = \frac{N!}{k!(N-k)!} \tag{6.31}$$

là hệ số nhị thức và P_e là xác suất lỗi truyền dẫn hay được xem là xác suất lỗi thực. Giá trị trung bình và phương sai của một biến ngẫu nhiên tuân theo phân bố nhị thức dễ dàng được xác định. Giá trị trung bình của N_e được xác định bởi

$$E\{N_e\} = NP_e \tag{6.32}$$

và phương sai của N_e được xác định bởi

$$\sigma_{N_e}^2 = NP_e(1-P_e) \tag{6.33}$$

Sử dụng các kết quả này vào (6.29), giá trị trung bình và phương sai của bộ ước tính Monte-Carlo cho xác suất lỗi sẽ là

$$E\{\hat{P}_e\} = \frac{E\{N_e\}}{N} \tag{6.34}$$

Thay (6.32) vào (6.34) ta được

$$E\{\hat{P}_e\} = \frac{NP_e}{N} = P_e \tag{6.35}$$

cái cho thấy rằng bộ ước tính Monte-Carlo cho xác suất lỗi là không chệch (unbiased). Phương sai của bộ ước tính Monte-Carlo cho xác suất lỗi là

$$\sigma_{\hat{P}_e}^2 = \frac{\sigma_{N_e}^2}{N^2} \tag{6.36}$$

Thay (6.33) vào (6.36) ta có

$$\sigma_{\hat{P}_e}^2 = \frac{P_e(1-P_e)}{N} \tag{6.37}$$

cái cũng cho thấy bộ ước tính là phù hợp vì phương sai giảm dần hay $\hat{P}_e \to P_e$ khi $N \to \infty$.

Trong thực tế số lượng ký hiệu truyền đi N trong mô phỏng là hữu hạn và một câu hỏi đặt ra là N sẽ là bao nhiêu. Nói cách khác, trong mô phỏng Monte-Carlo sẽ cần thực hiện bao nhiêu vòng lặp là đủ. Để xác định được câu trả lời cho câu hỏi trên cần xem xét bài toán khoảng tin cậy hay độ tin cậy của bộ ước tính $\hat{P}_e$. Khoảng tin cậy cho xác suất (1-α) để các ước tính nằm trong một phạm vi giá trị xác định $\left(\pm \beta\sigma_{\hat{P}_e}\right)$. Do đó có hai tham số cần quan tâm đó là: xác suất được xác định bởi α, và phạm vi giá trị được xác định bởi β. Biểu diễn ở dạng biểu thức, khoảng tin cậy được định nghĩa bởi

$$\Pr\left\{P_e - \beta\sigma_{\hat{P}_e} \le \hat{P}_e \le P_e + \beta\sigma_{\hat{P}_e}\right\} = 1-\alpha = p \tag{6.38}$$

như mô tả trong hình 6-5. Phương trình (6.38) có thể được viết lại theo sai số $\hat{P}_e - P_e$ như

$$\Pr\left\{-\beta\sigma_{\hat{P}_e} \le \hat{P}_e - P_e \le +\beta\sigma_{\hat{P}_e}\right\} = 1-\alpha \tag{6.39}$$

Giả sử sai số $\hat{P}_e - P_e$ là một biến ngẫu nhiên Gauss đối với N lớn, hàm mật độ xác suất của $\hat{P}_e - P_e$ gần đúng bởi

$$\frac{1}{\sqrt{2\pi}\sigma_{\hat{P}_e}} \exp\left(-\frac{\left(\hat{P}_e - P_e\right)^2}{2\sigma_{\hat{P}_e}^2}\right)$$

Chú ý rằng $\hat{P}_e - P_e$ là một biến ngẫu nhiên có trung bình bằng không vì ước tính không bị chệch. Do vậy ta có

$$\Pr\left\{\hat{P}_e - P_e \ge \beta\sigma_{\hat{P}_e}\right\} = \frac{1}{\sqrt{2\pi}\sigma_{\hat{P}_e}} \int_{\beta\sigma_{\hat{P}_e}}^{\infty} \exp\left(-\frac{t^2}{2\sigma_{\hat{P}_e}^2}\right) dt \tag{6.40}$$

Bằng việc thay đổi biến $y = t/\sigma_{\hat{P}_e}$ ta có

$$\Pr\left\{\hat{P}_e - P_e \ge \beta\sigma_{\hat{P}_e}\right\} = \frac{1}{\sqrt{2\pi}} \int_{\beta}^{\infty} \exp\left(-\frac{y^2}{2}\right) dy = Q(\beta) \tag{6.41}$$

trong đó $Q(.)$ được gọi là hàm Q theo phân bố Gauss. Từ hình 6-5 ta có

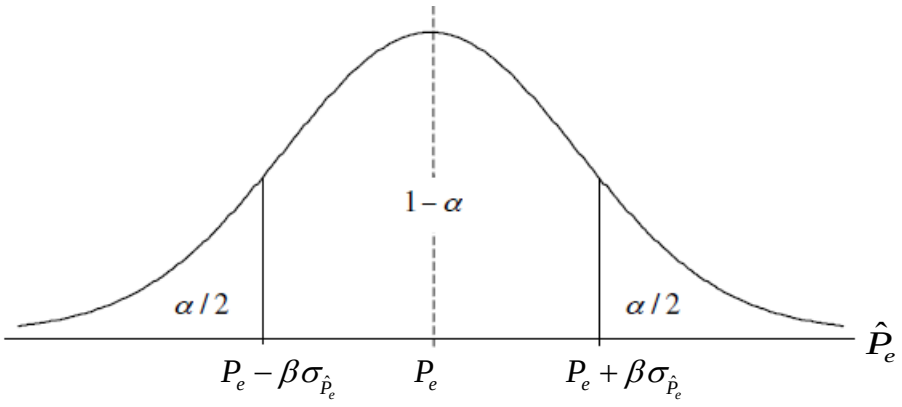
$$\Pr\left\{\hat{P}_e - P_e \geq \beta\sigma_{\hat{P}_e}\right\} = Q(\beta) = \frac{\alpha}{2} \tag{6.42}$$

do đó

$$\beta = Q^{-1}\left(\frac{\alpha}{2}\right) = Q^{-1}\left(\frac{1-p}{2}\right) \tag{6.43}$$

