

PGS. TS. Võ Nguyễn Quốc Bảo

MÔ PHỎNG HỆ THỐNG TRUYỀN THÔNG



NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT

LỜI NÓI ĐẦU

Cuốn sách “**Mô phỏng hệ thống truyền thông**” nhằm mục đích cung cấp kiến thức cho sinh viên hiểu và biết được cách thức mô phỏng một hệ thống truyền thông và có thể tiến hành cài đặt trên phần mềm mô phỏng Matlab. Cuốn sách này là giáo trình cho môn học “Mô phỏng các hệ thống truyền thông” tại Khoa Viễn thông, Học viện Công nghệ Bưu chính Viễn thông cơ sở tại TP. Hồ Chí Minh. Cuối các chương có phần câu hỏi và bài tập giúp cho sinh viên và người đọc có thể ôn lại nội dung trọng tâm trong các chương.

Cuốn sách này có thể là tài liệu giảng dạy đại học và sau đại học cho sinh viên chuyên ngành Kỹ thuật và Công nghệ điện tử viễn thông, Truyền thông và Mạng máy tính. Cuốn sách là tài liệu tham khảo không thể thiếu cho sinh viên nghiên cứu về thông tin vô tuyến, đặc biệt ở lớp vật lý. Sách có 6 chương với tổng cộng 170 ví dụ đi kèm giúp minh họa tất cả các khía cạnh của mô phỏng hệ thống truyền thông. Để đọc tốt cuốn sách này, người đọc cần có kiến thức về: lý thuyết xác suất, lý thuyết thông tin, thông tin số và thông tin vô tuyến bên cạnh kiến thức về lập trình cơ bản.

Các ví dụ trong cuốn sách được lập trình và kiểm tra trên Matlab phiên bản 2019a, có thể không hoàn toàn tương thích với các phiên bản Matlab khác. Nếu có vấn đề khó khăn trong thực thi các ví dụ, xin vui lòng liên hệ với tác giả qua email: baovnq@gmail.com.

LỜI CẢM ƠN

Tác giả xin cảm ơn gia đình đã tạo điều kiện và dành thời gian cho tôi hoàn thành cuốn sách này.

Tôi trân trọng cảm ơn Phòng Thí nghiệm Thông tin vô tuyến và Khoa Viễn thông 2 thuộc Học Viện Công Nghệ Bưu chính Viễn thông cơ sở tại TP. Hồ Chí Minh đã hỗ trợ để cuốn sách được ra đời.

Tôi xin được ghi nhận tất cả những thiếu sót (nếu có) trong cuốn sách này và trân trọng cảm ơn những góp ý để cuốn sách được hoàn thiện hơn. Các góp ý và thiếu sót xin vui lòng gửi về tác giả của cuốn sách theo địa chỉ email: baovnq@gmail.com.

Tác giả
Võ Nguyễn Quốc Bảo

MỤC LỤC

LỜI NÓI ĐẦU **2**

LỜI CẢM ƠN..... **4**

CHƯƠNG 1: LÝ THUYẾT MÔ PHỎNG **10**

1.1 **Tổng quan.....** **10**

1.2 **Một số khái niệm.....** **12**

 1.2.1 Mô phỏng máy tính..... **12**

 1.2.2 Mô hình..... **12**

1.3 **Ứng dụng của mô phỏng** **14**

1.4 **Vai trò của mô phỏng** **14**

1.5 **Các phần mềm mô phỏng.....** **14**

 1.5.1 Matlab..... **14**

 1.5.2 Mathematica..... **15**

 1.5.3 Scilab..... **15**

 1.5.4 NS-3..... **16**

 1.5.5 Opnet **16**

1.6 **Bài tập** **17**

CHƯƠNG 2: GIỚI THIỆU MATLAB **18**

2.1 **Giới thiệu tổng quan.....** **18**

 2.1.1 Giao diện bắt đầu..... **18**

 2.1.1.1 Biến **19**

 2.1.1.2 Môi trường làm việc **21**

 2.1.1.3 Cửa sổ lệnh **21**

 2.1.1.4 Phần giúp đỡ **22**

 2.1.1.5 Các nguồn tài nguyên khác để học Matlab..... **23**

 2.1.2 Ma trận và xử lý trên ma trận **24**

 2.1.2.1 Các toán tử và hàm **24**

 2.1.2.2 Tạo ma trận **24**

 2.1.2.3 Kích thước ma trận **28**

 2.1.2.4 Chuyển vị ma trận/vector **29**

 2.1.2.5 Tổng và trung bình **30**

 2.1.2.6 Tìm giá trị nhỏ nhất và lớn nhất trong ma trận **32**

 2.1.2.7 Nối và ghép ma trận..... **34**

 2.1.2.8 Chỉ số ma trận **35**

 2.1.2.9 Chỉ số ma trận logic..... **37**

 2.1.2.10 Phép gán trong ma trận **39**

 2.1.2.11 Phép xóa trong ma trận **40**

 2.1.2.12 Mở rộng ma trận..... **41**

2.1.2.13	Chỉ số tuyến tính	43
2.1.2.14	Phép điều chỉnh kích thước ma trận giữ nguyên số lượng phần tử.....	45
2.1.2.15	Phép toán trên từng phần tử ma trận.....	48
2.1.2.16	Phép nhân ma trận	50
2.1.2.17	Phép chia ma trận	51
2.1.2.18	Đại số tuyến tính	53
2.1.2.19	Mảng đa chiều.....	54
2.1.3	Các lệnh điều khiển	55
2.1.3.1	Lệnh if, else, elseif.....	55
2.1.3.2	Lệnh switch.....	56
2.1.3.3	Lệnh for	57
2.1.3.4	Lệnh while	58
2.1.4	Chuỗi	59
2.1.4.1	Mảng ký tự	60
2.1.4.2	Xuất chuỗi.....	64
2.1.4.3	Mảng cell	65
2.1.4.4	So sánh chuỗi	66
2.1.4.5	Các toán tử trên tập.....	68
2.1.4.6	Câu trúc.....	71
2.1.5	Sửa lỗi (debug).....	73
2.1.6	Phản chú giải.....	75
2.2	Vẽ trong Matlab	75
2.2.1.1	Vẽ đồ thị	75
2.2.1.2	Đặt tiêu đề và đặt tên cho các trục đồ thị	79
2.2.1.3	Vẽ trên cùng một đồ thị và ghi chú đồ thị	81
2.2.1.4	Chia nhỏ đồ thị và đồ thị con	83
2.2.1.5	Câu hình đồ thị	84
2.2.1.6	Biểu đồ tần suất	84
2.2.1.7	Vẽ hàm mật độ phân bố xác suất và hàm phân bố xác suất tích lũy	87
2.2.1.8	Đọc một file ảnh và trình chiếu	91
2.2.1.9	Hàm lưu trữ và in ấn	93
2.2.1.10	LaTeX trong Matlab	93
2.3	Lập trình giao diện và ứng dụng trong Matlab	96
2.4	Cách thức lập trình trong Matlab.....	100
2.5	Giới thiệu các toolbox Matlab quan trọng khác	103
2.6	Bài tập	103
CHƯƠNG 3: GIỚI THIỆU VỀ SIMULINK		106
3.1	Giới thiệu.....	106
3.2	Simulink	106
3.2.1	Khởi động Simulink.....	106
3.2.2	Các thành phần cơ bản của Simulink.....	107
3.2.3	Các bước xây dựng mô hình Simulink.....	108
3.3	Simulink Communication Toolbox	111

3.4	Bài tập	113
-----	---------------	-----

CHƯƠNG 4: MÔ PHỎNG TÍN HIỆU VÀ QUÁ TRÌNH THU PHÁT.....115

4.1	Biến ngẫu nhiên và tạo ra biến ngẫu nhiên.....	115
4.1.1	Biến ngẫu nhiên.....	115
4.1.1.1	Tạo ra số giả ngẫu nhiên	115
4.1.1.2	Tạo số ngẫu nhiên rời rạc với hàm xác suất khối cho trước.....	117
4.1.1.2.1	Phương pháp biến đổi ngược	117
4.1.1.2.2	Phương pháp chấp nhận - loại trừ (Acceptance - Rejection technique)	118
4.1.1.2.3	Phương pháp tổng hợp (Composition approach).....	120
4.1.1.3	Tạo số ngẫu nhiên liên tục cho hàm ngẫu nhiên.....	122
4.1.1.3.1	Phương pháp biến đổi ngược	122
4.1.1.3.2	Giải thuật loại bỏ.....	123
4.1.2	Kỳ vọng	125
4.1.3	Phương sai.....	126
4.1.4	Hiệp phương sai	128
4.1.5	Bất đẳng thức Markov	130
4.1.6	Bất đẳng thức Chebyshev.....	130
4.1.7	Định luật số lớn	131
4.1.8	Định lý giới hạn trung tâm	132
4.1.9	Biến ngẫu nhiên rời rạc.....	136
4.1.9.1	Phân bố nhị thức (Binomial distribution).....	136
4.1.9.2	Phân bố hình học (Geometric Random).....	139
4.1.9.3	Phân bố Poisson	140
4.1.10	Biến ngẫu nhiên liên tục.....	142
4.1.10.1	Phân bố đều.....	142
4.1.10.2	Phân bố chuẩn	148
4.1.10.3	Phân bố mũ	151
4.1.10.4	Phân bố Rayleigh.....	153
4.1.10.5	Phân bố Nakagami-m	156
4.2	Mô phỏng nguồn tín hiệu	160
4.2.1	Tín hiệu tương tự	160
4.2.2	Tín hiệu số.....	162
4.2.3	Tín hiệu ngẫu nhiên	165
4.3	Mã hóa.....	168
4.3.1	Mã hóa nguồn.....	168
4.3.2	Mã phát hiện và sửa lỗi	175
4.3.2.1	Mã CRC	176
4.3.2.2	Mã khôi.....	177
4.3.2.3	Mã xoắn	181
4.4	Điều chế và giải điều chế	182
4.4.1	Điều chế tương tự	183
4.4.1.1	Điều chế tương tự băng gốc	183
4.4.1.2	Điều chế tương tự băng dài	184
4.4.2	Điều chế số.....	188
4.5	Quá trình lọc	197

4.6	Bài tập	199
CHƯƠNG 5: MÔ PHỎNG KÊNH THÔNG TIN		201
5.1	Giới thiệu chung	201
5.2	Mô hình kênh	201
5.2.1	Kênh truyền nhị phân.....	202
5.2.2	Kênh nhiễu trắng	203
5.2.3	Kênh truyền log-normal	208
5.2.4	Kênh fading Rayleigh	210
5.2.5	Kênh fading Nakagami-m	213
5.2.6	Kênh truyền fading Rician	216
5.2.7	Kênh truyền tương quan.....	218
5.3	Bài tập	220
CHƯƠNG 6: ƯỚC TÍNH THAM SỐ VÀ ĐÁNH GIÁ HIỆU NĂNG		222
6.1	Các tham số hiệu năng của hệ thống thông tin	222
6.1.1	Tỷ số tín hiệu trên nhiễu trung bình.....	222
6.1.2	Xác suất dừng.....	222
6.1.3	Dung lượng dừng của kênh truyền	223
6.1.4	Xác suất lỗi bit.....	223
6.2	Ước tính các tham số hiệu năng	224
6.2.1	Giới thiệu về phương pháp Monte Carlo.....	224
6.2.2	Ước tính tỷ số tín hiệu trên nhiễu trung bình.....	225
6.2.3	Ước tính dung lượng hệ thống.....	229
6.2.4	Ước tính xác suất dừng	230
6.2.5	Ước tính tỷ lệ lỗi bit/symbol	232
6.3	Thực hiện mô phỏng hệ thống viễn thông.....	232
6.3.1	Mô phỏng hệ thống phân tập thu	232
6.3.1.1	Hệ thống kết hợp lựa chọn (Selection combiner)	232
6.3.1.2	Kỹ thuật kết hợp phân tập theo tỷ lệ tối ưu	237
6.3.2	Mô phỏng hệ thống mã không gian thời gian (phân tập phát)	241
6.3.3	Mô phỏng hệ thống truyền thích ứng	245
6.3.4	Mô phỏng hệ thống truyền giải mã và chuyển tiếp hai chặng	252
6.3.4.1	Phân tích chính xác.....	253
6.3.4.2	Phân tích xấp xỉ.....	254
6.4	Bài tập	257

CHƯƠNG 1: LÝ THUYẾT MÔ PHỎNG

1.1 Tổng quan

Chúng ta bắt đầu cuốn sách này với hai câu hỏi là (i) Tại sao chúng ta phải cần mô phỏng; (ii) Kết quả mô phỏng có ý nghĩa như thế nào với hệ thống thực tế.

Mô phỏng là quá trình ngược của quá trình xây dựng mô hình toán cho hệ thống thực tế. Quá trình thuận là từ hệ thống thực, chúng ta đo đạc lấy mẫu và xây dựng mô hình toán cho hệ thống thể hiện mối quan hệ giữa đầu vào và đầu ra.

Một ví dụ tiêu biểu cho quá trình này trong ngành điện tử viễn thông là quá trình xây dựng mô hình kênh truyền vô tuyến, các nhà khoa học đã thực đo đạc thực hiện địa kênh truyền và xây dựng hàm thể hiện đặc tính kênh truyền mà chúng ta có thể thấy qua mô hình kênh truyền Hata hay Okumura [1]. Sau khi có mô hình toán của hệ thống, chúng ta thực hiện mô phỏng hệ thống để có sự hiểu biết về hệ thống trong những tình huống và môi trường mà chúng ta giả định.

Với câu hỏi thứ nhất, câu trả lời là không phải lúc nào chúng ta có thể tiếp cận hệ thống thực để thực hiện đo đạc và thu thập dữ liệu để đánh giá hệ thống trong những tình huống và môi trường mong muốn vì nhiều lý do, đơn cử như:

- Lý do tồn tại: hệ thống thực chưa tồn tại trong thực tế mà chỉ tồn tại qua mô tả toán học và các giả định.
- Lý do thời gian: có những sự kiện mà thời gian xảy ra rất lâu hoặc rất ngắn hoặc không thể lặp lại vì chỉ xảy ra một lần.

Ví dụ: nhật thực.

- Lý do nguy hiểm: không thể thử nghiệm vì quá nguy hiểm hoặc để giảm bớt nguy hiểm.

Ví dụ: thử bom nhiệt hạch, những thí nghiệm ngoài không gian, dưới biển sâu, thử phản ứng thuỷ,...

- Lý do chi phí: giảm bớt chi phí vì quá tốn kém để có thể thực hiện trên hệ thống thực, hoặc thử nghiệm là phải phá hủy.

Ví dụ: thử nghiệm phá hủy kiểm chứng mức độ an toàn của xe hơi.

Thông thường, chúng ta sử dụng mô phỏng để hiểu và/hoặc điều khiển một hệ thống phức tạp mà các kỹ thuật khác không thể làm được, ví dụ như phương pháp phân tích đại số và phương pháp số. Đặc điểm chính của các phương pháp là như sau:

Phương pháp phân tích đại số:

- **Ưu điểm:**
 - + Cho kết quả chính xác và có thể khảo sát nhiều tham số quyết định đồng thời.
 - + Thời gian tính toán gần như tức thời và không phụ thuộc vào độ chính xác của tham số muôn ước lượng.
 - + Các kết quả đánh giá là không đổi giữa các lần tính toán.
- **Nhược điểm:**
 - + Chỉ áp dụng cho hệ thống đơn giản. Không phải hệ thống nào cũng có thể dùng phương pháp phân tích đại số, đặc biệt là các hệ thống có mô hình phức tạp và đa biến.

Phương pháp số:

- **Ưu điểm:**
 - + Có thể áp dụng cho hầu hết các hệ thống, kể cả các hệ thống phức tạp.
- **Nhược điểm:**
 - + Thời gian tính toán lớn và mức độ tính toán lớn.
 - + Khi thay đổi một tham số hệ thống và môi trường là phải tính toán lại từ đầu.
 - + Độ chính xác phụ thuộc vào thời gian tính toán.
 - + Các kết quả đánh giá là không đổi giữa các lần tính toán.