Ví dụ với xác suất yêu cầu p = 95%, từ (6.43) sẽ xác định được β = 1,96. Trong MATLAB các hàm Q và Q đảo có thể được xây dựng qua sử dụng các hàm lỗi `erfc` và `erfinv` như đoạn mã dưới đây.

```
% Ví dụ tính hàm Q và Q đảo
% Hàm Q
function y=Q(x)
y=0.5*erfc(x/sqrt(2)); % complementary error function

% Ví dụ sử dụng hàm Q
>> Q(1)
ans =
    0.1587

% Hàm Q đảo
function y=Qinv(x)
y=sqrt(2)*erfinv(1-2*x); % inverse error function

% Ví dụ sử dụng hàm Qinv
>> Qinv(0.025)
ans =
    1.960
```



Hình 6-5 Khoảng tin cậy của ước tính

Như vậy đối với mô phỏng Monte-Carlo, tiêu chuẩn dừng vòng lặp có thể được xác định từ độ tin cậy (xác suất p) mong muốn, khoảng tin cậy chấp nhận được (tham số β) và tốc độ lỗi kỳ vọng P_e. Từ (6.37) và (6.43) ta có thể xác định được tổng số ký hiệu cần truyền trong mô phỏng đó là

$$N = P_e(1-P_e)\left(\frac{\beta}{\beta\sigma_{\hat{P}_e}}\right)^2 \tag{6.44}$$

Ví dụ đối với tốc độ lỗi kỳ vọng P_e = 10^{-3}, p = 95% và khoảng tin cậy $\beta\sigma_{\hat{P}_e}$ = 10^{-4}, từ (6.44) điều kiện dừng mô phỏng Monte-Carlo được xác định $N \approx 400000$. Tuy nhiên trong các mô phỏng các hệ thống truyền thông, tốc độ lỗi có thể thường rất nhỏ, do đó điều đáng mong muốn là xem khoảng tin cậy là một phần nhỏ của tốc độ lỗi tức $\beta\sigma_{\hat{P}_e} = \gamma P_e$ với γ = 0,1 cho 10% sai số ước tính là có thể chấp nhận được. Vì vậy thay bằng xác định tổng số ký hiệu N cần truyền thì tổng số lỗi đếm được N_e được xác định như là điều kiện dừng mô phỏng. Từ (6.44) tổng số lỗi kỳ vọng xác định được là

$$N_e = N.P_e = (1-P_e)\left(\frac{\beta}{\gamma}\right)^2 \approx \left(\frac{\beta}{\gamma}\right)^2 \tag{6.45}$$

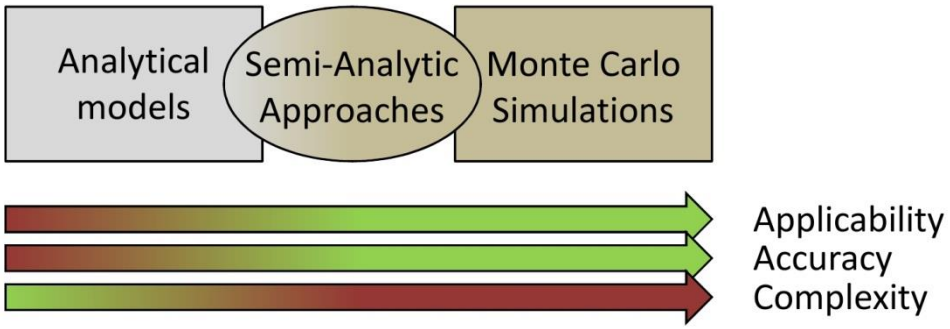
Ví dụ với p = 95% và γ = 0,1 thì mô phỏng sẽ dừng khi số lượng lỗi đếm được đạt đến khoảng 400 lỗi.

Một cấu trúc chương trình mô phỏng Monte-Carlo trong MATLAB được cho trong đoạn mã dưới đây

```
% Ví dụ cấu trúc chương trình mô phỏng Monte-Carlo
% Thiết lập tham số cho mô phỏng;
...
EsOverN0dB = 0:0.5:9; % Thay đổi SNR từ 0 đến 9dB
MaxSymbols = 1e6;     % Số symbol cực đại
...
% Xác định tham số dừng vòng lặp
...
MinErrors  = (betaValue/gammaValue)^2; % Xác định số lỗi cần đếm
...
% Các biến vòng lặp
NumErrors  = zeros( size( EsOverN0dB ) );
NumSymbols = zeros( size( EsOverN0dB ) );

% Vòng lặp ngoài
for kk = 1:length( EsOverN0dB )
    ...
    Done = false; % Khởi tạo lại điều kiện dừng cho vòng lặp trong
     % Vòng lặp trong
    while ( ~Done )
        ...
      % Tính điều kiện dừng
       Done = NumErrors(kk)>MinErrors||NumSymbols(kk)>MaxSymbols;
    end
     % Tính tốc độ lỗi
      ...
end
```

6.3.2 Phương pháp bán giải tích

Đối với phương pháp Monte-Carlo, tải tính toán có thể rất lớn khi tốc độ lỗi kỳ vọng nhỏ. Ví dụ tại tốc độ lỗi cỡ 10^{-8} đòi hỏi ít nhất 10^9 – 10^{10} ký hiệu được truyền đi. Trong khi phương pháp giải tích khó đánh giá tốc độ lỗi chính xác đối với những hệ thống phức tạp. Do vậy phương pháp bán giải tích có thể được áp dụng để khắc phục giới hạn của các phương pháp Monte-Carlo và giải tích. Trong phương pháp bán giải tích, kiến thức của hệ thống đang phân tích được sử dụng để giảm tải tính toán và mức độ phức tạp mà mô phỏng Monte-Carlo đòi hỏi.



Hình 6-6 Các phương pháp khác nhau để đánh giá hiệu năng hệ thống.

Ý tưởng thực hiện phương pháp bán giải tích đó là: Thực hiện phát một chuỗi ký hiệu ngẫu nhiên có độ dài L^m (L là bậc điều chế) qua hệ thống mô phỏng, tiếp sau xác suất lỗi trung bình được tính dựa trên phương pháp giải tích thông qua đánh giá hàm mật độ xác suất của mẫu thu được tại bộ quyết định.

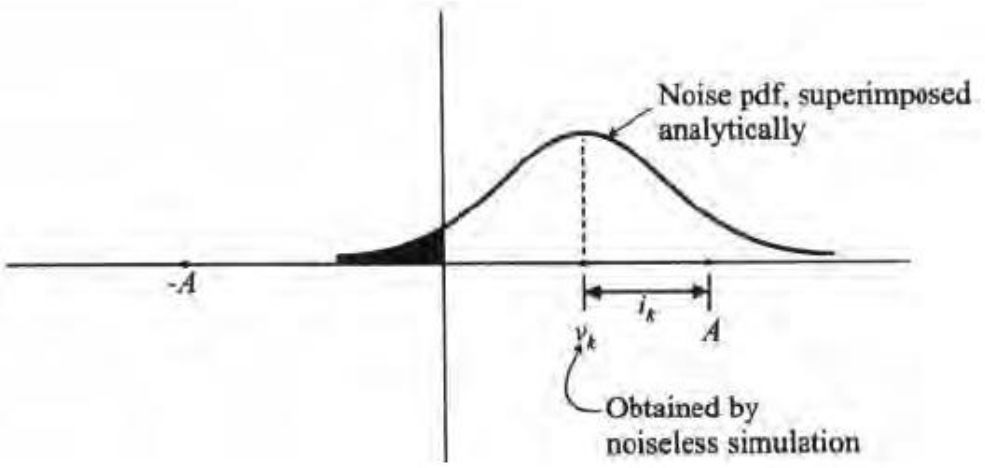
Xét trường hợp đối với hệ thống nhị phân đơn giản như mô tả trong hình 6-7. Trong hệ thống này các điểm thu được lý tưởng là $\pm A$, trong khi sự méo dạng sẽ tự biểu hiện trong các mẫu biên độ quanh những điểm này như cho thấy bởi mẫu ví dụ tại $v_k = (1 - \varepsilon_k)A$, trong đó ε_k là tỉ phần lỗi cho mẫu thứ k. Đối với mẫu đó, xác suất lỗi có điều kiện được xác định bởi

$$p_k = \int_{-\infty}^{-v_k} f_n(\eta)d\eta = F_n(-v_k) \tag{6.45}$$

trong đó f_n là hàm xác suất một chiều và F_n là hàm phân bố tích lũy. Đối với nguồn nhiễu Gauss, $p_k = \frac{1}{2}erfc\left(v_k/\sqrt{2}\sigma\right)$, trong đó $v_k = (1 - \varepsilon_k)A$, và σ^2 là công suất nhiễu. Nếu một chuỗi N bít được phát đi, BER tổng cộng trung bình chỉ là phép tính trung bình của p_k

$$p = \frac{1}{N}\sum_{k=1}^{N} p_k \tag{6.46}$$

Nói cách khác phương pháp bán giải tích sử dụng công thức tính BER bằng phương pháp giải tích, nhưng các tham số trong công thức được ước tính từ các mẫu thu được tại bộ quyết định sau khi chạy mô phỏng.



Hình 6-7 Mô tả phương pháp bán giải tích cho hệ thống nhị phân:

Một đoạn mã MATLAB dưới đây cho ví dụ về ước tính BER dựa trên phương pháp bán giải tích:

```
% Ví dụ ước tính BER dựa trên phương pháp bán giải tích
% xx – vector tín hiệu thu sau mô phỏng
% nbw – độ rộng băng tần nhiễu
% EsOverN0dB – Tỉ số tín hiệu trên nhiễu theo dB
…
nx = length(xx);  % Độ dài chuỗi ký hiệu đầu vào bộ thu;

% Xác định phương sai nhiễu.
EsOverN0 = 10^(EsOverN0dB/10); % dB to linear
N0 = Es/EsOverN0; % Công suất nhiễu
sigma = sqrt(N0*nbw*2); % Phương sai nhiễu
d = abs(xx);

pe = sum(Q(d/sigma));   % Ước tính xác suất lỗi
pe = pe/nx;
```

6.3.3 Các phương pháp khác

Để giảm thời gian ước tính hiệu năng của hệ thống, một số phương pháp khác cũng có thể được sử dụng.

Phương pháp lấy mẫu quan trọng (IS):

- Kỹ thuật để ước tính các tính chất của một phân bố xác định trong khi chỉ có các mẫu từ một dạng phân bố khác.
- Thực hiện lấy mẫu các quá trình: các quá trình này đặc trưng cho các quá trình khác nhau ảnh hưởng lên tín hiệu

Phương pháp ngoại suy đuôi phân bố:

- Kỹ thuật cho phép mở rộng sự quan tâm
- Thực hiện xác định tốc độ lỗi tại chế độ lỗi cao: các ước tính có độ tin cậy cao hơn
- Mở rộng các ước tính để dự đoán tốc độ lỗi tại những điều kiện SNR mà ở đó tốc độ lỗi là nhỏ

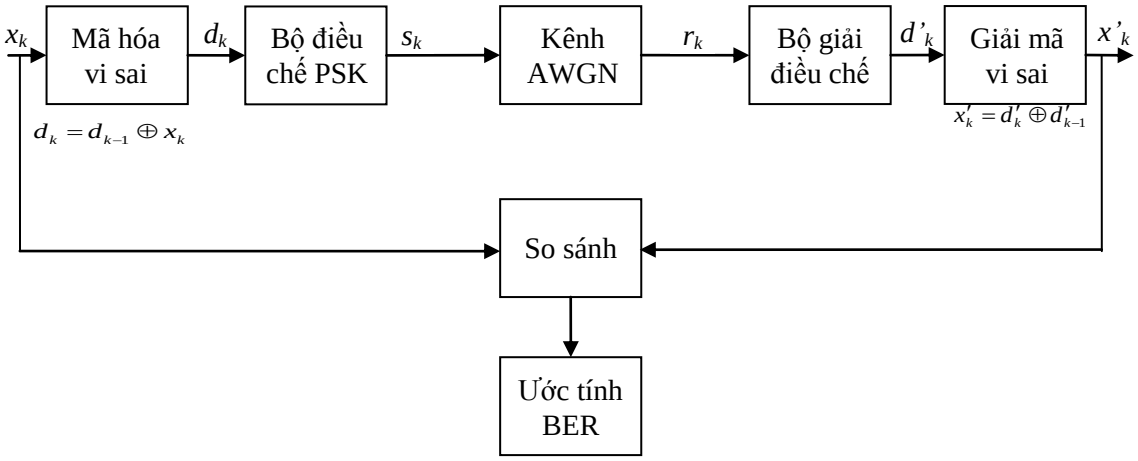
Tuy nhiên việc cải thiện tốc độ chạy mô phỏng của các phương pháp này cũng có thể làm cho sai số kết quả đầu ra tăng lên. Do vậy cần chú ý khi áp dụng và có sự đánh giá phương pháp bằng việc so sánh kiểm chứng với phương pháp Monte-Carlo.