Phương pháp mô phỏng:

- **Ưu điểm:** Áp dụng cho các hệ thống phức tạp gần thực tế và cả hệ thống thực tế.
- **Nhược điểm:** Kết quả thu được giữa các lần mô phỏng là không giống nhau

Một điểm quan trọng mà chúng ta cần chú ý là cho cả ba phương pháp, chúng ta cần phải có mô hình toán học của hệ thống. Mô hình toán học có thể là hàm truyền thể hiện mối quan hệ giữa đầu vào và đầu ra của hệ thống kèm theo các giả định về đặc điểm của dữ liệu đầu vào. Sử dụng chung một mô hình toán học là đảm bảo cho kết quả của ba phương pháp là hội tụ giống nhau.

Một câu hỏi thứ hai quan trọng là các kết quả mô phỏng có ý nghĩa như thế nào với hệ thống thực. Hay nói cách khác là các kết quả mô phỏng đạt được có thực sự phản ánh kết quả của hệ thống thực hay không. Câu trả lời là phụ thuộc vào mô hình toán mà chúng ta xây dựng cho hệ thống thực. Nếu mô hình toán phản ánh đúng các đầu vào và đầu ra của hệ thống, các trạng thái hệ thống và hàm truyền hệ thống, thì kết quả mô phỏng sẽ tương tự như kết quả của hệ thống thực và ngược lại. Do đó, chúng ta cần phải thực sự hiểu rõ các giả sử của các mô hình toán để diễn giải các kết quả mô phỏng và sự khác biệt của kết quả mô phỏng với kết quả thực tế nếu có.

Trong quá trình đánh giá hiệu năng và khảo sát đặc điểm của hệ thống thông tin, chúng ta thông thường sử dụng hai hoặc cả ba phương pháp nêu trên trong một số trường hợp. Phương pháp phân tích đại số gần như là bắt buộc nếu chúng ta có thể thực hiện được. Phương pháp số và phương pháp mô phỏng là dùng để kiểm chứng lại phương pháp phân tích đại số.

1.2 Một số khái niệm

Mục này sẽ cung cấp một số khái niệm của lý thuyết mô phỏng, giúp chúng ta dễ dàng hơn trong việc nghiên cứu lý thuyết mô phỏng.

1.2.1 Mô phỏng máy tính

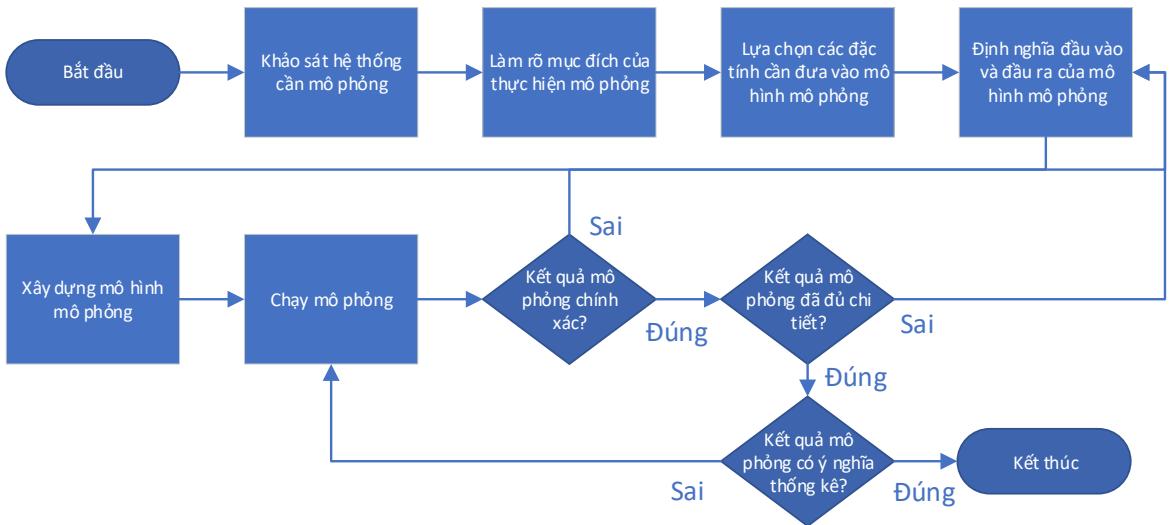
Là dùng phần mềm thiết lập một tình huống giả định (một quá trình, một trạng thái) trên máy tính thông qua một mô hình toán cho trước.

1.2.2 Mô hình

Mô hình mô phỏng là một cách mô tả toán học về một hệ thống vật lý. Một hệ thống có thể được nghiên cứu bằng nhiều mô hình toán học khác, sao cho phù hợp với bài toán đặt ra và phương pháp giải quyết bài toán.

Việc lựa chọn mô hình toán học để thực hiện mô phỏng là một bước quan trọng đảm bảo kết quả mô phỏng có ý nghĩa khoa học. Nếu lựa chọn đúng mô hình mô phỏng thì sẽ làm đơn giản hóa quá trình phân tích và quá trình mô phỏng. Ví dụ khi xem xét mô hình kênh fading giữa máy phát và máy thu có tầm nhìn thẳng, việc sử dụng kênh truyền Rician là phù hợp hơn là kênh truyền fading Rayleigh hay là kênh truyền nhiễu trắng. Một ví dụ tiêu biểu khác là nếu chúng ta muốn kiểm chứng một kiểu mã hóa mới hay một kiểu điều chế mới thì có nên chọn những kênh truyền phức tạp như kênh truyền Nakagami- m ? Thực tế là nếu chúng ta chỉ tập trung vào độ lợi mã thì kênh truyền nhiễu trắng là đủ và sẽ làm mô hình toán của hệ thống cần khảo sát đơn giản và dễ phân tích. Các kết quả mô phỏng cũng dễ dàng được đánh giá và so sánh khi không bị trộn lẫn bởi các hiệu ứng khác.

Do đó, không phải mô hình mô phỏng càng phức tạp càng gần với hệ thống thực là tốt mà phụ thuộc vào vấn đề chúng ta khảo sát và nghiên cứu. Để kết quả nghiên cứu chuẩn xác, chúng ta cần phải lựa chọn mô hình mô phỏng phù hợp nhất để tiến hành nghiên cứu. Đồng thời các kết quả mô phỏng đạt được cần được diễn giải theo mô hình mô phỏng sử dụng và các điều kiện giả sử kèm theo. Các bước tiến hành mô phỏng trình bày ở Hình 1.1.



Hình 1.1: Các bước tiến hành mô phỏng.

Ví dụ 1.1

Xem xét hệ thống thông tin vô tuyến, hãy khảo sát các bước cần thiết để tiến hành mô phỏng trên máy tính.

Giải: Khi muốn mô phỏng hệ thống thông tin vô tuyến, chúng ta cần thực hiện các bước sau:

- Bước 1: Khảo sát hệ thống cần mô phỏng. Cần trả lời các câu hỏi sau, ví dụ: Hệ thống thông tin vô tuyến phục vụ cho mục đích gì và có chất lượng dịch vụ yêu cầu là như thế nào? Đặc tính kỹ thuật của hệ thống là gì? Các tiêu chuẩn kỹ thuật của hệ thống? Hệ thống có những khói chức năng chính nào? Đặc điểm chính của hệ thống là gì?
- Bước 2: Mục đích của mô phỏng là khảo sát yếu tố/đặc điểm gì của mạng thông tin vô tuyến? Từ đó chúng ta sẽ quyết định tập trung vào mô phỏng ở lớp nào. Nếu chúng ta muốn khảo sát về kênh truyền hay kiểu điều chế, chúng ta sẽ tập trung vào lớp vật lý. Nếu chúng ta muốn khảo sát về đa truy nhập giữa các người dùng, chúng ta sẽ tập trung vào lớp MAC.
- Bước 3: Lựa chọn đặc tính đưa vào mô hình mô phỏng. Tùy thuộc vào mục đích mô phỏng mà chúng ta chọn các đặc tính tương ứng. Mục đích mô phỏng sẽ quyết định thông số hiệu năng chọn để mô phỏng. Ví dụ, nếu chúng ta muốn khảo sát đặc tính của một kiểu mã hóa mới, tỷ lệ lỗi bit và dung lượng Shannon là các tham số hiệu năng nên chọn.
- Bước 4: Định nghĩa đầu vào và đầu ra. Thông số hiệu năng là giá trị đầu ra của mô phỏng và quyết định đến đầu vào của hệ thống.
- Bước 5: Xây dựng mô hình mô phỏng. Để kết quả mô phỏng có ý nghĩa, mô hình mô phỏng phải phù hợp với mục tiêu mô phỏng. Mô hình mô phỏng đơn giản, kết quả mô phỏng không có tính thực tế và không có ý nghĩa khoa học. Mô hình mô phỏng

quá phức tạp, thời gian và mức độ phức tạp mô phỏng sẽ tăng lên và khó giải thích ý nghĩa khoa học của kết quả mô phỏng.

- Bước 6: Kiểm tra tính chính xác và ý nghĩa khoa học của kết quả mô phỏng? Làm sao để biết kết quả mô phỏng là đúng với mô hình toán học của hệ thống mô phỏng? Phương pháp nào để kiểm chứng kết quả mô phỏng? Khảo sát từng tham số đặc tính của hệ thống để từ đó làm rõ ảnh hưởng của tham số đó lên hiệu năng của hệ thống.

1.3 Ứng dụng của mô phỏng

Ngày nay, cùng với sự phát triển mạnh mẽ của máy tính cá nhân, máy chủ và máy chủ ảo, mô phỏng trở thành một phần không thể thiếu trong các quy trình thiết kế, chế tạo và tối ưu hệ thống. Lĩnh vực ứng dụng của mô phỏng trải dài tất cả các lĩnh vực như kinh tế, tài chính, sinh học, công nghệ và khoa học. Trong viễn thông, mô phỏng là một phần bắt buộc khi thiết kế hệ thống viễn thông.

1.4 Vai trò của mô phỏng

Về cơ bản, mô phỏng có vai trò rất quan trọng hiện nay trong nghiên cứu khoa học và phát triển công nghệ. Ưu điểm của mô phỏng là không thể chối cãi.

- Kiểm chứng lý thuyết và tiên đoán thực nghiệm: Kết quả mô phỏng có thể kiểm chứng lại tính chính xác so với kết quả thí nghiệm nếu là ngành khoa học thực nghiệm và kết quả phân tích đại số nếu là ngành khoa học cơ bản. Kết quả mô phỏng có thể dùng để dự đoán và định hướng kết quả thí nghiệm/phân tích trong một số trường hợp. Nhanh và rẻ hơn so với làm thí nghiệm.
- Thực hiện nghiên cứu: Mô phỏng có thể được thực hiện để dự đoán hành vi, tính chất của đối tượng nghiên cứu trước khi đối tượng đó tồn tại. Ví dụ như mô phỏng dùng để tìm hiểu cơ chế, tính chất của đối tượng cần nghiên cứu.

1.5 Các phần mềm mô phỏng

Hiện nay, do mô phỏng có vai trò rất quan trọng trong nghiên cứu và học tập, nên mỗi ngành đều có một phần mềm mô phỏng chuyên biệt. Riêng đối với hệ thống thông tin, các phần mềm sau đây là tiêu biểu và cần thiết:

1.5.1 Matlab

MATLAB® là một nền tảng lập trình được thiết kế dành riêng cho các kỹ sư và các nhà khoa học. Phần chính của nền tảng Matlab là ngôn ngữ Matlab, là ngôn ngữ bậc cao dựa trên ma trận cho phép biểu hiện tự nhiên nhất của toán học tính toán. Người dùng định hướng của ngôn ngữ Matlab là những người không chuyên về lập trình, có thể nhanh chóng sử dụng Matlab phục vụ cho công việc nghiên cứu và học tập.

Sử dụng Matlab, chúng ta có thể:

- Phân tích dữ liệu;
- Phát triển thuật toán;
- Tạo mô hình và ứng dụng.

Ngôn ngữ, ứng dụng và các hàm toán học tích hợp trong Matlab cho phép chúng ta nhanh chóng khám phá nhiều cách tiếp cận để đi đến một giải pháp. Matlab cho phép chúng ta kiểm chứng ý tưởng của mình từ nghiên cứu đến sản xuất bằng cách triển khai các ứng dụng doanh nghiệp và thiết bị nhúng, cũng như tích hợp với Simulink® và thiết kế dựa trên mô hình.

Matlab có một nhóm các giải pháp dành riêng cho từng chuyên ngành được gọi là toolbox. Các toolbox là tập hợp các hàm Matlab (có đuôi file là .m) hay các đối tượng để giúp chúng ta giải quyết một vấn đề cụ thể trong một hướng chuyên môn. Toolbox cho phép chúng ta nhanh chóng tìm hiểu, kiểm chứng hay xây dựng một giải thuật. Ví dụ trong lĩnh vực truyền thông và xử lý tín hiệu, chúng ta có toolbox communications và toolbox signal processing.

1.5.2 *Mathematica*

Mathematica là một chương trình tính toán toán học theo tham số (theo symbol), được sử dụng trong nhiều lĩnh vực khoa học, kỹ thuật và toán học. Mathematica được tạo ra bởi Stephen Wolfram và được phát triển bởi trung tâm nghiên cứu Wolfram. Ngôn ngữ Wolfram là ngôn ngữ lập trình được sử dụng trong Mathematica.

Trong nghiên cứu và phân tích hiệu năng hệ thống thông tin, Mathematica rất hữu dụng và là phần mềm thường được sử dụng để phân tích đại số, ví dụ như thực hiện các tích phân, đạo hàm, phân tích chuỗi.

Nếu muốn nghiên cứu sâu về Mathematica, chúng ta có thể truy xuất các nguồn tài liệu sau:

- Trung tâm học tập Wolfram: <http://www.wolfram.com/support/learn/>
- Trung tâm tài liệu Mathematica: <http://reference.wolfram.com>
- Thư viện lưu trữ Wolfram: <http://library.wolfram.com/>
- Wolfram Alpha Computational Knowledge Engine: <http://www.wolframalpha.com/>
- Eric Weisstein MathWorld Mathematical resources: <http://mathworld.wolfram.com/>

1.5.3 *Scilab*

Scilab là phần mềm mã nguồn mở và miễn phí cho tính toán số cho các ứng dụng khoa học và kỹ thuật. Hay nói cách khác, Scilab là một phần mềm phiên bản mở thay thế cho Matlab.

Scilab được phát hành dưới dạng mã nguồn mở theo giấy phép GPL và có sẵn để tải xuống miễn phí. Scilab bao gồm hàng trăm chức năng toán học. Scilab có ngôn ngữ lập trình cấp cao, tương tự như ngôn ngữ Matlab, cho phép truy cập vào các cấu trúc dữ liệu nâng cao, chức năng đồ họa 2-D và 3-D.

Scilab có thể thực hiện các chức năng sau đây:

- Toán và mô phỏng
- Biểu diễn 2-D và 3-D: Tập hợp các chức năng đồ họa để trực quan hóa, chú thích và xuất dữ liệu và nhiều cách để tạo và tùy chỉnh các loại ô và biểu đồ khác nhau.
- Tối ưu hóa: các thuật toán để giải quyết các vấn đề tối ưu hóa liên tục và rời rạc và không ràng buộc.
- Số liệu thống kê: công cụ để thực hiện phân tích dữ liệu và mô hình hóa.
- Hệ thống điều khiển: các thuật toán và công cụ tiêu chuẩn để nghiên cứu hệ thống điều khiển.
- Xử lý tín hiệu: trực quan hóa, phân tích và lọc tín hiệu trong các miền thời gian và tần số.
- Xây dựng và phát triển ứng dụng.
- Xcos: mô hình hệ thống động bao gồm: mô hình hệ thống cơ khí, mạch thủy lực và hệ thống điều khiển.

1.5.4 NS-3

NS viết tắt của chữ Network Simulator, là tên của ba thế hệ trình giả lập mạng ns-1, ns-2 và ns-3. Phiên bản hiện tại là ns-3 và không tương thích với ns-2 và ns-1. ns-3 là phần mềm miễn phí, được cấp phép theo giấy phép GNU GPLv2 và được cung cấp công khai để nghiên cứu, phát triển và sử dụng.

- Thông tin về ns-3: <https://www.nsnam.org/about/>
- Trang phát triển ns-3 tại gitlab: <https://gitlab.com/nsnam/ns-3-dev/>

1.5.5 Opnet

Opnet là phần mềm thương mại mô phỏng mạng được phát triển tại MIT. Opnet bao gồm:

- Công cụ để mô hình hóa mạng bao gồm LAN, WAN, mạng có dây, mạng không dây, mạng di động và mạng vệ tinh.
- Công cụ để xây dựng, thực thi mô phỏng và debug mô phỏng.
- Công cụ để thu thập, phân tích và trình bày kết quả mô phỏng.
- Tài liệu tra cứu và hướng dẫn.