6.3.4 Một số ví dụ mô phỏng hệ thống viễn thông

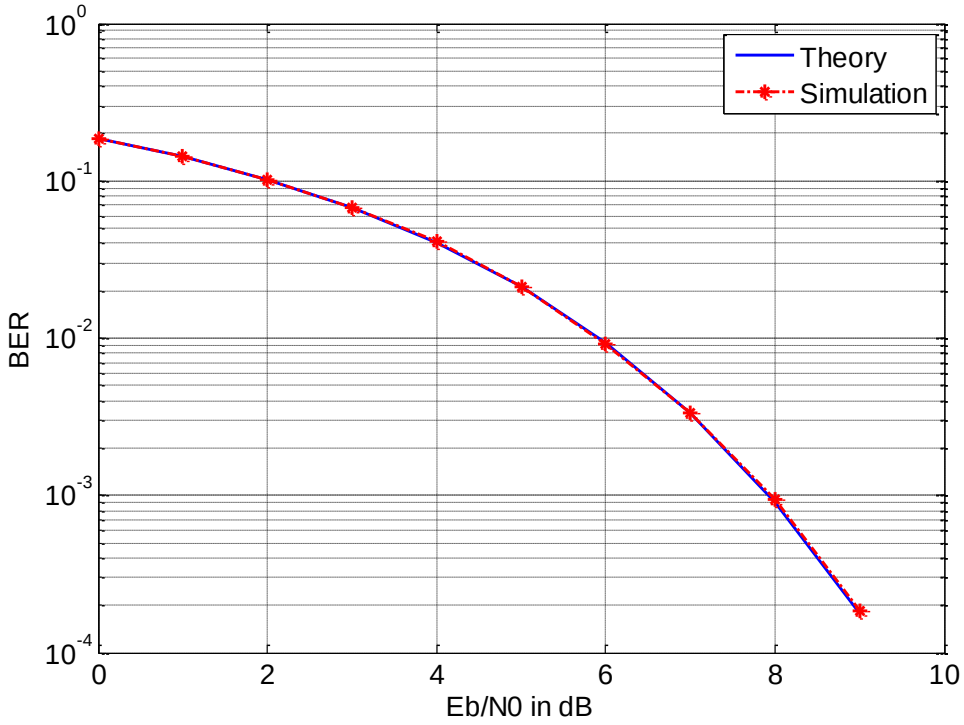
Ví dụ 6.1: Mô phỏng hệ thống DPSK qua kênh AWGN

Một mô hình mô phỏng hệ thống đơn giản sử dụng điều chế DPSK được cho thấy trong hình 6-8 ở mức ký hiệu, hiệu năng hệ thống được ước tính bằng phương pháp Monte Carlo. Chương trình MATLAB mô phỏng hệ thống đơn giản này được cho dưới đây và kết quả được cho thấy trong hình 6-9.

```
% Ví dụ mô phỏng hệ thống DPSK qua kênh AWGN
% Tạo bản tin tín hiệu số cho điều chế DPSK
M = 2;
N = 1000000;
x = randint(1,N,M);
% Điều chế DPSK
s = dpskmod(x,M);
% Truyền qua kenh AWGN
SNRdB = 0:9;
SNR = 10.^(SNRdB/10);
ber = [];
Pb = [];
for k = 1:10
    r = awgnchan(s,SNRdB(k));
    xr = dpskdemod(r,M);  % Giai dieu che DPSK
    % Xac dinh loi
    e = (x ~=xr);
    ne = sum(e);
    ber(k) = ne/N;  % Xác suất lỗi ước tính
    % Xác suất lỗi lý thuyết
    Pb(k) = 1/2*exp(-SNR(k));
end
% Hiển thị kết quả
figure(1); % BER curve
semilogy(SNRdB,Pb,'-',SNRdB,ber,'r*-.');grid;
xlabel('Eb/N0 in dB');ylabel('BER');
legend('Theory','Simulation');
```



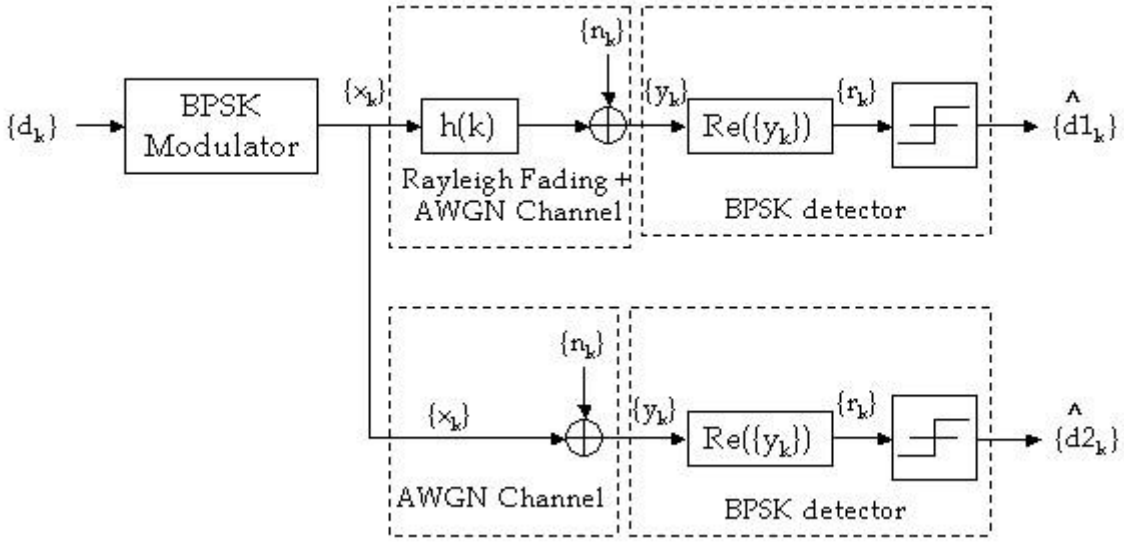
Hình 6-8 Mô hình mô phỏng hệ thống DPSK qua kênh AWGN