Tham khảo thêm ở:

<https://support.riverbed.com/content/support/software/opnet-model/modeler.html>.

1.6 Bài tập

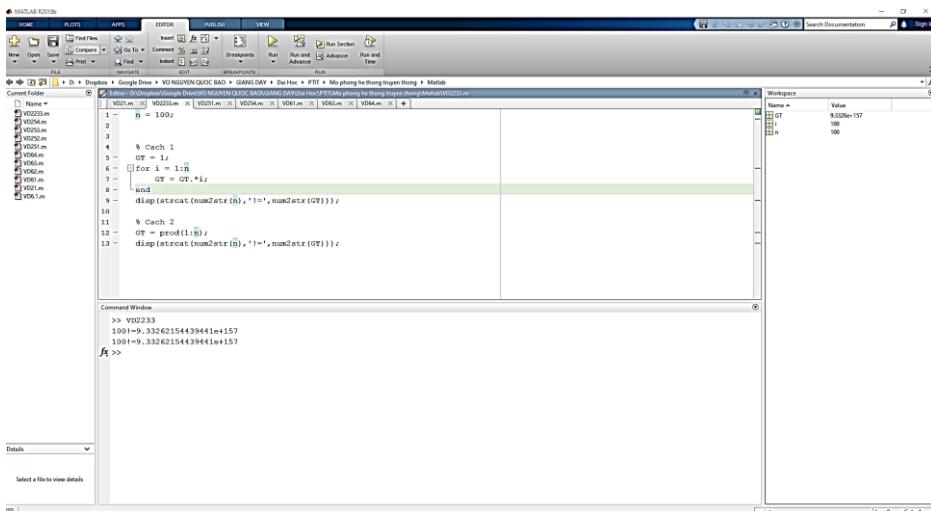
1. So sánh ưu điểm và nhược điểm của ba phương pháp phân tích đại số, phương pháp số và phương pháp mô phỏng? Nêu ví dụ minh họa.
2. Hãy nêu vai trò của mô phỏng trong thiết kế hệ thống thông tin vô tuyến?
3. Khi không có mô hình toán của hệ thống, mà chỉ có tập giá trị đầu vào và đầu ra, chúng ta có thể mô phỏng hệ thống hay không?
4. So sánh ưu và nhược điểm của phần mềm Matlab và phần mềm Mathematica trong mô phỏng hệ thống truyền thông.
5. Phần mềm Excel có thể thực hiện mô phỏng được không? Hãy sử dụng phần mềm Excel tạo biến phân phối đều và vẽ biểu đồ tần suất. So sánh với Matlab.
6. So sánh ưu điểm và nhược điểm của phần mềm Matlab và phần mềm Scilab.
7. So sánh tốc độ thực thi của phần mềm Matlab và phần mềm Scilab của phương trình $AX = B$ với A có kích thước 1000×1000 .
8. Hãy so sánh thư viện symbolic của Matlab và phần mềm Mathematica.
9. Cài đặt ns-3 và thử một ví dụ đơn giản.
10. Cài đặt Opnet và thử một ví dụ đơn giản.

CHƯƠNG 2: GIỚI THIỆU MATLAB

2.1 Giới thiệu tổng quan

2.1.1 Giao diện bắt đầu

Để bắt đầu sử dụng Matlab, chúng ta cần cài đặt Matlab trên máy tính sử dụng hệ điều hành Window hay Mac. Sau khi cài đặt xong, chúng ta khởi động Matlab bằng cách nhấp đôi vào biểu tượng Matlab và giao diện mặc định sau khi cài như Hình 2.1.



Hình 2.1: Giao diện Matlab sau khởi động.

Trong cửa sổ khởi động, chúng ta có thể thấy các cửa sổ con bao gồm:

- Folder hiện tại (Current Folder): liệt kê các file có trong folder hiện tại. Chúng ta có thể thực hiện chạy một file Matlab trong folder này bằng cách gấp và thả file này vào phần cửa sổ lệnh hay thực hiện chỉnh sửa bằng cách gấp và thả vào cửa sổ biên soạn.
- Trình biên soạn (Editor): là một giao diện cho phép chúng ta lập trình và chỉnh sửa chương trình cũng như tiến hành chạy chương trình Matlab. Bên cạnh đó trình biên soạn còn cung cấp các chức năng để kiểm lỗi.
- Workspace: là nơi lưu trữ các biến hiện tại, bao gồm hai cột tên biến và giá trị lưu trong biến. Để biết rõ thêm về từng biến, ta nhấp đôi vào biến.
- Cửa sổ lệnh (Command Window): là cửa sổ cho phép chạy các lệnh Matlab nhập trực tiếp vào hay xuất kết quả của quá trình thực thi lệnh Matlab.
- Details: sẽ xuất hiện các hướng dẫn dưới dạng comment (sau dấu %) trong script Matlab.

Trong một số trường hợp, một số cửa sổ con mất đi, chúng ta có thể phục hồi lại các cửa sổ con mặc định bằng cách vào **Tab Home - Layout - Default**.

Trước tiên ta sẽ thử một số lệnh cơ bản trong cửa sổ lệnh của Matlab trong Ví dụ 2.1 sau.

Ví dụ 2.1

Nhập vào cửa sổ lệnh $1 + 1$, 2^3 , $2/3$ và $\text{or}(\text{true},\text{false})$.

Giải:

```
>> 1+1  
  
ans =  
  
    2  
  
>> 2^3  
  
ans =  
  
    8  
  
>> 2/3  
  
ans =  
  
    0.6667  
  
>> or(true,false)  
  
ans =  
  
logical  
  
    1
```

2.1.1.1 Biến

Với Matlab chúng ta rất dễ dàng để tạo ra một biến, theo phương thức khai báo và gán đồng thời. Đồng thời Matlab cũng tự thực hiện khai báo bộ nhớ và kiểu biến cho chúng ta. Sau khi khai báo, các biến sẽ xuất hiện trong phần Workspace.

Matlab phân biệt chữ thường và chữ hoa, nên biến **A** sẽ khác với biến **a**. Bên cạnh đó, tên biến không được đặt trùng với tên của các hàm lõi (built-in) của Matlab.

Ví dụ 2.2

Khởi tạo một biến $x = 1$, y là một chuỗi với “hello world”, z được gán là kết quả của hàm $\log10(100)$ và t là một vector từ 1 đến 4.

Giải: Chúng ta bắt đầu từ cửa sổ lệnh (“Command Window”) và từ dấu $>>$.

```
x = 1  
y = 'hello world'
```

```
z = log10(100)
t = [1 2 3 4]
```

Kết quả xuất hiện trên cửa sổ lệnh như sau:

```
>>
x =
1
y =
'hello world'
z =
2
t =
1 2 3 4
```

Lưu ý rằng tên biến phải bắt đầu bằng 1 ký tự và chiều dài tối đa của tên biến là 63 ký tự. Sau đây là một số đề nghị khi đặt tên biến trong Matlab:

- Tên biến phải có tính gợi nhớ, phải có ý nghĩa trong ngữ cảnh để dễ dàng nhận biết, không đặt quá ngắn và cũng không đặt quá dài.
- Không đặt tên biến trùng với các hàm lõi và hàm có sẵn của Matlab.
- Chỉ nên viết hoa toàn bộ tên biến nếu biến là hằng số.
- Tên biến bắt đầu với chữ n thường có thể dùng để hiển thị chiều dài hay số phần tử của biến.
- Nên dùng các biến i, j, k, id, idx, iRun là các biến chạy trong các hàm **for** hay **while**.
- Hạn chế tái sử dụng tên biến trong một chương trình.

Tất cả các biến sau khi khai báo hợp lý sẽ xuất hiện trong phần Workspace. Để lưu trữ các biến này thành dạng file và sử dụng lại, ta dùng lệnh **save()** và **load()**.

Lệnh **save()** và **load()** đặc biệt hữu dụng và nên dùng khi tiến hành mô phỏng các hệ thống trên Matlab, đặc biệt các chương trình mô phỏng có thời gian thực thi lâu và có nhiều bước hay trạng thái chuyển đổi. Lệnh **save()** cho phép chúng ta lưu các biến mong muốn tại thời điểm lưu và lệnh **load()** cho phép tải lại giá trị của biến đã lưu vào Workspace khi cần thiết.

Lệnh **clear all** nhằm để xóa tất cả các biến hiện tại có trong Workspace.

Ví dụ 2.3

Hãy lưu tất cả các biến hiện có ở Workspace thành file luutam.mat. Thực hiện xóa Workspace và gọi lại các biến có trong file luutam.mat.

Giải:

```
>> save luutam  
>> clear all  
>> load luutam
```

Sau khi thực hiện lệnh **save**, ta sẽ thấy file có tên là luutam.mat xuất hiện ở folder hiện tại.

2.1.1.2 Môi trường làm việc

Như đã nói ở trên, môi trường làm việc (Workspace) là nơi lưu trữ các biến hiện có. Khi chúng ta thoát Matlab, các biến này sẽ mất đi kể cả khi khởi động Matlab lại. Liên quan tới Workspace, chúng ta có các hàm sau:

- **whos**: liệt kê tên, kích thước, số byte và các thông số khác của các biến;
- **workspace**: mở cửa sổ con Workspace;
- **clear**: xóa một hay nhiều biến, hoặc tất cả các biến.

```
>> whos  
Name      Size            Bytes  Class     Attributes  
GT        1x1              8  double  
ans       1x1              1  logical  
i         1x1              8  double  
n         1x1              8  double
```

Ví dụ 2.4

Nhập vào cửa sổ lệnh x, y, z với giá trị lần lượt là 1, 2, 3. Dùng lệnh clear để xóa x và y, sau đó dùng lệnh clear all. Quan sát Workspace.

Giải: Chúng ta nhập x, y và z cùng một lúc vào cửa sổ lệnh như sau.

```
>> x=1; y=2; z=3;  
>> clear x y  
>> clear all
```

2.1.1.3 Cửa sổ lệnh

Trong cửa sổ lệnh, chúng ta có thể gõ từng dòng lệnh và xem kết quả thực thi của Matlab. Nếu sau dòng lệnh có dấu ; thì Matlab vẫn thực thi lệnh nhưng không xuất kết quả thực thi ra màn hình.

Ví dụ 2.5

Nhập vào cửa sổ lệnh x = 1; và x = 1. Quan sát kết quả.

Giải: Kết quả ở cửa sổ lệnh như sau.

```
>> x=1;  
>> x=1  
  
x =  
  
1
```

Có một số lưu ý khi làm việc trong cửa sổ lệnh:

- Để truy xuất lại lịch sử các lệnh đã dùng, chúng ta dùng phím mũi tên lên ↑;
- Dùng dấu TAB khi gõ tên hàm để Matlab gợi ý các hàm mà Matlab có;
- Để xóa trắng cửa sổ lệnh, chúng ta dùng lệnh **clc**.

2.1.1.4 Phần giúp đỡ

Một trong những ưu điểm hay lợi thế của ngôn ngữ Matlab đối với kỹ sư không chuyên lập trình và người mới bắt đầu là cách Matlab cung cấp hướng dẫn sử dụng. Có bốn cách cơ bản để tìm hướng dẫn là:

- Sử dụng hàm **help()** với cú pháp “help tên hàm/toán tử” ở ngay tại dấu nhắc cửa sổ lệnh.
- Sử dụng dấu? tại thanh Quick Access Toolbar ở góc phải trên của màn hình.
- Sử dụng phím giúp đỡ F1.
- Sử dụng hàm **lookfor** trong trường hợp chúng ta không biết tên hàm chính xác.

Ví dụ 2.6

Sử dụng hàm help cho toán tử * và lệnh size.

Giải:

```
>> help *  
* Matrix multiply.  
X*Y is the matrix product of X and Y. Any scalar (a 1-by-1 matrix)  
may multiply anything. Otherwise, the number of columns of X must  
equal the number of rows of Y.
```

C = mtimes(A,B) is called for the syntax 'A * B' when A or B is an object.

See also times.

Reference page for mtimes

Other functions named mtimes

```
>> help size  
size Size of array.
```

$D = \text{size}(X)$, for M-by-N matrix X, returns the two-element row vector D = [M,N] containing the number of rows and columns in the matrix.
For N-D arrays, size(X) returns a 1-by-N vector of dimension lengths.
Trailing singleton dimensions are ignored.

[M,N] = size(X) for matrix X, returns the number of rows and columns in X as separate output variables.

[M1,M2,M3,...,MN] = size(X) for N>1 returns the sizes of the first N dimensions of the array X. If the number of output arguments N does not equal NDIMS(X), then for:

N > NDIMS(X), size returns ones in the "extra" variables, i.e., outputs NDIMS(X)+1 through N.

N < NDIMS(X), MN contains the product of the sizes of dimensions N through NDIMS(X).

M = size(X,DIM) returns the length of the dimension specified by the scalar DIM. For example, size(X,1) returns the number of rows. If DIM > NDIMS(X), M will be 1.

When size is applied to a Java array, the number of rows returned is the length of the Java array and the number of columns is always 1. When size is applied to a Java array of arrays, the result describes only the top level array in the array of arrays.

Example:

If

X = rand(2,3,4);

then

d = size(X) returns d = [2 3 4]

[m1,m2,m3,m4] = size(X) returns m1 = 2, m2 = 3, m3 = 4, m4 = 1

[m,n] = size(X) returns m = 2, n = 12

m2 = size(X,2) returns m2 = 3

See also length, ndims, numel.

Reference page for size

Other functions named size

2.1.1.5 Các nguồn tài nguyên khác để học Matlab

Matlab là một ngôn ngữ dễ dàng tự học, có nhiều nguồn hỗ trợ để tự học và tham khảo, ví dụ:

- Trang bắt đầu với Matlab: <https://www.mathworks.com/help/matlab/getting-started-with-matlab.html>
- Trung tâm Matlab: <https://www.mathworks.com/matlabcentral/?refresh=true>
- Matlab blog: [https://blogs.mathworks.com/?](https://blogs.mathworks.com/)

- Matlab trên facebook: <https://www.facebook.com/MATLAB>

2.1.2 Ma trận và xử lý trên ma trận

2.1.2.1 Các toán tử và hàm

- : [] ‘ + – * .* / \ ./ ^ .^ end && || & | ~ < > =
- zeros, ones, rand, randn, meshgrid,
- true, false, logspace, perms, randperm
- eye, diag, blkdiag, tril, triu, rot90
- size, length, numel
- ind2sub, sub2ind, find
- sort, sortrows
- sum, mean, max, min, abs
- isequal, isprime
- reshape, repmat
- inv, pinv, det, svd, eig, trace, rref, rank, orth, cov, chol
- squeeze, ndgrid, cat, ndims
- sparse, full, nnz, nzmax, nonzeros, spy
- cast, class, int32, uint8, ...

2.1.2.2 Tạo ma trận

Matlab hỗ trợ nhiều cách để khai báo và tạo ma trận trong Matlab. Cách đầu tiên là khai báo trực tiếp ở cửa sổ lệnh (“Command”) với quy ước sau:

- Dấu [] đại diện cho ma trận. Lưu ý vector là một trường hợp đặc biệt của ma trận.
- Các phần tử trên hàng cách nhau bằng khoảng trắng hoặc dấu , .
- Các cột cách nhau bằng dấu ; .

Ví dụ 2.7

Hãy khai báo một ma trận A như sau trong Matlab

$$A = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 1 \\ 3 & 3 & 3 \\ 2 & 1 & 9 \end{bmatrix}.$$

Giải:

```
>> A = [1 3 5 ; 2 4 1 ; 3 3 3 ; 2 1 9]
```

A =

1	3	5
2	4	1
3	3	3
2	1	9

Vector là một loại ma trận hàng, hay cột, ngoài phép gán có thể tạo theo cú pháp sau.

Cú pháp:

A = giá trị bắt đầu: khoảng cách tăng: giá trị cuối

Lưu ý: nếu khoảng cách tăng là dương thì giá trị cuối phải lớn hơn giá trị đầu và ngược lại.

Ví dụ 2.8

Tạo ra vector hàng chứa các số lẻ nhỏ hơn 10.

Giải: Số lẻ bắt đầu từ 1, bước nhảy là 2. Do đó, giá trị bắt đầu là 1, khoảng cách tăng là 2 và giá trị cuối là 10.

```
>> A = 1:2:10
```

A =

1	3	5	7	9
---	---	---	---	---

Ví dụ 2.9

Tạo ma trận các số lẻ dương giảm dần từ 9.

Giải: Ta có giá trị đầu là 9, khoảng cách tăng là -2 và giá trị cuối là 0.

```
>> A = 9:-2:0
```

A =

9	7	5	3	1
---	---	---	---	---

Trong một số trường hợp, chúng ta cần tạo ra ma trận rỗng, cú pháp như sau:

Cú pháp:

Tên ma trận = []

Ví dụ 2.10

Tạo ma trận rỗng dùng cú pháp [] và kiểm tra kích thước ma trận sau khi tạo ra.

Giải: Chúng ta dùng hàm size() để kiểm tra kích thước của ma trận.

```
>> A = []
```

A =

```
[]  
=> size(A)  
ans =  
    0 0
```

Số dòng và số cột của ma trận rỗng đều bằng 0.

Matlab hỗ trợ một số hàm để tạo ra các ma trận đặc biệt, cụ thể là

- **zeros**: ma trận toàn số không.
- **ones**: ma trận toàn số 1.
- **true**: ma trận logic 1.
- **false**: ma trận logic 0.

Ví dụ 2.11

Hãy tạo ra các ma trận sau:

- a. Ma trận 2×3 toàn 1.
- b. Ma trận 3×3 có các phần tử theo phân bố đều trong khoảng $[0, 1]$.
- c. Ma trận 2×5 có các phần tử phân bố chuẩn trong khoảng $[2, 5]$.
- d. Ma trận E và F theo dạng lưới từ 1 đến 5.
- e. Ma trận đơn vị 4×4 .
- f. Ma trận đường chéo 4×4 .
- g. Ma trận 1×6 với các giá trị tăng theo bậc mũ từ 10 đến 10^2 .
- h. Ghép hai ma trận theo đường chéo từ hai hàm $\text{rand}(2,2)$ và $\text{ones}(3,2)$.
- i. Lấy phần dưới đường chéo chính của ma trận $\text{ones}(3,4)$.

Giải: Chúng ta sử dụng lần lượt các hàm $\text{zeros}()$, $\text{ones}()$, $\text{rand}()$, $\text{randn}()$, $\text{meshgrid}()$, $\text{eye}()$, $\text{diag}()$, $\text{logspace}()$, $\text{blkdiag}()$ và $\text{tril}()$. Kết quả trên cửa sổ lệnh là như sau.

```
A = zeros(4,5)  
B = ones (2,3)  
C = rand(3,3)  
D = randn(2,5)  
[E,F] = meshgrid(1:5)  
G = eye(4)  
H = diag(1:4)  
I = logspace(0,2,6)  
J = blkdiag(rand(2,2),ones(3,2))  
K = tril(ones(3,4))
```

A =

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

B =

1	1	1
1	1	1

C =

0.8147	0.9134	0.2785
0.9058	0.6324	0.5469
0.1270	0.0975	0.9575

D =

2.7694	3.0349	-0.0631	-0.2050	1.4897
-1.3499	0.7254	0.7147	-0.1241	1.4090

E =

1	2	3	4	5
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5

F =

1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4
5	5	5	5	5

G =

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

H =

```

1 0 0 0
0 2 0 0
0 0 3 0
0 0 0 4

```

I =

```
1.0000 2.5119 6.3096 15.8489 39.8107 100.0000
```

J =

```

0.9595 0.0357 0 0
0.6557 0.8491 0 0
0 0 1.0000 1.0000
0 0 1.0000 1.0000
0 0 1.0000 1.0000

```

K =

```

1 0 0 0
1 1 0 0
1 1 1 0

```

2.1.2.3 Kích thước ma trận

Liên quan đến kích thước ma trận, Matlab hỗ trợ ba hàm như sau:

- **size()**: trả về kích thước các chiều của ma trận.
- **length()**: trả về chiều dài của vector hàng/cột, nếu đầu vào của hàm là ma trận thì kết quả trả về là chiều có kích thước lớn nhất.
- **numel()**: trả về tổng số phần tử trong ma trận.

Cú pháp:

```
[nrows,ncols] = size(A)
```

```
L = length(A)
```

```
N = numel(A)
```

Ví dụ 2.12

Tạo ra ma trận 3 chiều có kích thước [2 3 4] với các phần tử đều là 1. Dùng hàm size, length, numel để kiểm tra kích thước, chiều dài và tổng số phần tử.

Giải: Chúng ta dùng hàm ones() để tạo ra ma trận và các hàm size(), length() và numel() để kiểm tra.

```
>> A=ones(2,3,4);
```

```
>> [nr,nc]= size(A)
```

```
nr =
```

```
2
```

```
nc =
```

```
12
```

```
>> l=length(A)
```

```
l =
```

```
4
```

```
>> totel=numel(A)
```

```
totel =
```

```
24
```

2.1.2.4 Chuyển vị ma trận/vector

Để thực hiện chuyển vị ma trận, Matlab hỗ trợ hai cách:

- Cách 1: dùng toán tử chuyển vị ‘.
- Cách 2: dùng hàm chuyển vị **tranpose()**.

Để chuyển một ma trận ở dạng bất kỳ về vector cột, ta dùng toán tử :

Ví dụ 2.13

Cho ma trận $A = [1 \ 2 \ 3 \ 4 ; 5 \ 6 \ 7 \ 8]$, chuyển thành ma trận hàng.

Giải: Chúng ta lần lượt thực hiện các lệnh sau.

```
>> A = [1 2 3 4 ; 5 6 7 8]
```

```
A =
```

```
1 2 3 4  
5 6 7 8
```

```
>> B=A(:)
```

```
B =
```

```
1  
5  
2  
6  
3  
7
```

```

4
8

>> C=B'

C =

1   5   2   6   3   7   4   8

>> D=transpose(B)

D =

1   5   2   6   3   7   4   8

```

2.1.2.5 Tổng và trung bình

Matlab cung cấp hàm **sum()** và hàm **mean()** để tính tổng và tính trung bình của ma trận, lưu ý là có thể tính theo hàng, theo cột, hay cụ thể hơn là tính theo chiều của ma trận.

Cú pháp:

$S = \text{sum}(X, \text{'all'})$: tính tổng tất cả các phần tử của ma trận X.

$S = \text{sum}(X, \text{DIM})$: tính tổng các phần tử X theo chiều DIM.

$S = \text{sum}(X, \text{VECDIM})$: tính tổng các phần tử theo chiều chỉ ra trên vector chiều VECDIM.

Ví dụ 2.14

Cho ma trận A với 3 hàng và 3 cột, có giá trị tăng từ 1 đến 9.

- Tính tổng theo hàng, theo cột và toàn ma trận theo cách truyền thống.
- Tính tổng theo hàng, theo cột và toàn ma trận dùng hàm sum().

Giải:

```

X = [ 1 2 3; 4 5 6; 7 8 9];

%% Câu a
[nr,nc] = size(X);
tongH = zeros(1,nr);
tongC = zeros(1,nc);
tong = 0;
% Duyệt theo hàng và cột
for i=1:nr
    for j=1:nc
        tongH(i) = tongH(i) + X(i,j);
        tongC(j) = tongC(j) + X(i,j);
        tong = tong + X(i,j);
    end
end
tongH

```

```
tongC
```

```
tong
```

```
%% Câu b
```

```
sum(X,2)
```

```
sum(X,1)
```

```
sum(X,'all')
```

```
>> VD211
```

```
tongH =
```

```
6 15 24
```

```
tongC =
```

```
12 15 18
```

```
tong =
```

```
45
```

```
ans =
```

```
6
```

```
15
```

```
24
```

```
ans =
```

```
12 15 18
```

```
ans =
```

```
45
```

Cú pháp:

`mean(X,'all')`: lấy trung bình tất cả các phần tử của X.

`mean(X,DIM)`: lấy trung bình tất cả các phần tử theo chiều DIM.

`mean(X,VECDIM)`: lấy trung bình tất cả các phần tử lần lượt theo chiều quy định ở vector chiều VECDIM.

Ví dụ 2.15

Cho ma trận A với 3 hàng và 3 cột, có giá trị tăng từ 1 đến 9. Tính tổng theo hàng, theo cột và toàn ma trận.

Giải: Chúng ta dùng hàm mean() cho ma trận A như sau.

```
>> X = [ 1 2 3; 4 5 6; 7 8 9]
```

```
X =
```

```
1 2 3  
4 5 6  
7 8 9
```

```
>> mean(X,1)
```

```
ans =
```

```
4 5 6
```

```
>> mean(X,2)
```

```
ans =
```

```
2  
5  
8
```

```
>> mean(X,'all')
```

```
ans =
```

```
5
```

```
>> mean(X,[2 1])
```

```
ans =
```

```
5
```

2.1.2.6 Tìm giá trị nhỏ nhất và lớn nhất trong ma trận

Matlab cung cấp hàm **min()** và **max()** để tìm giá trị nhỏ nhất và giá trị lớn nhất của ma trận theo chiều bất kỳ hay của toàn ma trận.

Lưu ý đây là các hàm lõi của Matlab, có thể xử lý theo ma trận với tốc độ xử lý tối ưu. Do đó, chúng ta hạn chế việc sử dụng phương pháp duyệt ma trận để tìm giá trị nhỏ nhất và giá trị lớn nhất của một ma trận.

Cú pháp:

$M = \text{min}(X, [], \text{'all'})$: trả về giá trị nhỏ nhất của ma trận X.

$[M, I] = \text{min}(X, [], \text{DIM})$: tìm giá trị nhỏ nhất của ma trận X theo chiều DIM. M là kết quả trả về, I là kết quả trả về ở dạng chỉ số của giá trị nhỏ nhất.

$M = \min(X, [], VECDIM)$: tìm giá trị nhỏ nhất của ma trận X theo các chiều quy định tại vector VECDIM.

Cú pháp:

$[M, I] = \max(X)$: trả về giá trị lớn nhất của vector X. Nếu X là một ma trận, M sẽ là vector hàng chứa các phần tử lớn nhất của mỗi cột. I là vector chỉ số của các phần tử lớn nhất.

$M = \max(X, [], 'all')$: trả về giá trị lớn nhất của ma trận X.

$M = \max(X, [], DIM)$: trả về giá trị lớn nhất theo chiều DIM.

Ví dụ 2.16

Cho ma trận $X = [1 \ 2 \ 3; 4 \ 5 \ 6; 7 \ 8 \ 9]$. Hãy

- Tìm giá trị lớn nhất và giá trị nhỏ nhất theo hàng.
- Tìm giá trị lớn nhất và nhỏ nhất theo cột.
- Tìm giá trị lớn nhất và nhỏ nhất của X.

Giải: Chúng ta thực hiện các dòng lệnh sau: $\max(X, 1)$, $\min(X, 1)$, $\max(X, 2)$, $\min(X, 2)$, $\max(X, [], 'all')$, $\min(X, [], 'all')$.

```
X = [1 2 3; 4 5 6; 7 8 9]
```

```
max(X,[],1)
```

```
min(X,[],1)
```

```
max(X,[],2)
```

```
min(X,[],2)
```

```
max(X,[],'all')
```

```
min(X,[],'all')
```

ans =

7 8 9

ans =

1 2 3

ans =

3
6
9

ans =

3
6

9

ans =

9

ans =

1

2.1.2.7 Nối và ghép ma trận

Chúng ta có thể ghép nối các ma trận sử dụng phương pháp tương tự như phương pháp khởi tạo ma trận bằng cách dùng dấu [] cùng với khoảng trắng hay dấu ; để phân cách chiều.

Khi ghép nối ma trận, chúng ta cần lưu ý về số chiều của các ma trận được nối hay ghép, nếu số chiều không đồng nhất, Matlab sẽ báo lỗi.

Ví dụ 2.17

Thực thi ghép nối các ma trận sau trên Matlab và nhận xét kết quả thu được:

$$A = [[1 \ 2 \ 3], zeros(1,3)]$$

$$B = [[1 \ 2 \ 3]; ones(1,3)]$$

$$C = [99 \ A \ 42]$$

$$D = [A ; A]$$

Giải: Kết quả chúng ta có được ở cửa sổ lệnh như sau.

A =

1 2 3 0 0 0

B =

1 2 3
1 1 1

C =

99 1 2 3 0 0 0 42

D =

1 2 3 0 0 0
1 2 3 0 0 0

2.1.2.8 Chỉ số ma trận

Chỉ số ma trận là một trong những kỹ thuật rất mạnh của Matlab cho phép chúng ta trích xuất một hay nhiều phần tử ma trận theo chỉ số. Cú pháp thực hiện là như sau:

Cú pháp:

$A=X(I)$ với X là ma trận nguồn, I là chỉ số, vector, hay ma trận chỉ số và A là ma trận kết quả trả về.

Ví dụ 2.18

Thực hiện các phép toán cho ma trận A chỉ bằng một dòng lệnh như sau:

- a. Cho ma trận $A = \text{reshape}(1:36, 6, 6)$.
- b. Trích xuất phần tử hàng 3, cột 5.
- c. Trích xuất phần tử $(1,4) ; (2,4) ; (3,4)$.
- d. Trích xuất phần tử ở hàng 4 cột 1 3 lần.
- e. Trích xuất 4 phần tử có chỉ số như sau: $(2,3) ; (2,1) ; (5,3) ; (5,1)$.
- f. Trích xuất cột thứ 4.
- g. Trích xuất hàng thứ 4.
- h. Chuyển ma trận về thành vector cột.
- i. Trích xuất các phần tử ở hàng cuối cùng, cột 3.
- j. Trích xuất ma trận vuông 2×2 trên đường chéo chính có phần tử ở hàng và cột cuối cùng.
- k. Trích xuất 4 phần tử cuối cùng ở cột đầu tiên.
- l. Trích xuất tất cả phần tử theo thứ tự hàng ngược.
- m. Trích xuất ma trận đường chéo chính.

Giải:

```
A= reshape(1:36,6,6);
A=A'
A(3,5)
C = A([ 1 2 3],4)
D = A(4,[1 1 1])
E = A([2,5],[3,1])
F = A(:,4)
G = A(4,:)
H = A(:)'
I = A(end,3)
J = A(2:end,2:end)
K = A(end-4:end,1)
L = A(end:-1:1,:)
M = diag(A)

A =
```

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

ans =

17

C =

4
10
16

D =

19 19 19

E =

9 7
27 25

F =

4
10
16
22
28
34

G =

19 20 21 22 23 24

H =

Columns 1 through 18

1 7 13 19 25 31 2 8 14 20 26 32 3 9 15 21 27 33

Columns 19 through 36

```
4 10 16 22 28 34 5 11 17 23 29 35 6 12 18 24 30 36
```

I =

```
33
```

J =

```
8 9 10 11 12  
14 15 16 17 18  
20 21 22 23 24  
26 27 28 29 30  
32 33 34 35 36
```

K =

```
7  
13  
19  
25  
31
```

L =

```
31 32 33 34 35 36  
25 26 27 28 29 30  
19 20 21 22 23 24  
13 14 15 16 17 18  
7 8 9 10 11 12  
1 2 3 4 5 6
```

M =

```
1  
8  
15  
22  
29  
36
```

2.1.2.9 Chỉ số ma trận logic

Chỉ số logic trong Matlab rất hiệu quả khi chúng ta muốn trích xuất các phần tử có cùng định dạng mong muốn cho trước. Có hai bước để sử dụng chỉ số ma trận logic như sau:

- Bước 1: Tạo ra ma trận chỉ số logic ở dạng 0 và 1. Ở Bước 1 này, chúng ta có thể sử dụng các toán tử như `<>`, `<=`, `>=`, `==`, `~=`, hoặc các hàm có giá trị trả về dạng logic như `isprime()` or `isfinite()`. Trong một số trường hợp, chúng ta cần dùng lệnh `find()` để tạo ra ma trận chỉ số logic;

- Bước 2: Áp dụng vào trong ma trận muốn trích xuất dữ liệu.

Ví dụ 2.19

Cho ma trận $A = [1 \ 2 \ 3; 4 \ 5 \ 6; 7 \ 8 \ 9]$, thực hiện các phép toán sau dùng Matlab:

- Trích xuất các phần tử là số nguyên tố.
- Trích xuất các phần tử không phải là số nguyên tố.
- Trích xuất các phần tử có giá trị lớn hơn trung bình cộng của ma trận A.
- Trích xuất các phần tử lớn hơn 3 và nhỏ hơn 6.