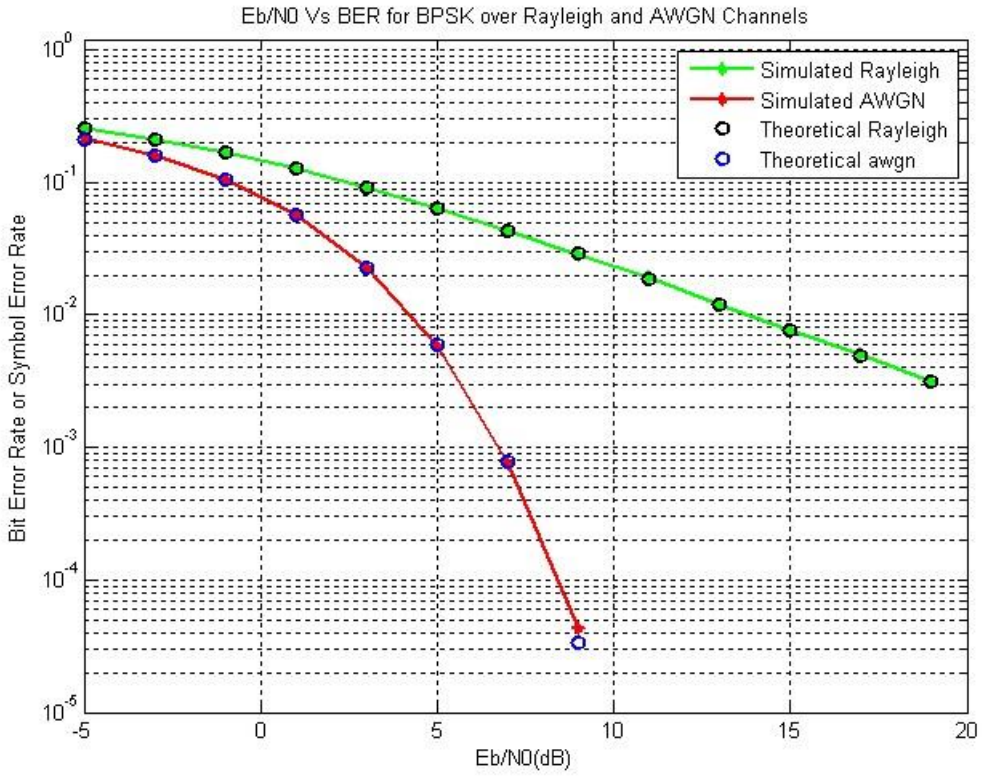
Hình 6-9 Kết quả BER là hàm SNR của hệ thống DPSK truyền qua kênh AWGN

Ví dụ 6.2: Mô phỏng hệ thống BPSK qua kênh AWGN và kênh pha đinh Rayleigh

Mô hình mô phỏng hệ thống được cho thấy trong hình 6-10 trong đó hệ thống sử dụng điều chế BPSK và được truyền trên hai kênh AWGN và kênh pha đinh Rayleigh để so sánh hiệu năng. Kết quả mô phỏng được cho thấy trên hình 6-11.



Hình 6-10 Mô hình mô phỏng hệ thống



Hình 6-11 Kết quả mô phỏng hệ thống BPSK

6.4 Tổng kết chương

Nội dung chương này đã đề cập đến các kỹ thuật ước tính các tham số bao gồm các tham số của tín hiệu, tỉ số tín hiệu trên nhiễu và tham số hiệu năng của hệ thống. Trong các kỹ thuật ước tính tham số hiệu năng qua xác suất lỗi, phương pháp Monte Carlo đã được trình bày cụ thể từ phương pháp luận đến cách thức thực hiện. Ngoài ra phương pháp bán giải tích cũng được giới thiệu như một cách để giúp ước tính xác suất lỗi nhanh hơn. Các phương pháp khác như lấy mẫu quan trọng và ngoại suy đuôi cũng giới thiệu để gợi mở cho người học tìm tòi thêm. Một vài ví dụ mô phỏng cũng đã được đưa ra để minh họa nội dung của chương.

Câu hỏi/bài tập chương 6

6-1/ Hãy tính số lượng ký hiệu cần truyền trong mô phỏng Monte Carlo khi ước tính xác suất lỗi 10^{-3} đảm bảo khoảng tin cậy là 10^{-4} với xác suất 90%.

6-2/ Mô phỏng hiệu năng hệ thống điều chế DPSK như cho trong ví dụ 6.1 bằng phương pháp Monte-Carlo với dạng xung điều chế là xung vuông và có sử dụng bộ lọc phối hợp tại bộ thu.

6-3/ Mô phỏng hiệu năng hệ thống điều chế DPSK như cho trong ví dụ 6.1 bằng phương pháp Monte-Carlo với dạng xung điều chế là xung raised cosine và có sử dụng bộ lọc phối hợp tại bộ thu.

6-4/ Mô phỏng hệ thống điều chế DPSK như cho trong ví dụ 6.1 và đánh giá hiệu năng bằng phương pháp bán giải tích với dạng xung điều chế là xung vuông và có sử dụng bộ lọc phối hợp tại bộ thu.

6-5/ Mô phỏng hiệu năng hệ thống điều chế BPSK truyền qua kênh pha đinh đa đường có đặc tính như cho trong bài tập 5-7 bằng phương pháp Monte-Carlo với dạng xung điều chế là xung vuông và có sử dụng bộ lọc phối hợp tại bộ thu.

Tài liệu tham khảo

[1] Michel C. Jeruchim, Philip Balaban, *Simulation of Communication Systems: Modeling, Methodology and Techniques*, 2nd ed., Kluwer Academic/Plenum Publishers, 2000.

[2] Nguyễn Viết Đảm, *Mô phỏng hệ thống viễn thông và ứng dụng MATLAB*, NXB Bưu Điện, 2007.

[3] J. G. Proakis, M. Salehi, G. Bauch, *Contemporary Communication Systems Using MATLAB and Simulink*, 3rd ed., Cengage Learning, 2012.

[4] O. Beucher, M. Weeks, *Introduction to MATLAB and Simulink: A Project Approach*, 3rd ed., Infinity Science Press, 2008.

[5] Mathworks Inc., *MATLAB and Simulink Student Version: Getting Started With MATLAB*, 2007.

[6] Steven C. Chapra, R. P. Canale, *Numerical Methods for Engineers*, 6th ed., Mcgraw-Hill, 2010.

[7] Dennis Silages, *Digital Communication Systems using MATLAB and Simulink*, Bookstand Publishing, 2009.

[8] K. C. Raveendranathan, *Communication Systems Modeling and Simulation using MATLAB and Simulink*, Universities Press, 2011.

[9] Mohsen Guizani, Ammar Rayes, Bilal Khan, Ala Al-Fuqaha, *Network Modeling and Simulation: A Practical Perspective*, Wiley, 2010.

Phụ lục A

Một số thư viện hàm xử lý và tính toán trong MATLAB cho mô phỏng hệ thống truyền thông

A.1 Thư viện Communications Toolbox

Kênh truyền

`awgn`	Kênh AWGN
`bsc`	Mô hình kênh đối xứng nhị phân
`Doppler`	Gói các lớp Doppler
`filter(channel)`	Lọc tín hiệu theo đối tượng kênh
`mimochan`	Tạo đối tượng kênh pha đinh MIMO
`rayleighchan`	Xây dựng đối tượng kênh pha đinh Rayleigh
`ricianchan`	Xây dựng đối tượng kênh pha đinh Rice
`stdchan`	Xây dựng đối tượng kênh từ tập các mô hình được tiêu chuẩn hóa

Nguồn tín hiệu

`commsrc.pattern`	Xây dựng đối tượng bộ tạo mẫu tín hiệu
`randerr`	Tạo các mẫu lỗi bit
`randint`	Tạo ma trận các số nguyên ngẫu nhiên phân bố đều
`randsrc`	Tạo ma trận ngẫu nhiên sử dụng bảng ký tự được quy định
`wgn`	Tạo nhiễu Gauss trắng

Mã hóa nguồn

`arithdeco`	Giải mã nhị phân sử dụng giải mã số học
`arithenco`	Mã hóa chuỗi ký hiệu sử dụng mã hóa số học
`compand`	Bộ nén/giãn luật μ hoặc luật A mã nguồn
`dpcmdeco`	Giải mã sử dụng điều chế mã xung vi sai
`dpcmenco`	Mã hóa sử dụng điều chế mã xung vi sai
`dpcmopt`	Tối ưu hóa các tham số điều chế mã xung vi sai
`huffmandeco`	Giải mã Huffman
`huffmandict`	Tạo từ điển mã Huffman cho nguồn bằng mô hình xác suất đã biết
`huffmanenco`	Mã hóa Huffman

`Lloyds`	Tối ưu hóa các tham số lượng tử hóa sử dụng thuật toán Lloyd
`quantize`	Tạo chỉ số lượng tử hóa và giá trị đầu ra được lượng tử hóa

Mã sửa lỗi

`bchdec`	Bộ giải mã BCH
`bchenc`	Bộ mã hóa BCH
`bchgenpoly`	Bộ tạo đa thức tạo mã BCH
`bchnumerr`	Số lỗi có thể sửa của mã BCH
`convenc`	Mã chập dữ liệu nhị phân
`cyclgen`	Tạo ma trận sinh và kiểm tra chẵn lẻ của mã cyclic
`cyclpoly`	Tạo đa thức tạo mã của mã cyclic
`decode`	Bộ giải mã khối
`dvbs2ldpc`	Các mã LDPC theo tiêu chuẩn DVB-S.2
`encode`	Bộ mã hóa khối
`fec.bchdec`	Xây dựng đối tượng giải mã BCH
`fec.bchenc`	Xây dựng đối tượng mã hóa BCH
`fec.ldpcdec`	Xây dựng đối tượng giải mã LDPC
`fec.ldpcenc`	Xây dựng đối tượng mã hóa LDPC
`fec.rsdec`	Xây dựng đối tượng giải mã Reed-Solomon
`fec.rsenc`	Xây dựng đối tượng mã hóa Reed-Solomon
`gen2par`	Chuyển đổi giữa ma trận sinh và kiểm tra chẵn lẻ
`gfweight`	Tính khoảng cách tối thiểu của mã khối tuyến tính
`hammgen`	Tạo các ma trận sinh và kiểm tra chẵn lẻ của mã Hamming
`rsdec`	Bộ giải mã Reed-Solomon
`rsdecof`	Giải mã tệp ASCII được mã hóa sử dụng mã Reed-Solomon
`rsenc`	Bộ mã hóa Reed-Solomon
`rsgenpoly`	Tạo đa thức tạo mã của mã Reed-Solomon
`syndtable`	Tạo bảng syndrome giải mã
`vitdec`	Giải mã chập dữ liệu nhị phân sử dụng thuật toán Viterbi

Điều chế/giải điều chế tương tự

`amdemod`	Giải điều chế AM
`ammod`	Điều chế AM
`fmdemod`	Giải điều chế FM
`fmmod`	Điều chế FM
`pmdemod`	Giải điều chế PM
`pmmod`	Điều chế PM
`ssbdemod`	Giải điều chế SSB
`ssbmod`	Điều chế SSB