Giải: Thực hiện trên cửa sổ lệnh của Matlab cho kết quả như sau.

```
>> A = [ 1 2 3; 4 5 6; 7 8 9]
```

A =

```
1 2 3  
4 5 6  
7 8 9
```

```
>> B= A(isprime(A))
```

B =

```
7  
2  
5  
3
```

```
>> B= A(~isprime(A))
```

B =

```
1  
4  
8  
6  
9
```

```
>> B= A(A > mean(A))
```

B =

```
7  
8  
9
```

```
>> B= A(A > 3 & A < 6)
```

B =

2.1.2.10 Phép gán trong ma trận

Phép gán là một trong những phép cơ bản trong Matlab, tuy nhiên phép gán của Matlab lại phức tạp hơn rất nhiều so với các ngôn ngữ khác khi có nhiều cách sử dụng. Hiểu biết hết các khả năng của phép gán trong Matlab, sẽ giúp chúng ta loại bỏ được rất nhiều vòng lặp duyệt và tăng tốc độ thực thi của chương trình một cách đáng kể.

Điểm khác biệt của phép gán trong Matlab là kết hợp với chỉ số logic, chúng ta có thể gán cho toàn bộ các phần tử trong ma trận, một hàng, một cột, nhiều hàng, nhiều cột, một ma trận con, hay các phần tử thỏa mãn một tiêu chí cho trước.

Ví dụ 2.20

Cho ma trận A tương tự như ví dụ trên, $A = [1 \ 2 \ 3; 4 \ 5 \ 6; 7 \ 8 \ 9]$, thực hiện các phép toán sau dùng Matlab.

- Gán phần tử ở hàng 1, cột 3 bằng 5.
- Gán các phần tử ở hàng chẵn bằng 10.
- Gán các phần tử bằng 5 bằng -5.
- Tìm các giá trị nhỏ hơn hoặc bằng không và bình phương lên.

Giải: Thực hiện trực tiếp trên cửa sổ lệnh như sau:

```
>> A = [ 1 2 3; 4 5 6; 7 8 9]
```

```
A =
```

```
1 2 3
4 5 6
7 8 9
```

```
>> A(3,5) = 1
```

```
A =
```

```
1 2 3 0 0
4 5 6 0 0
7 8 9 0 1
```

```
>> A([3,5]) = 1
```

```
A =
```

```
1 2 3 0 0
4 1 6 0 0
1 8 9 0 1
```

```
>> A(2:2:end,:) = 10
```

```
A =  
1 2 3 0 0  
10 10 10 10 10  
1 8 9 0 1
```

```
>> A(A==5)=-5
```

```
A =  
1 2 3 0 0  
10 10 10 10 10  
1 8 9 0 1
```

```
>> A=A(A<=0).^2
```

```
A =  
0  
0  
0
```

2.1.2.11 Phép xóa trong ma trận

Một điểm thú vị trong Matlab là cho phép chúng ta xóa trên ma trận bằng phép gán ma trận rỗng, ký hiệu là [].

Phép gán ma trận rỗng cho phép chúng ta xóa 1 hàng, 1 cột, nhiều hàng và cột cùng lúc và xóa cả ma trận. Phép xóa ma trận không cho phép chúng ta xóa đơn lẻ 1 phần tử trong ma trận.

Ví dụ 2.21

Cho ma trận A như ví dụ trên $A = [1\ 2\ 3; 4\ 5\ 6; 7\ 8\ 9]$, thực hiện các phép tính sau bằng hai phương pháp: lập trình truyền thống và dùng Matlab.

- Xóa dòng 2 và dòng 3.
- Xóa cột cuối cùng.

Giải: Thực hiện trực tiếp trên cửa sổ lệnh như sau:

```
>> A = [ 1 2 3; 4 5 6; 7 8 9]
```

```
A =
```

```
1 2 3  
4 5 6  
7 8 9
```

```
>> A(2:3,:)=[]
```

```
A =
```

```

1 2 3

>> A(:,end)=[]

A =

1 2

```

Ví dụ 2.22

Xóa phần tử (1,1) ở ma trận A ở Ví dụ 2.21.

Giải: Chúng ta sẽ không thực hiện được việc này. Khi thực hiện phép xóa, chúng ta sẽ thấy một dòng thông báo lỗi của Matlab như bên dưới:

```

>> A = [ 1 2 3; 4 5 6; 7 8 9]

A =

1 2 3
4 5 6
7 8 9

>> A(1,1)=[]
A null assignment can have only one non-colon index.

```

2.1.2.12 Mở rộng ma trận

Chúng ta có thể mở rộng ma trận trong Matlab theo chiều bất kỳ, mở rộng số chiều đã có, hay mở thêm chiều mới. Để mở rộng ma trận trong Matlab, chúng ta có thể dùng một số cách sau:

- Gán giá trị cho một phần tử ma trận lớn hơn kích thước của ma trận. Matlab sẽ không báo lỗi mà sẽ tự động mở rộng ma trận, tuy nhiên không khuyến khích sử dụng phương pháp này vì hiệu quả thực thi của Matlab sẽ thấp. Chúng ta nên tính toán kích thước ma trận và khai báo trước bằng cách sử dụng các hàm như ones() hay zeros().
- Ghép ma trận với nhau: điều kiện là số chiều ghép của các ma trận ghép phải bằng nhau.
- Nhân vector hàng với vector cột bằng phép nhân phần tử .*: tạo ra ma trận có số hàng bằng chiều dài vector hàng và số cột bằng chiều dài vector cột.

Ví dụ 2.23

Cho ma trận $A = [1 \ 2 \ ; \ 3 \ 4]$, hãy thực hiện:

- a. Mở rộng ma trận A thành 10 hàng 10 cột, với phần tử ở hàng 10 cột 10 có giá trị là -10 và các phần tử còn lại là bằng -3.
- b. Từ ma trận A ở câu a, tạo ma trận vuông B có kích thước gấp 4 lần ma trận A.

Giải: Thực hiện trực tiếp trên cửa sổ lệnh của Matlab như sau:

```
>> A = [1 2; 3 4]
```

A =

1	2
3	4

```
>> A(10,10)=-10
```

A =

1	2	0	0	0	0	0	0	0	0
3	4	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	-10

```
>> A(A==0)=-3
```

A =

1	2	-3	-3	-3	-3	-3	-3	-3	-3
3	4	-3	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	-3	-10

```
>> B=[A A;A A];
```

```
>> size(B)
```

ans =

20	20
----	----

Ví dụ 2.24

Cho $A = (1:5)'$ và $B = 1:5$. Hãy tìm kết quả $C = A.*B$.

Giải: Thực hiện trực tiếp trên cửa sổ lệnh như sau:

```
>> A=(1:5)'
```

A =

1	
2	

```

3
4
5

>> B=1:5

B =

1 2 3 4 5

>> A.*B

ans =

1 2 3 4 5
2 4 6 8 10
3 6 9 12 15
4 8 12 16 20
5 10 15 20 25

```

2.1.2.13 Chỉ số tuyến tính

Để định vị một phần tử trong ma trận, Matlab hỗ trợ hai loại chỉ số:

- Chỉ số theo số chiều: ví dụ như số hàng và số cột của ma trận hai chiều.
- Chỉ số tuyến tính: là một số chỉ ra thứ tự của phần tử tuần tự từ trên xuống dưới, từ trái qua phải và theo chiều tăng dần của ma trận.

Ví dụ 2.25

Cho ma trận 2×2 $A = [1\ 2; 3\ 4]$:

- a. Gán phần tử ở hàng 2 cột 2 bằng 5.
- b. Gán phần tử ở hàng 2 cột 2 bằng -5 dùng chỉ số tuyến tính.

Giải: Thực hiện trực tiếp trên cửa sổ lệnh như sau:

```
>> A=[1 2; 3 4]
```

```
A =
1 2
3 4
```

```
>> A(2,2)=5
```

```
A =
```

```
1 2
3 5
```

```
>> A(4)=-5
```

```
A =
```

1	2
3	-5

Ví dụ 2.26

Tạo ra ma trận 3×5 có phần tử tăng dần theo chỉ số tuyến tính tăng dần từ 1 đến 15.

Giải: Thực hiện trực tiếp trên cửa sổ lệnh của Matlab như sau:

```
>> A=zeros(3,5)
```

A =

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

```
>> A(1:15)=1:15
```

A =

1	4	7	10	13
2	5	8	11	14
3	6	9	12	15

Trong một số trường hợp, chúng ta cần chuyển chỉ số thường sang chỉ số tuyến tính và ngược lại. Để thực hiện điều đó, Matlab cung cấp hai hàm như sau:

Cú pháp:

[I,J] = **ind2sub**(SIZ,IND): chuyển từ chỉ số tuyến tính IND sang chỉ số thường với dòng I và cột J của ma trận có kích thước SIZ.

IND = **sub2ind**(SIZ,I,J): chuyển đổi chỉ số thường với tham số cột I và tham số dòng J sang chỉ số tuyến tính IND của ma trận có kích thước SIZ.

Chúng ta lưu ý là hai hàm trên có thể hỗ trợ cho ma trận kích thước bất kỳ.

Ví dụ 2.27

Cho ma trận ngẫu nhiên 100×100 , tìm chỉ số thường của phần tử có chỉ số tuyến tính 872.

Giải: Chúng ta sử dụng hàm **ind2sub()** và nhập các tham số là số hàng và số cột của ma trận kèm theo chỉ số tuyến tính.

```
>> [rowNDX, colNDX] = ind2sub([100 100],872)
```

rowNDX =
72

colNDX =
9

Ví dụ 2.28

Cho ma trận ngẫu nhiên 100×100 , tìm chỉ số tuyến tính của phần tử ở hàng 72 và cột 9.

Giải: Chúng ta sử dụng hàm sub2ind() như sau.

```
>> linearNDX=sub2ind([100 100], 72, 9)  
linearNDX =  
872
```

2.1.2.14 Phép điều chỉnh kích thước ma trận giữ nguyên số lượng phần tử

Trong một số trường hợp, chúng ta cần chuyển đổi kích thước của ma trận mà vẫn giữ nguyên tổng số phần tử của ma trận. Ví dụ thường gặp là khi chúng ta làm việc với các hàm mã hóa và giải mã.

Thay vì thực hiện hai bước truyền thông là chuyển đổi về thành vector và gán lại ma trận mong muốn, chúng ta có thể thực hiện yêu cầu trên với một lệnh **reshape()**.

Cú pháp:

$Y = \text{reshape}(X, M, N)$

$Y = \text{reshape}(X, [M, N])$

% với Y là ma trận trả về có kích thước $M \times N$ và Y là ma trận đầu vào. Lỗi sẽ xảy ra nếu tổng số phần tử của X không bằng $M \times N$.

Ví dụ 2.29

Cho ma trận tuyến tính 3×10 , hãy thực hiện:

- Chuyển thành ma trận 1×30 .
- Chuyển thành ma trận 30×1 .
- Chuyển thành ma trận 5×6 .
- Chuyển thành ma trận 10×3 .

Giải: Ma trận tuyến tính là ma trận mà giá trị các phần tử tăng tuyến tính theo chỉ số tuyến tính.

```
>> A=zeros(3,10);  
>> A(1:30)=1:30  
  
A =  
 1   4   7   10  13  16  19  22  25  28  
 2   5   8   11  14  17  20  23  26  29  
 3   6   9   12  15  18  21  24  27  30  
  
>> B=A(:)  
  
B =  
 1  
 2  
 3
```

```

4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

>> C=A(:)'

C =
Columns 1 through 23

 1   2   3   4   5   6   7   8   9   10   11   12   13   14   15   16   17   18   19   20   21   22   23
Columns 24 through 30

 24   25   26   27   28   29   30

>> D=reshape(A,1,30)

D =
Columns 1 through 23

 1   2   3   4   5   6   7   8   9   10   11   12   13   14   15   16   17   18   19   20   21   22   23
Columns 24 through 30

 24   25   26   27   28   29   30

>> E=reshape(A,30,1)

E =
 1
 2
 3

```

```
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
```

```
>> F=reshape(A,5,6)
```

```
F =
 1   6   11   16   21   26
 2   7   12   17   22   27
 3   8   13   18   23   28
 4   9   14   19   24   29
 5  10   15   20   25   30
```

```
>> F=reshape(A,10,3)
```

```
F =
 1   11   21
 2   12   22
 3   13   23
 4   14   24
 5   15   25
 6   16   26
 7   17   27
 8   18   28
 9   19   29
10   20   30
```

Thông thường hàm size() hoặc hàm numel() sẽ được dùng chung với hàm reshape() để giảm các lỗi sai sót về số chiều khai báo không bằng tổng số phần tử. Hàm numel() sẽ trả về tổng số phần tử của ma trận và hàm size() sẽ trả về số chiều của ma trận.

Ví dụ 2.30

Cho ma trận 9×10 có giá trị giảm dần từ 90 đến 1, sử dụng Matlab tạo ma trận có số hàng giảm dần từ 10 đến 1.

Giải: Chúng ta thực hiện các lệnh sau ở cửa sổ lệnh của Matlab.

```
A=ones(9,10);
A(1:numero(A))=numero(A):-1:1;
for idx = 9:-3:3
    reshape(A,idx,numero(A)/idx)
end
```

2.1.2.15 Phép toán trên từng phần tử ma trận

Với Matlab, chúng ta có hai cách để làm việc với ma trận, hoặc với toàn bộ ma trận hay với từng phần tử ma trận. Chúng ta có thể thực hiện các phép cộng, trừ, nhân, chia, lũy thừa với từng phần tử trong ma trận với một toán tử, bằng cách thêm dấu . trước các toán tử, cụ thể như cho trong Bảng 2.1.

Bảng 2.1: Các toán tử trên Matlab

	Toán tử	Toán tử áp dụng cho ma trận	Toán tử áp dụng cho từng phần tử trong ma trận	Yêu cầu
Phép cộng	+	+	.+	
Phép trừ	-	-	.-	
Phép nhân	*	*	.*	Kích thước của hai ma trận là phải giống nhau
Phép chia	/	/	./	
Lũy thừa	^	^	.^	
Phép chia nghịch đảo		\		$A./B = B.\A$ (Hạn chế sử dụng vì không quy chuẩn)

Ví dụ 2.31

Cho ma trận A có giá trị như sau: $A = [1 \ 2 \ 3; 4 \ 5 \ 6]$, thực hiện các phép toán sau:

- $B = 3*A + 1$
- $C = 1./A$
- $D = A.*C$
- $E = D - A$
- $F = 1 - A$
- $G = A.^2$

g. $H = A^2$

h. $K = A \cdot A'$

Giải: Thực hiện trực tiếp trên cửa sổ lệnh như sau. Chúng ta sẽ thấy Matlab sẽ báo lỗi với phép toán ở câu g.

```
>> A = [ 1 2 3; 4 5 6]
```

A =

$$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{matrix}$$

```
>> B=2*A + 1
```

B =

$$\begin{matrix} 3 & 5 & 7 \\ 9 & 11 & 13 \end{matrix}$$

```
>> C=1./A
```

C =

$$\begin{matrix} 1.0000 & 0.5000 & 0.3333 \\ 0.2500 & 0.2000 & 0.1667 \end{matrix}$$

```
>> D= A.*C
```

D =

$$\begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

```
>> E = D - A
```

E =

$$\begin{matrix} 0 & -1 & -2 \\ -3 & -4 & -5 \end{matrix}$$

```
>> F = 1 - A
```

F =

$$\begin{matrix} 0 & -1 & -2 \\ -3 & -4 & -5 \end{matrix}$$

```
>> G = A.^2
```

G =

$$\begin{matrix} 1 & 4 & 9 \\ 16 & 25 & 36 \end{matrix}$$

```

>> H = A^2
Error using ^ (line 51)
Incorrect dimensions for raising a matrix to a power. Check that the matrix is square and the power is a
scalar. To perform elementwise
matrix powers, use '!^'.
>> G = A.^2
G =
1   4   9
16  25  36
>> K = A*A'
K =
14   32
32   77

```

2.1.2.16 Phép nhân ma trận

Phép nhân giữa ma trận A có kích thước $m \times n$ và ma trận B có kích thước $n \times p$ tạo ra ma trận C có kích thước $m \times p$.

Nếu A là ma trận vuông có kích thước là n , phép lũy thừa ma trận A có thể thực hiện bằng cách nhân liên tiếp A cho chính nó k lần.

Ví dụ 2.32

Cho ma trận $A = [1 \ 2 \ 3; 4 \ 8 \ -4; 6 \ 9 \ 4]$, thực hiện các tính toán như sau:

- $B = A^{-1/2}$
- $C = A.*B$
- $D = A*B$
- $E = A.*A.*A$
- $F = A.^3$
- $G = A^3$

Giải: Khi thực hiện các lệnh trên chúng ta không sử dụng dấu ';' ở cuối lệnh để quan sát kết quả trả về trên cửa sổ lệnh.

```

>> A = [1 2 3; 4 8 -4; 6 9 4]
A =
1   2   3
4   8  -4
6   9   4

```

```

>> B = A - 1/2

B =

0.5000 1.5000 2.5000
3.5000 7.5000 -4.5000
5.5000 8.5000 3.5000

>> C = A.*B

C =

0.5000 3.0000 7.5000
14.0000 60.0000 18.0000
33.0000 76.5000 14.0000

>> D = A*B

D =

24.0000 42.0000 4.0000
8.0000 32.0000 -40.0000
56.5000 110.5000 -11.5000

>> E = A.*A.*A

E =

1   8   27
64  512  -64
216 729   64

>> F = A^3

F =

249      477      -71
-60      -12     -252
534     1074     -290

>> G = A.^3

G =

1   8   27
64  512  -64
216 729   64

```

2.1.2.17 Phép chia ma trận

Để thực hiện phép chia ma trận trong Matlab, $\mathbf{Y} = \mathbf{X}\mathbf{W}$, tìm \mathbf{X} với \mathbf{X} là ma trận $n \times d$, \mathbf{W} là ma trận $d \times k$ và \mathbf{Y} là ma trận $n \times k$, chúng ta có hai cách như sau:

- Sử dụng toán tử /

- Sử dụng hàm **inv()**

Nếu X là khai nghịch, thì $\mathbf{W} = \text{inv}(\mathbf{X}) * \mathbf{Y}$. Nếu $n \neq d$, ta vẫn có thể tìm \mathbf{W} thông qua phép giả nghịch đảo, $\mathbf{W} = \text{inv}(\mathbf{X}' * \mathbf{X}) * \mathbf{X}' * \mathbf{Y}$, hay dùng hàm **pinv()**, cụ thể là $\mathbf{W} = \text{pinv}(\mathbf{X}) * \mathbf{Y}$. Lưu ý rằng: $\mathbf{X} \setminus \mathbf{Y} = (\mathbf{Y}' / \mathbf{X}')$.

Ví dụ 2.33

Chạy từng dòng lệnh Matlab sau và giải thích ý nghĩa của từng dòng lệnh, cho nhận xét.

- $x = \text{randn}(3,3)$
- $y = \text{randn}(3,3)$
- $x \setminus y$
- $(y' / x)'$

Giải: Thực hiện trực tiếp trên cửa sổ lệnh của Matlab như sau:

```
>> x = randn(3,3)

x =
-1.7947  0.1001 -0.6003
 0.8404 -0.5445  0.4900
-0.8880  0.3035  0.7394

>> y = randn(3,3)

y =
 1.7119 -0.8396  0.9610
 -0.1941  1.3546  0.1240
 -2.1384 -1.0722  1.4367

>> x \ y

ans =
-0.2938  0.4053 -0.9411
-2.2030 -1.9929 -0.6930
-2.3406 -0.1452  1.0973

>> (y' / x)'

ans =
-0.2938  0.4053 -0.9411
-2.2030 -1.9929 -0.6930
-2.3406 -0.1452  1.0973
```

2.1.2.18 Đại số tuyến tính

Là ngôn ngữ dành cho ma trận, Matlab cung cấp rất nhiều hàm đại số tuyến tính, cụ thể:

- Hàm **magic()**
- Hàm **det()**
- Hàm **rank()**
- Hàm **rref()**
- Hàm **trace()**
- Hàm **cov()**
- Hàm **orth()**
- Hàm **null()**
- Hàm **chol()**
- Hàm **eigs()**
- Hàm **svd()**
- Hàm **qr()**

Cú pháp và cách sử dụng cụ thể của từng hàm có thể tham khảo thông qua lệnh **help()** với tên hàm ở trên. Xem thêm ở Ví dụ 2.34.

Ví dụ 2.34

Thực hiện các dòng lệnh sau trên Matlab và giải thích ý nghĩa từng dòng lệnh và kết quả đạt được:

- a. A = magic(5);
- b. B = det(A);
- c. C = rank(A);
- d. D = rref(A);
- e. E = trace(A);
- f. F = cov(A);
- g. G = orth(A);
- h. H = null(A);
- i. I = chol(A*A');
- j. [evecs,evals] = eigs(A);
- k. [U,S,V] = svd(A);

I. $[Q,R] = qr(A)$;

Giải: Sử dụng hàm **help** để xem chức năng của từng hàm và nhận xét kết quả.

2.1.2.19 Mảng đa chiều

Mảng số trong Matlab có thể mở rộng số chiều lớn hơn 2. Chúng ta có thể dùng các hàm như: zeros(), ones(), rand(), để tạo ra ma trận có nhiều chiều. Để tìm số chiều của ma trận, chúng ta có thể dùng hàm **ndims()**.

Cú pháp:

`ndims(A)`

% A là ma trận cần tìm số chiều

Ví dụ 2.35

Thực hiện các lệnh sau trong Matlab và giải thích từng dòng lệnh.

`A = magic(3)`

`A(:,:,2) = magic(3)'`;

`size(A)`

`A(:,:,2) = [];`

`ndims(A)`

Giải: Sử dụng hàm **help()** để tìm hiểu chức năng của từng hàm.

```
>> A = magic(3)
```

```
A =
```

```
8 1 6  
3 5 7  
4 9 2
```

```
>> A(:,:,2) = magic(3)'
```

```
A(:,:,1) =
```

```
8 1 6  
3 5 7  
4 9 2
```

```
A(:,:,2) =
```

```
8 3 4  
1 5 9  
6 7 2
```

```
>> size(A)
```

```

ans =
3   3   2
>> ndims(A)
ans =
2

```

Ví dụ 2.36

Thực hiện các dòng lệnh sau trong Matlab và giải thích ý nghĩa của từng dòng lệnh:

- $A = \text{ones}(3,5,9,2,2);$
- $B = \text{cat}(3,[1\ 2\ ;\ 4\ 5],[3\ 2\ ;\ 1\ 1])$
- $C = \text{repmat}([1\ 2\ ;\ 3\ 4],[2,2,2,2]);$
- $D = C(1,1,1,1)$
- $A(1,2,2,1) = 99;$
- $F = \text{mean}(C,4);$
- $G = \text{max}(A,[],3);$

Giải: Giải thích các dòng lệnh như sau:

- Tạo ma trận 5 chiều A.
- Tạo ma trận B bằng cách kết nối hai ma trận theo chiều thứ 3.
- Tạo ma trận B bốn chiều với ma trận [1 2 ; 3 4].
- Trích giá trị của phần tử có tọa độ (1,1,1,1) của ma trận C.
- Gán phần tử ở vị trí (1,2,2,1) của ma trận A giá trị 99.
- Lấy trung bình theo chiều thứ 4 của ma trận C.
- Tìm giá trị lớn nhất của ma trận A ở chiều thứ 3.

2.1.3 Các lệnh điều khiển

2.1.3.1 Lệnh if, else, elseif

Cú pháp của hàm if có dạng như sau:

Cú pháp:

```

if biểu thức logic
    lệnh thực hiện
elseif biểu thức logic
    lệnh thực hiện

```

```
elseif
    lệnh thực hiện
end
```

Ví dụ 2.37

Nhập vào tháng và năm, tính số ngày của tháng.

Giải: Trong 12 tháng, các tháng có 31 ngày là tháng 1, 3, 5, 7, 8, 10 và 12. Tháng 2 nếu thuộc năm chia hết cho 4 là 28 ngày và ngược lại là 29 ngày. Các tháng còn lại là 30 ngày.

```
clc

m = input('Nhập tháng =');
y = input('Nhập năm =');

month31 = [1 3 5 7 8 10 12];
month30 = [4 6 9 11];

if any(m == month31)
    disp('Tháng có 31 ngày');
elseif any(m==month30)
    disp('Tháng có 29 ngày');
elseif mod(y,4)==0
    disp('Tháng có 29 ngày');
else
    disp('Tháng có 28 ngày');
end
```

2.1.3.2 Lệnh switch

Lệnh **switch** cho phép thực hiện những trường hợp mà phải thực hiện bằng nhiều lệnh elseif. Cú pháp của lệnh switch như sau:

Cú pháp:

```
switch biểu thức lựa chọn
    CASE biểu thức trường hợp,
        thực thi các lệnh
    CASE biểu thức trường hợp
        thực thi các lệnh
    ...
    OTHERWISE,
        thực thi các lệnh
END
```

Ví dụ 2.38

Lập trình nhập vào tháng và năm và trả lời số ngày trong tháng sử dụng lệnh switch và case.

Giải: Giải thuật là tương tự như Ví dụ 2.37 trên.

```
m = input('Nhập tháng =');  
y = input('Nhập năm=');  
  
switch m  
    case {1, 3, 5, 7, 8, 10, 12}  
        disp('Tháng có 31 ngày');  
    case 2  
        if mod(y,4)==0  
            disp('Tháng có 29 ngày');  
        else  
            disp('Tháng có 28 ngày');  
        end  
    otherwise  
        disp('Tháng có 30 ngày');  
end
```

2.1.3.3 Lệnh for

Lệnh **for** là một trong những lệnh vòng lặp chủ đạo của Matlab. Lệnh for có cấu trúc đơn giản như sau:

Cú pháp:

```
for biến chạy = giá trị bắt đầu: giá trị nhảy: giá trị cuối  
    lệnh thực thi  
END
```

Vòng lặp for thường dùng khi chỉ số bắt đầu và chỉ số kết thúc vòng lặp là rõ ràng. Nhược điểm của vòng lặp for là sẽ tốn nhiều thời gian khi chiều dài của vector hay số chiều của ma trận lớn.

Ví dụ 2.39

Sử dụng vòng lặp for để tính $n!$.

- Cách 1: dùng vòng for.
- Cách 2: không dùng vòng lặp for.

Giải: Định nghĩa của giai thừa là $n! = 1 \times 2 \times \dots \times n$.

```
n = 100;  
  
% Cách 1  
GT = 1;  
for i = 1:n  
    GT = GT.*i;
```

```

end
disp(strcat(num2str(n),'!=',num2str(GT)));
% Cách 2
GT = prod(1:n);
disp(strcat(num2str(n),'!=',num2str(GT)));

```

So Cách 1 với Cách 2, ta thấy rằng Cách 2 chỉ thực hiện trong vòng một dòng lệnh và đơn giản hơn rất nhiều.

Ví dụ 2.40

Cho vector nhị phân có chiều dài N có các phần tử có phân bố đều. Hãy tìm các giá trị bằng 1 trong vector và thay thế bằng 2.

- Dùng lệnh for.
- Không dùng lệnh for tạo ma trận rỗng.

Giải: Để tạo vector có phân bố đều chúng ta dùng hàm randi(). Chương trình Matlab trong hai trường hợp là như sau:

```

% Khởi tạo ma trận
N = 10;
x = randi([0 1], 1, N)

% Dùng lệnh for
for idx = 1:N
    if x(idx)==1
        x(idx)=2;
    end
end

% Không dùng lệnh for
x(x==1)=2

```

2.1.3.4 Lệnh while

Lệnh **while** thường được dùng để thực thi một đoạn mã chương trình cho đến khi điều kiện dừng thỏa mãn. Điểm khác biệt chính của vòng lặp **while** so với vòng lặp **for** là điều kiện dừng của vòng lặp **while** thường phức tạp hơn và không đơn giản là lặp lại như vòng lặp **for**.

Cú pháp:

while biểu thức logic

thực thi lệnh

END

Trong một số trường hợp, chúng ta có thể dùng vòng lặp **while** với lệnh thoát vòng lặp **break** với cú pháp như sau:

Cú pháp:

```
while(true)  
    % Thực hiện lệnh  
    if(condition)  
        break;  
    end  
end
```

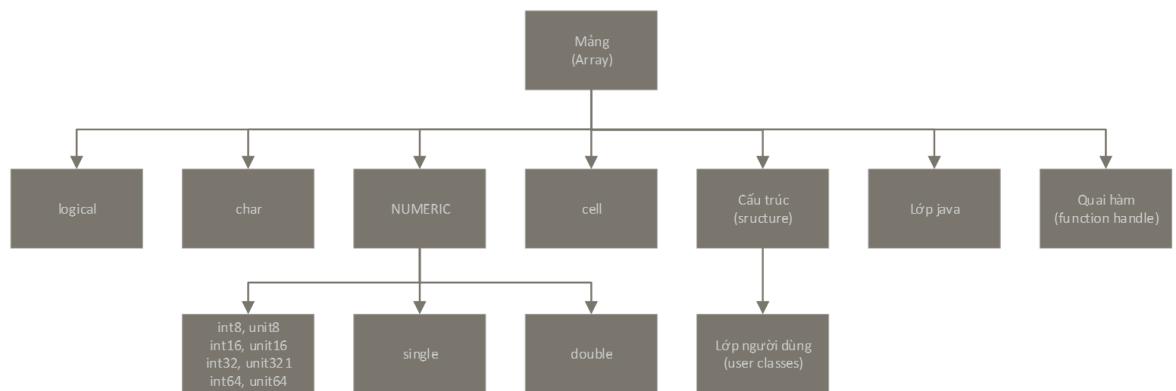
Ví dụ 2.41

Sử dụng vòng lặp while viết chương trình tính n!

Giải: Chúng ta dùng lệnh input() để nhập n từ bàn phím.

```
n = input('Nhập vào n =');  
GT = 1;  
  
if n == 0  
    GT = 1;  
else  
    k=1;  
    while k <= n  
        GT = GT.*k;  
        k = k + 1;  
    end  
end  
disp(strcat(num2str(n),'!=',num2str(GT)));
```

2.1.4 Chuỗi



Hình 2.2: Phân loại mảng trong Matlab.

Bên cạnh mảng, Matlab có cấu trúc dữ liệu ở hai dạng: mảng cell và cấu trúc. Chúng ta sẽ nghiên cứu kỹ ở phần sau. Sau đây là danh mục các hàm cho mảng và cấu trúc dữ liệu như sau:

- repmat, ischar, isletter, isspace, upper, lower, strtrim, deblank,
- isstrprop, char, abs, dec2hex, hex2dec, bin2dec, dec2bin, num2str,
- mat2str, str2num, strcat, strvcat, sortrows, strjust, sprintf, fprintf,
- cell, iscell, num2cell, mat2cell, cell2mat, cellstr, iscellstr, cellfun,
- strcmp, strcmpl, strncmp, strncmp, strfind, strmatch, strtok,
- intersect, union, setdiff, setxor, ismember, all, any, perms, issorted,
- unique, struct, isstruct, fieldnames, isfield, orderfields, rmfield,
- isvarname, genvarname, vertcat, cell2struct, struct2cell.

2.1.4.1 Mảng ký tự

Chuỗi trong Matlab là ma trận các ký tự, do đó chúng ta có thể tương tác và xử lý chuỗi trong Matlab tương tự như cách chúng ta xử lý ma trận số. Hãy xem xét Ví dụ 2.42.

Ví dụ 2.42

Thực hiện các dòng lệnh sau trong Matlab và xem xét kết quả.

- a. A = 'This is Test String #1! '
- b. B = A(1:5)
- c. C = [A ; A]
- d. D = repmat('@!',2,5)
- e. E = 'z':-1:'a'
- f. check = ischar(A)
- g. F = isletter(A(1:6))
- h. G = isspace(A(1:6))
- i. H = upper(A)
- j. I = lower(A)
- k. J = strtrim(A)
- l. K = deblank(A)

Giải: Nhập các dòng lệnh trên vào cửa sổ lệnh của Matlab và xem xét kết quả:

```
A =
This is Test String #1!
B =
This
C =
This is Test String #1!
This is Test String #1!
```

```

D =
@!@!@!@!@!
@!@!@!@!@!
E =
zyxwvutsrqponmlkjihgfedcba
check =
    1
F =
    0   1   1   1   1   0
G =
    1   0   0   0   0   1
H =
THIS IS TEST STRING #1!
I =
this is test string #1!
J =
This is Test String #1!
K =
This is Test String #1!