Điều chế/giải điều chế số

`dpskdemod`	Giải điều chế DPSK
`dpskmod`	Điều chế DPSK
`fskdemod`	Giải điều chế FSK
`fskmod`	Điều chế FSK
`genqamdemod`	Giải điều chế QAM tổng quát
`genqammod`	Điều chế QAM tổng quát
`modem`	Gói các lớp modem
`modnorm`	Hệ số định cỡ cho chuẩn hóa đầu ra điều chế
`mskdemod`	Giải điều chế MSK
`mskmod`	Điều chế MSK
`oqpskdemod`	Giải điều chế QPSK bù
`oqpskmod`	Điều chế QPSK bù
`pamdemod`	Giải điều chế xung biên
`pammod`	Điều chế xung biên
`pskdemod`	Giải điều chế PSK
`pskmod`	Điều chế PSK
`qamdemod`	Giải điều chế QAM
`qammod`	Điều chế QAM

Đánh giá hiệu năng

`berawgn`	BER cho các kênh AWGN không sử dụng mã
`bercoding`	BER cho các kênh AWGN mã hóa
`berconfint`	BER và khoảng tin cậy của mô phỏng Monte-Carlo
`berfading`	BER cho các kênh pha đinh Rayleigh và Rice
`berfit`	Làm khớp đường cong đối với dữ liệu BER thực nghiệm không phẳng
`bersync`	BER cho quá trình đồng bộ không hoàn hảo
`biterr`	Tính số lượng lỗi bít và BER
`commeasure.EVM`	Tạo đối tượng đo EVM
`commeasure.MER`	Tạo đối tượng đo MER
`commscope`	Gói các lớp hiển thị thông tin
`commscope.eyediagram`	Phân tích mẫu mắt
`commscope.ScatterPlot`	Tạo hiển thị biểu đồ chòm sao
`distspec`	Tính phổ khoảng cách của mã chập
`eyediagram`	Tạo biểu đồ mắt
`EyeScope`	Tạo hiển thị biểu đồ mắt cho đối tượng mẫu mắt H
`noisebw`	Độ rộng băng tần nhiễu tương đương của bộ lọc

`scatterplot`	Tạo biểu đồ chòm sao
`semianalytic`	Tính BER sử dụng kỹ thuật bán giải tích
`symerr`	Tính số lượng lỗi ký hiệu và SER

Các bộ lọc

`intdump`	Lấy tích phân và khởi tạo (I&D)
`hank2sys`	Chuyển đổi ma trận Hankel thành mô hình hệ thống tuyến tính
`hilbiir`	Thiết kế bộ lọc IIR khai triển Hilbert
`rcosine`	Thiết kế bộ lọc raised cosine
`rcosfir`	Thiết kế bộ lọc FIR raised cosine
`rcosiir`	Thiết kế bộ lọc IIR raised cosine
`rectpulse`	Tạo dạng xung vuông

Bộ cân bằng

`cma`	Xây dựng đối tượng giải thuật modul hằng số
`dfe`	Xây dựng đối tượng bộ cân bằng hồi tiếp quyết định (DFE)
`equalize`	Cân bằng tín hiệu sử dụng đối tượng bộ cân bằng
`lineareq`	Xây dựng đối tượng bộ cân bằng tuyến tính
`lms`	Xây dựng đối tượng giải thuật thích ứng trung bình bình phương tối thiểu (LMS)
`mlseeq`	Cân bằng tín hiệu điều chế tuyến tính sử dụng thuật toán Viterbi
`normlms`	Xây dựng đối tượng giải thuật thích ứng LMS chuẩn hóa
`reset(equalizer)`	Thiết lập lại đối tượng bộ cân bằng
`rls`	Xây dựng đối tượng giải thuật thích ứng bình phương tối thiểu đệ quy (RLS)
`signlms`	Xây dựng đối tượng giải thuật thích ứng LMS có dấu
`varlms`	Xây dựng đối tượng giải thuật thích ứng LMS cỡ bước biến đổi

Trường Galois

`convmtx`	Ma trận tích chập của vector trường Galois
`cosets`	Tạo các coset cyclotomic của trường Galois
`dftmtx`	Ma trận khai triển Fourier rời rạc trong trường Galois
`filter(gf)`	Bộ lọc số 1-D trên trường Galois
`gf`	Tạo mảng trường Galois
`gftable`	Sinh tệp để gia tốc tính toán trường Galois
`minpol`	Tìm đa thức nhỏ nhất của phần tử trường Galois
`mldivide`	Phép chia trái ma trận \ của các mảng Galois
`primpoly`	Tìm các đa thức nguyên của trường Galois

Interleaving/Deinterleaving

`convdeintrlv`	Khôi phục trật tự ký hiệu sử dụng các thanh ghi dịch
`convintrlv`	Giao hoán các ký hiệu sử dụng các thanh ghi dịch
`deintrlv`	Khôi phục trật tự các ký hiệu
`intrlv`	Sắp xếp lại chuỗi ký hiệu
`randdeintrlv`	Khôi phục trật tự các ký hiệu sử dụng giao hoán ngẫu nhiên
`randintrlv`	Sắp xếp lại các ký hiệu sử dụng giao hoán ngẫu nhiên

Một số hàm tiện ích

`alignsignals`	Đồng chỉnh hai tín hiệu bởi trễ tín hiệu trước
`bi2de`	Chuyển đổi vector nhị phân thành các số thập phân
`bin2gray`	Chuyển đổi các số nguyên dương thành các số nguyên mã Gray tương ứng
`de2bi`	Chuyển đổi các số thập phân thành các vector nhị phân
`finddelay`	Ước tính trễ giữa các tín hiệu
`gray2bin`	Chuyển đổi các số nguyên mã Gray thành các số nguyên dương tương ứng
`marcumq`	Hàm Q Marcum tổng quát
`oct2dec`	Chuyển đổi các số bát phân sang thập phân
`poly2trellis`	Chuyển đổi các đa thức mã chập sang mô tả dạng lưới
`qfunc`	Hàm Q
`qfuncinv`	Hàm Q đảo
`seqgen`	Gói tạo chuỗi
`seqgen.pn`	Xây dựng đối tượng tạo chuỗi giả ngẫu nhiên
`vec2mat`	Chuyển đổi vector thành ma trận
`erf`	Hàm lỗi
`erfc`	Hàm lỗi bù
`erfinv`	Hàm lỗi đảo
`erfcinv`	Hàm lỗi bù đảo