```

Qua Ví dụ 2.42, chúng ta học thêm chức năng của các hàm: **repmat()**, **ischar()**, **isletter()**, **isspace()**, **upper()**, **lower()**, **strtrim()**, **deblank()**. Để hiểu thêm về các hàm này, chúng ta sử dụng hàm **help()** với các hàm như hướng dẫn ở phần trên. Lưu ý là khi xử lý chuỗi, chúng ta muốn phân biệt đó là ký tự, là số, là số La Mã, là khoảng trắng,... và hàm **isstrprop()** giúp chúng ta làm điều đó.

Ví dụ 2.43

Thực hiện các dòng lệnh trên Matlab và xem xét kết quả:

- a. str = 'a1!'
- b. A = isstrprop(str,'punct') % dấu câu
- c. B = isstrprop(str,'alphanum') % ký tự số hoặc chữ
- d. C = isstrprop(str,'digit') % số thập phân
- e. D = isstrprop('3A','xdigit') % số thập lục phân

Giải: Kết quả của các dòng lệnh là như sau:

```

str =
a1!
A =
    0   0   0   1
B =
    0   1   1   0
C =
    0   0   1   0
D =
    1   1

```

Trong một số trường hợp, chúng ta cần xử lý với mã ASCII của ký tự. Matlab cung cấp hàm **char()** và hàm **abs()** cho việc chuyển từ ký tự qua mã ASCII và ngược lại.

Ví dụ 2.44

Viết chương trình nhập tên của bạn. Chuyển thành mã ASCII và viết hoa toàn bộ.

Giải: Chúng ta biết rằng mã ASCII của ký tự hoa và ký tự thường cách nhau 32.

```
s='vo nguyen quoc bao'
abs(s)
char(abs(s)-32)

s =
'vo nguyen quoc bao'

ans =
118 111 32 110 103 117 121 101 110 32 113 117 111 99 32 98 97 111

ans =
'VO NGUYEN QUOC BAO'
```

Để chuyển đổi từ số hệ thập phân sang các hệ cơ số khác như nhị phân, thập lục phân và ngược lại, Matlab cung cấp cho chúng ta các hàm sau: **dec2hex()**, **hex2dec()**, **dec2bin()**, and **bin2dec()**. Chú ý rằng Matlab lưu trữ các số nhị phân và số thập lục phân ở dạng chuỗi.

Ví dụ 2.45

Thực hiện các hàm sau và quan sát kết quả:

- A = dec2hex(211)
- B = hex2dec('D3')
- C = dec2bin(211)
- D = bin2dec('11010011')

Giải:

```
A =
D3

B =
211

C =
11010011

D =
211
```

Ngoài ra, chúng ta còn có hàm **num2str()**, **mat2str()** và **str2num()** để chuyển đổi số thành chuỗi và ngược lại.

Ví dụ 2.46

Thực hiện các hàm Matlab sau và nhận xét:

- $A = \text{num2str}([1:5;1:5])$
- $B = \text{mat2str}([1:5;1:5])$
- $C = \text{str2num}('44')$

Giải:

```
A =  
1 2 3 4 5  
1 2 3 4 5  
B =  
[1 2 3 4 5;1 2 3 4 5]  
C =  
44
```

Ví dụ 2.47

Kiểm chứng chức năng của các hàm `strcat()`, `strvcat()`, `sortrows()` và `strjust()` thông qua 3 dòng lệnh sau:

- $C = \text{strvcat}('hello','this','is','a','test')$
- $D = \text{sortrows}(C)$
- $E = \text{strjust}(C)$

Giải:

```
% Nối chuỗi theo hàng  
C = strvcat('hello','this','is','a','test')  
C =  
hello  
this  
is  
a  
test  
  
% Xếp chuỗi theo thứ tự ABC  
D = sortrows(C)  
D =  
a  
hello  
is  
test  
this  
  
% Điều chỉnh mang ký tự  
E = strjust(C)  
E =  
hello  
this  
is
```

```
a  
test
```

2.1.4.2 Xuất chuỗi

Để xuất chuỗi ra màn hình hay lưu dưới dạng file trong Matlab, ta dùng hàm **sprint()** hay hàm **fprint()**. Lưu ý là hàm **sprint()** tương tự như **fprint()** nhưng **sprint()** chỉ xuất ra chuỗi hay vector ký tự trên màn hình.

Cú pháp:

```
str = sprintf(formatSpec,A1,...,An)  
[str,errmsg] = sprintf(formatSpec,A1,...,An)  
str = sprintf(literalText)
```

formatSpec là chuỗi ký tự xuất ra màn hình theo mong muốn. Thông tin cụ thể hơn về hàm **sprint()** và **fprint()** có thể tìm hiểu ở phần mềm Matlab thông qua lệnh: **doc sprint** và **doc fprintf**.

Ví dụ 2.48

Thực hiện đoạn mã Matlab sau với hàm **sprintf()**.

```
formatSpec = "Thời gian hiện tại là: Năm: %d Tháng: %d Ngày: %d, %d Giờ, %d Phút, %d Giây";  
str = sprintf(formatSpec,clock)
```

Giải: Kết quả trả về như sau:

```
str =  
"Thời gian hiện tại là: Năm: 2020 Tháng: 4 Ngày: 10, 13 Giờ, 50 Phút, 1.565000e+01 Giây"
```

Ví dụ 2.49

Hãy biểu diễn số π sau dấu thập phân 10 số và so sánh với lệnh **format long**:

Giải: Chúng ta sử dụng lệnh **sprint**. Số π trong Matlab là **pi**.

```
>> str = sprintf('%1.10f',pi)  
  
str =  
'3.1415926536'  
  
>> format long  
>> pi  
  
ans =  
3.141592653589793
```

2.1.4.3 Mảng cell

Mảng cell (cell array) là một loại dữ liệu có cấu trúc trong Matlab. Mảng cell có thể chứa bất kỳ dữ liệu nào từ ma trận số với kích thước khác nhau tới mảng ký tự và mảng cell,...

Chúng ta thường sử dụng mảng cell để lưu trữ dữ liệu khi thực hiện mô phỏng, ví dụ toàn bộ tính chất của đối tượng cần mô phỏng.

Để tạo mảng cell, chúng ta dùng dấu {} hoặc dùng hàm **cell()**.

Ví dụ 2.50

Hãy tạo một cell lưu trữ toàn bộ thông tin của một sinh viên bao gồm: họ và tên, năm sinh, mã số sinh viên và chuyên ngành.

Giải:

```
>> SV = {"Tran Van A" '1998' 'D19CV01' 'DTVT'}
```

SV =

1×4 cell array

```
{'Tran Van A'} {'1998'} {'D19CV01'} {'DTVT'}
```

Ví dụ 2.51

Hãy xóa mã số sinh viên ở Ví dụ 2.50.

Giải:

```
>> SV(3)=[]
```

SV =

1×3 cell array

```
{'Tran Van A'} {'1998'} {'DTVT'}
```

Chúng ta có thể thực hiện nhiều phép biến đổi trên mảng cell tương tự như thực hiện trên ma trận, như chuyển vị, nối mảng cell, xóa mảng cell, lặp lại,... kể cả truy nhập các phần tử trong mảng cell. Trong một số trường hợp, để chắc chắn đó là mảng cell, chúng ta dùng hàm **iscell()**.

Chúng ta có thể dùng hàm **num2cell()**, **mat2cell()**, hay **cell2mat()** để chuyển đổi giữa ma trận và mảng cell.

Ví dụ 2.52

Thực hiện đoạn mã sau và quan sát kết quả:

```
% Chuyển đổi [1,2,3,4,5] to {[1],[2],[3],[4],[5]}
A = num2cell(1:5)
% Chia ma trận ones(4,8) thành 6 cell
B = mat2cell(ones(4,8),[2,2],[3,3,2])
% Gom ma trận
C = cell2mat(B)
```

Giải: Kết quả quan sát ở cửa sổ lệnh là như sau.

```
A =  
[1] [2] [3] [4] [5]  
B =  
[2x3 double] [2x3 double] [2x2 double]  
[2x3 double] [2x3 double] [2x2 double]  
C =  
1 1 1 1 1 1 1  
1 1 1 1 1 1 1  
1 1 1 1 1 1 1  
1 1 1 1 1 1 1
```

Chúng ta có thể đổi từ ma trận ký tự qua mảng chuỗi và ngược lại dùng hàm **cellstr()** và hàm **char()**.

Ví dụ 2.53

Thực hiện nối họ, tên lót và tên: “Tran” “Van” “Tien” và chuyển đổi kết quả thu được thành mảng cell.

Giải: Chúng ta có kết quả như sau.

```
A = strvcat('Tran','Van','Tien')  
B = cellstr(A)  
check = iscellstr(B)  
C = char(B)  
A =  
Tran  
Van  
Tien  
B =  
'Tran'  
'Van'  
'Tien'  
check =  
1  
C =  
Tran  
Van  
Tien
```

2.1.4.4 So sánh chuỗi

Khi xử lý trên chuỗi, chúng ta thường có các tác vụ sau: tìm một ký tự hay một tập ký tự trên chuỗi, so sánh hai chuỗi và nối các chuỗi.

Bảng 2.2: Các hàm xử lý trên chuỗi.

TT	Hàm	Tác vụ
1	strcmp()	So sánh toàn bộ hai chuỗi
2	strcmpi()	So sánh hai chuỗi nhưng không quan tâm chữ viết hoa hay viết thường
3	strncmp()	So sánh từng phần hai chuỗi

4	strncmpi()	Giống lệnh strcmp() nhưng không quan tâm ký tự hoa hay thường
5	strfind()	Tìm một chuỗi trong một chuỗi khác, trả về số lần xuất hiện
6	strtok()	Tách chuỗi ra làm hai phần ngăn cách bởi khoảng trắng. Một ứng dụng dễ thấy là tách họ ra từ họ và tên
7	contains()	Trả về giá trị logic nếu tìm thấy chuỗi muốn tìm trong chuỗi cho trước

Ví dụ 2.54

Thực hiện đoạn mã sau trên Matlab để kiểm chứng sự khác nhau giữa các hàm so sánh chuỗi:

```
A = 'testString'
```

```
% So sánh hai chuỗi
```

```
test1 = strcmp(A,'testString')
```

```
% So sánh hai chuỗi không phân biệt viết thường hay viết hoa
```

```
test2 = strncMPI(A,'TESTstring')
```

```
% So sánh hai chuỗi với 4 ký tự đầu tiên
```

```
test3 = strncmp(A,'testFoo',4)
```

```
% So sánh hai chuỗi cho 4 ký tự đầu tiên không phân biệt chữ thường hay chữ hoa
```

```
test4 = strncMPI(A,'TEST',4)
```

Giải: Để dễ dàng thấy được là kết quả của test1, test2, test3 và test4 là 1.

Ví dụ 2.55

Tìm chuỗi ‘ctga’ trong chuỗi ‘actgcgctgacgctgatacacacggagctgacgacttgaggacgagc’

Giải: Chúng ta dùng hàm **strfind()** như sau:

```
>> str = 'actgcgctgacgctgatacacacggagctgacgacttgaggacgagc'  
A = strfind(str,'ctga')
```

```
str =
```

```
'actgcgctgacgctgatacacacggagctgacgacttgaggacgagc'
```

```
A =
```

```
7 13 27 34
```

Ví dụ 2.56

Tìm từ ‘foo’ trong mảng cell {‘foobar’, ‘bar’, ‘barfoo’, ‘foofoo’}

Giải: Ta dùng hàm **strmatch()**.

```
>> str2 = {'foobar','bar','barfoo','foofoo'}  
B = strmatch('foo',str2)
```

```
str2 =  
1×4 cell array  
{'foobar'}  {'bar'}  {'barfoo'}  {'foofoo'}  
  
B =  
1  
4
```

2.1.4.5 Các toán tử trên tập

Matlab cho phép chúng ta thực hiện các toán tử trên tập, ví dụ như:

Cú pháp:

```
union()  
intersect()  
setdiff()  
setxor()  
ismember()  
unione()  
mysetdiff()  
unique()  
perms()
```

Ví dụ 2.57

Thực hiện dòng lệnh sau và giải thích lệnh thực thi:

- a. set1 = 1:2:9
- b. set2 = 1:4
- c. int = intersect(set1, set2)
- d. uni = union(set1, set2)
- e. dif = setdiff(set1, set2)
- f. xor = setxor(set1, set2)
- g. check = ismember(3, set1)
- h. set3 = {'alpha', 'beta', 'gamma'}
- i. set4 = {'delta', 'beta', 'epsilon'}

- j. `intc = intersect(set3,set4)`
k. `check2 = ismember('delta',set4)`

Giải: Kết quả trên Matlab như sau:

```
set1 =
1   3   5   7   9

set2 =
1   2   3   4

int =
1   3

uni =
1   2   3   4   5   7   9

dif =
5   7   9

xor =
2   4   5   7   9

check =
logical
1

set3 =
1×3 cell array
{'alpha'}  {'beta'}  {'gamma'}
```

```
set4 =
1×3 cell array
{'delta'}  {'beta'}  {'epsilon'}
```

```
intc =  
1×1 cell array  
{'beta'}  
  
check2 =  
logical  
1
```

Khi làm việc trên mảng logic, hàm **all()** và **any()** sẽ giúp chúng ta thực hiện phép AND và phép OR của tất cả các phần tử trên mảng, cụ thể là phép all() sẽ trả về 1 nếu tất cả giá trị của mảng đều là 1 và phép any() sẽ trả về 1 nếu chỉ cần 1 giá trị của mảng là 1.

Ví dụ 2.58

Kiểm tra tất cả các số 1 3 5 7 có phải là số nguyên tố không? Kiểm tra có số nguyên tố nào trong 4 số 1 3 5 7 không?

Giải: Để kiểm tra số nguyên tố, ta dùng hàm **isprime()**.

```
>> forall = all(isprime(1:2:7))  
exists = any(isprime(1:2:7))  
  
forall =  
logical  
0  
  
exists =  
logical  
1
```

Trong một số trường hợp, chúng ta cần tìm một phần tử xuất hiện ít nhất một lần trong ma trận hay trong chuỗi. **unique()** là hàm mà chúng ta cần.

Ví dụ 2.59

Nhập vào họ của 3 người trong gia đình bạn, tìm các họ có trong gia đình bạn.

Giải: Chúng ta dùng hàm **unique()**.

```
>> FamilyNames = {'Nguyen','Vo','Tran','Vo'}  
  
FamilyNames =  
1×4 cell array
```

```

{'Nguyen'}  {'Vo'}  {'Tran'}  {'Vo'}
>> UniqueNames = unique(FamilyNames)

UniqueNames =
1×3 cell array

{'Nguyen'}  {'Tran'}  {'Vo'}

```

2.1.4.6 Cấu trúc

Bên cạnh ma trận và mảng cell, Matlab còn hỗ trợ chúng ta mảng có cấu trúc hay gọi tắt là cấu trúc (structs). Cấu trúc về mặt tổ chức dữ liệu sẽ chặt chẽ hơn và chúng ta có thể truy cập dữ liệu theo tên của trường trong cấu trúc. Chúng ta có thể tạo cấu trúc dữ liệu trong Matlab theo hàm **struct()**. Lưu ý tên các trường phải đặt theo quy cách tương tự như đặt tên biến. Để chắc chắn tên trường trong cấu trúc là hợp lệ, ta cần dùng hàm **isvarname()**.

Khi làm việc trên cấu trúc, ta có các hàm sau: **isstruct()**, **fieldnames()**, **orderfields()** và **rmfield()**.

Chúng ta có thể tạo cấu trúc từ mảng cell và ngược lại sử dụng hàm **cell2struct()** và **struct2cell()**.

Ví dụ 2.60

Hãy khai báo một cấu trúc thông tin của sinh viên gồm ba thành phần gồm họ và tên, năm sinh và giới tính.