A.2 Thư viện Signal Processing Toolbox

Các hàm xử lý tín hiệu cơ bản

`conv`	Tích chập và nhân đa thức
`conv2`	Tích chập 2-D
`corrcoef`	Các hệ số tương quan
`cov`	Ma trận tương quan
`cplxpair`	Sắp xếp các số phức thành các cặp liên hợp phức

`deconv`	Khử tích chập và chia đa thức
`fft`	Khai triển Fourier rời rạc
`fft2`	Khai triển Fourier rời rạc 2-D
`fftshift`	Dịch thành phần tần số 0 về trung tâm phổ
`filter2`	Bộ lọc số 2-D
`freqspace`	Khoảng cách tần số trong đáp ứng tần
`ifft`	Khai triển Fourier rời rạc ngược
`ifft2`	Khai triển Fourier rời rạc ngược 2-D
`unwrap`	Hiệu chỉnh pha để tạo ra đồ thị pha bằng phẳng

Khai triển

`bitrevorder`	Giao hoán dữ liệu thành trật tự đảo bit
`czt`	Khai triển z chirp
`dct`	Khai triển cosine rời rạc
`dftmtx`	Ma trận khai triển Fourier rời rạc
`digitrevorder`	Giao hoán đầu vào thành trật tự đảo số
`fwht`	Khai triển Walsh-Hadamard nhanh
`goertzel`	Khai triển Fourier rời rạc sử dụng thuật toán Goertzel bậc hai
`hilbert`	Tín hiệu giải tích rời rạc thời gian sử dụng khai triển Hilbert
`idct`	Khai triển cosine rời rạc ngược
`ifwht`	Khai triển Walsh-Hadamard nhanh ngược

Xử lý tín hiệu thống kê

`corrmtx`	Ma trận dữ liệu cho ước tính ma trận tự tương quan
`cpsd`	Mật độ phổ công suất chéo
`dspdata`	Thông tin tham số dữ liệu DSP
`mscohere`	Tính kết hợp bình phương biên độ
`pburg`	PSD sử dụng phương pháp Burg
`pcov`	PSD sử dụng phương pháp đồng phương
`peig`	Giả phổ sử dụng phương pháp vector trị riêng
`periodogram`	PSD sử dụng phương pháp periodogram
`pmcov`	PSD sử dụng phương pháp đồng phương biến đổi
`pmtm`	PSD sử dụng phương pháp đa nón
`pmusic`	Giả phổ sử dụng thuật toán MUSIC
`pwelch`	PSD sử dụng phương pháp Welch
`pyulear`	PSD sử dụng phương pháp AR Yule-Walker
`spectrogram`	Vẽ và tính phổ sử dụng khai triển Fourier thời gian ngắn
`spectrum`	Ước tính phổ
`tfestimate`	Ước tính hàm truyền

xcorr	Tương quan chéo
xcorr2	Tương quan chéo 2-D
xcov	Đồng phương chéo

Xử lý tín hiệu đa tốc độ

decimate	Giảm tốc độ mẫu
downsample	Giảm tốc độ mẫu bởi hệ số nguyên
interp	Nội suy – Tăng tốc độ mẫu bởi hệ số nguyên
resample	Thay đổi tốc độ mẫu bởi một hệ số tỉ lệ
upfirdn	Tăng mẫu áp dụng bộ lọc FIR và giảm mẫu
upsample	Tăng tốc độ mẫu bởi hệ số nguyên

Tạo dạng sóng

chirp	Hàm cosine tần số quét
diric	Hàm sinc tuần hoàn hoặc Dirichlet
gauspuls	Xung hình sin được điều chế dạng Gauss
gmonopuls	Đơn xung Gauss
pulstran	Tạo chuỗi xung
rectpuls	Xung vuông không tuần hoàn được lấy mẫu
sawtooth	Sóng tam giác hoặc răng cưa
sinc	Hàm sinc
square	Sóng bình phương
tripuls	Xung tam giác không tuần hoàn được lấy mẫu
vco	Bộ dao động điều khiển bằng điện áp

Các bộ lọc số

cfirpm	Thiết kế bộ lọc FIR pha phi tuyến và phức
fir1	Thiết kế bộ lọc FIR dựa trên cửa số
fir2	Thiết kế bộ lọc FIR dựa trên lấy mẫu tần số
firls	Thiết kế bộ lọc FIR pha tuyến tính bình phương tối thiểu
intfilt	Thiết kế bộ lọc FIR nội suy
firrcos	Thiết kế bộ lọc FIR raised cosine
gaussfir	Bộ lọc tạo dạng xung FIR Gauss
butter	Thiết kế bộ lọc Butterworth
cheby1	Thiết kế bộ lọc Chebyshev kiểu 1
cheby2	Thiết kế bộ lọc Chebyshev kiểu 2
freqz	Đáp ứng tần của bộ lọc số
fvtool	Công cụ trực quan bộ lọc
grpdelay	Độ trễ bộ lọc trung bình (độ trễ nhóm)

`impz`	Đáp ứng xung của bộ lọc số
`phasedelay`	Trễ pha của bộ lọc số
`phasez`	Đáp ứng pha của bộ lọc số
`stepz`	Đáp ứng bậc của bộ lọc số
`zerophase`	Đáp ứng pha zero của bộ lọc số
`zplane`	Vẽ đồ thị zero-pole
`fftfilt`	Lọc FIR dựa trên FFT sử dụng phương pháp xếp chồng-cộng
`filter`	Lọc dữ liệu bằng bộ lọc FIR hoặc IIR
`filtfilt`	Lọc số pha zero

Một số hàm đặc biệt

`buffer`	Đệm vector tín hiệu thành ma trận các khung dữ liệu
`db2mag`	Chuyển đổi dB sang biên độ
`dB2pow`	Chuyển đổi dB sang công suất
`eqtflength`	Cân bằng độ dài tử và mẫu số hàm truyền
`findpeaks`	Tìm các cực đại cục bộ
`mag2db`	Chuyển đổi biên độ sang dB
`pow2db`	Chuyển đổi công suất sang dB
`udecode`	Giải mã đầu vào các số nguyên lượng tử hóa 2^n mức thành các giá trị mẫu đầu ra
`uencode`	Lượng tử hóa và mã hóa các mẫu đầu vào thành các mức số nguyên đầu ra