- Kiểm tra có phải là cấu trúc không?
- Liệt kê ra các trường của cấu trúc
- Kiểm tra ‘Gioi_Tinh’ có phải là một trường trong cấu trúc sinh viên không?
- Xếp thứ tự các trường
- Thêm một trường ‘Noi_Sinh’
- Kiểm tra tên trường “Ho_va_Ten” có hợp lệ không?
- Tạo ra một mảng cấu trúc với sinh viên thứ hai.
- Lấy năm sinh của hai sinh viên

Giải: Chúng ta bắt đầu với hàm **struct()**.

```

SV=struct('Ho_va_Ten','Tran Van A','Nam_Sinh',1990,'Gioi_Tinh','Nam')
isstruct(SV)
fieldnames(SV)
isfield(SV,'Gioi_Tinh')
SV = orderfields(SV)
SV.Noi_Sinh = 'Khanh Hoa'
SV = rmfield(SV,'Noi_Sinh')

```

```
SV(2).Ho_va_Ten = 'Tran Thi B'  
SV(2).Nam_Sinh = 1991  
SV(2).Gioi_Tinh = 'Nu'  
SV.Nam_Sinh
```

SV =

struct with fields:

```
Ho_va_Ten: 'Tran Van A'  
Nam_Sinh: 1990  
Gioi_Tinh: 'Nam'
```

ans =

```
logical  
1
```

ans =

3×1 cell array

```
{'Ho_va_Ten'}  
{'Nam_Sinh' }  
{'Gioi_Tinh'}
```

ans =

```
logical  
1
```

SV =

struct with fields:

```
Gioi_Tinh: 'Nam'  
Ho_va_Ten: 'Tran Van A'  
Nam_Sinh: 1990
```

SV =

struct with fields:

```
Gioi_Tinh: 'Nam'  
Ho_va_Ten: 'Tran Van A'  
Nam_Sinh: 1990  
Noi_Sinh: 'Khanh Hoa'
```

SV =

struct with fields:

```
Gioi_Tinh: 'Nam'  
Ho_va_Ten: 'Tran Van A'  
Nam_Sinh: 1990
```

```
SV =
```

```
1×2 struct array with fields:
```

```
Gioi_Tinh  
Ho_va_Ten  
Nam_Sinh
```

```
SV =
```

```
1×2 struct array with fields:
```

```
Gioi_Tinh  
Ho_va_Ten  
Nam_Sinh
```

```
SV =
```

```
1×2 struct array with fields:
```

```
Gioi_Tinh  
Ho_va_Ten  
Nam_Sinh
```

```
ans =
```

```
1990
```

```
ans =
```

```
1991
```

2.1.5 Sửa lỗi (debug)

Gặp lỗi và sửa lỗi trong quá trình lập trình là một việc gần như không thể tránh khỏi. Có nhiều nguyên nhân dẫn đến lỗi lập trình, về cơ bản bao gồm: lỗi cú pháp, lỗi giải thuật và lỗi lập trình.

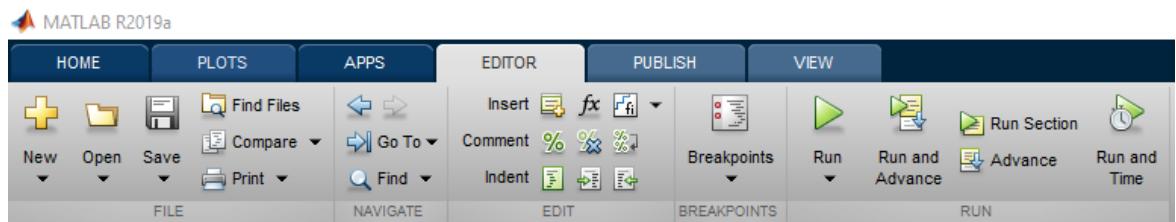
Trình soạn thảo (Editor) của Matlab cung cấp cho chúng ta một số phương pháp trực quan để phát hiện và giảm thiểu các lỗi trong quá trình lập trình (quá trình chuyển đổi từ ý tưởng và giải thuật thành mã Matlab).

Sau đây là các gợi ý để giảm thiểu lỗi trong quá trình lập trình:

- Tên biến phải đặt hợp lý, có tính gợi nhớ, phân biệt biến tham số và biến chạy, tránh sử dụng biến tham số làm biến chạy hay cho các biến chạy lồng vào nhau, hạn chế sử dụng biến toàn cục.
- Các chức năng khác nhau có tính lặp lại khi thực hiện nên tách ra thành hàm và thực hiện gọi hàm khi cần thiết.
- Sử dụng “smart indent”, phím tắt là Ctrl+I, trong trình soạn thảo của Matlab, để làm tường minh chương trình Matlab.

- Cảnh báo toàn bộ lỗi cú pháp nếu có do trình soạn thảo chỉ ra, lưu ý các dấu ~ màu đỏ và chỉ dấu tông vàng xuất hiện trong trình soạn thảo.
- Kiểm tra giá trị các biến có phù hợp hay không bằng cách sử dụng breakpoint.
- Tại các điểm breakpoint, kiểm tra số lượng, giá trị và định dạng của tham số đầu vào hàm có phù hợp với định nghĩa của hàm hay không? Kết quả trả ra của hàm về kích thước, về định dạng, về giá trị có theo như định nghĩa hay không?
- Lưu trữ các biến trong quá trình mô phỏng để phân tích lỗi.

Ngoài việc sử dụng chức năng debug trong trình biên soạn của Matlab, chúng ta có thể sử dụng các hàm sau: dbstop, dbquit, dbclear, dbstep, dbstep nlines, dbstep in, dbstep out, dbcont, dbstatus, keyboard, workspace, tic, toc, profile on, profile off, profile viewer.



Hình 2.3: Giao diện chức năng của trình soạn thảo.

Khi đã thực hiện xong việc biên tập mã nguồn Matlab của chương trình và chạy chương trình bằng nút “Run” trong chương trình như Hình 2.3, nếu kết quả trả về không như mong đợi hoặc chương trình có lỗi, chúng ta bắt đầu quy trình tìm lỗi:

- Sử dụng chế độ Run và dừng có điều kiện:
 - + Pause on Errors: sẽ dừng chương trình khi có lỗi xảy ra. Chế độ này cho phép kiểm tra giá trị của biến trước khi hay đang xảy ra lỗi, cho phép chúng ta dự đoán nguyên nhân lỗi. Chúng ta cũng có thể tận dụng chế độ này để biết được điểm lỗi, sau đó dùng breakpoints trước điểm lỗi và chạy chương trình ở chế độ từng bước (Step, Step in, Step out) để rà soát lỗi.
 - + Pause on Warnings: sẽ dừng chương trình khi có cảnh báo.
 - + Pause on NaN or Inf: sẽ dừng khi xuất hiện giá trị NaN hay Inf.
- Khi chương trình có nhiều hàm hoặc chia làm nhiều phần, chúng ta có thể thực hiện chế độ chạy từng phần - Run Section - để dễ dàng định vị vị trí lỗi. Nên tận dụng chức năng này để thực hiện chức năng chia vùng để tìm lỗi theo phương pháp “chia để trị”.

Trong một số trường hợp muốn tối ưu mã nguồn của chương trình, chúng ta sử dụng chức năng “Run and Time” để đo thời gian thực thi của các hàm và các lệnh trong chương trình và từ đó có hình thức tối ưu chương trình Matlab cho phù hợp.

2.1.6 Phần chú giải

Phần chú giải (hay chú thích) trong Matlab gọi là comment. Trong Matlab, phần chú giải bắt đầu bằng dấu %. Phần biên dịch của Matlab sẽ bỏ qua nội dung sau dấu %.

Khi lập trình, nên tiến hành ghi phần chú giải ngay sau lúc viết mã chương trình sẽ là hiệu quả nhất. Phần chú giải càng chi tiết sẽ giúp cho người lập trình và những người sử dụng hiểu rõ hơn về toàn bộ chương trình, về giải thuật, về cách sử dụng đoạn mã, hay về một hàm. Nếu có thể, chúng ta nên cung cấp thêm ví dụ sử dụng hay ví dụ minh họa trong phần chú giải để giúp người đọc/người sử dụng dễ hiểu hơn.

Nguyên tắc viết phần chú giải cho một hàm thường được viết sau lệnh **function**:

- Tóm tắt chức năng của hàm.
- Tham số đầu vào: các định nghĩa và quy định.
- Tham số đầu ra: các định nghĩa và quy định.
- Ví dụ sử dụng hàm (nếu có).

Phần chú giải xuất hiện khi chúng ta sử dụng hàm **help()**, hàm **doc()**, hay hàm **lookfor()**.

2.2 Vẽ trong Matlab

Matlab cung cấp rất nhiều hàm để biểu diễn dữ liệu từ hai chiều (2D) đến ba chiều (3D). Phần sau đây sẽ trình bày một số hàm vẽ của Matlab.

2.2.1.1 Vẽ đồ thị

Khi thu thập dữ liệu và muốn biểu diễn phân bố hai chiều, chúng ta dùng hàm **plot()**. Hàm **plot()** của Matlab cho phép chúng ta tùy chỉnh đồ thị thông qua các tính chất của đồ thị. Cú pháp của hàm **plot()** là như sau:

Cú pháp:

```
plot(X1,Y1,LineSpec1,...,Xn,Yn,LineSpecn)
```

% X1, Y1 là dữ liệu trực X và dữ liệu trực Y của đồ thị thứ nhất. Lưu ý là số chiều của X1 và Y1 phải giống nhau.

% LineSpecn là tham số để điều chỉnh đường vẽ, ví dụ như độ dày, loại nét vẽ.

Để điều chỉnh các đường vẽ, chúng ta có thể điều chỉnh độ dày, loại nét vẽ và các loại điểm vẽ thông qua các tham số “Linestyle”, “Linewidth” và “Marker” cho trong Bảng 2.3.

Bảng 2.3: Bảng tham số quy định nét vẽ, điểm vẽ và màu sắc cho hàm **plot()**.

Loại nét vẽ	Mô tả
-	Nét liền
--	Nét gạch đứt
:	Nét chấm

-.	Nét gạch chấm
Loại điểm vẽ	Mô tả
o	Dấu hình tròn
+	Dấu cộng
*	Dấu hoa thị
.	Dấu chấm
x	Dấu chéo
s	Dấu hình vuông
d	Dấu hình kim cương
^	Dấu tam giác hướng lên
v	Dấu tam giác hướng xuống
>	Dấu tam giác hướng bên phải
<	Dấu tam giác hướng bên trái
p	Dấu ngũ giác
h	Dấu lục giác
Màu sắc	Mô tả
y	Vàng
m	Tím
c	Xanh lơ
r	Đỏ
g	Xanh lá cây
b	Xanh nước biển
w	Trắng
k	Đen

Bên cạnh hàm plot() để vẽ biểu đồ hai chiều, Matlab còn hỗ trợ rất nhiều hàm vẽ biểu đồ khác để phục vụ nhu cầu biểu diễn dữ liệu như Bảng 2.4.

Bảng 2.4: Các hàm vẽ biểu đồ khác của Matlab.

TT	Hàm	Chức năng
1	semilogx()	Vẽ trực x theo log
2	semilogy()	Vẽ trực y theo log
3	loglog()	Vẽ hai trực theo log
4	plotyy()	Vẽ đồ thị theo hai trực y, một trái và một phải
5	plot3()	Vẽ đồ thị 3D
6	subplot()	Chia đồ thị ra nhiều đồ thị con
7	scatter()	Vẽ đồ thị phân tán
8	scatter3()	Vẽ đồ thị phân tán 2 chiều
9	plotmatrix()	Ma trận đồ thị phân tán
10	bar()	Vẽ biểu đồ cột
11	bar3()	Vẽ biểu đồ cột dạng 3D

12	bar3h()	Vẽ biểu đồ cột 3D theo phương ngang
13	histogram()	Vẽ biểu đồ tần suất
14	fplot()	Vẽ hàm theo quai hàm (function handle)
15	fplot3()	Vẽ đồ thị 3D theo quai hàm
16	fsurf()	Vẽ biểu đồ dạng sóng theo quai hàm
17	fcontour()	Vẽ biểu đồ đường bao theo quai hàm
18	fimplicit()	Vẽ đồ thị $f(x,y)=0$ theo quai hàm
19	fimplicit3()	Vẽ đồ thị $f(x,y,z)=0$ theo quai hàm
20	function_handle()	Trả về quai hàm

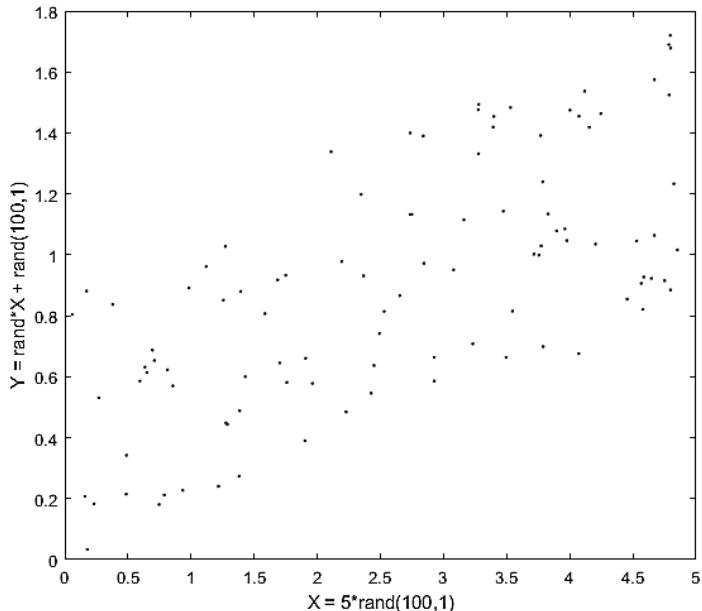
Ví dụ 2.61

Vẽ đồ thị các điểm dữ liệu với dữ liệu ngẫu nhiên được tạo như sau:

$X = 5 * \text{rand}(100,1);$

$Y = \text{rand} * X + \text{rand}(100,1);$

Giải: Chương trình vẽ đồ thị với hàm plot() như sau với kết quả ở Hình 2.4:



Hình 2.4: Minh họa vẽ đồ thị 2D với dữ liệu ngẫu nhiên.

```
X = 5 * rand(100,1);
Y = rand * X + rand(100,1);
figure(1);
plot(X,Y,'.');
xlabel('X = 5 * rand(100,1)');
ylabel('Y = rand * X + rand(100,1)');
```

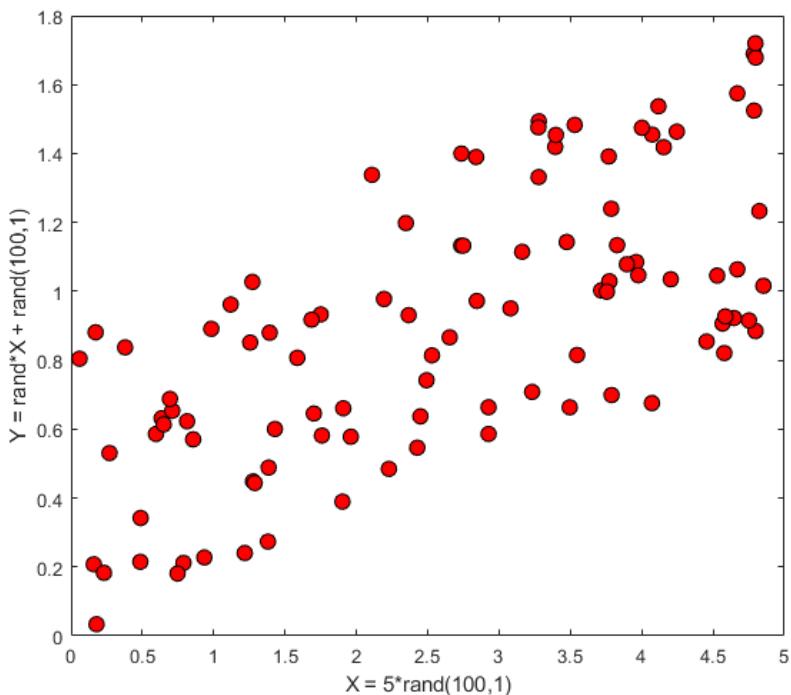
Matlab cho phép chúng ta điều chỉnh cách thức thể hiện đồ thị, về loại đường, độ dày và nhiều tính chất khác. Chúng ta có thể tham khảo thông qua lệnh **doc linespec**.

Ví dụ 2.62

Hãy điều chỉnh kích thước, màu sắc và độ dày của mỗi điểm biểu diễn trên đồ thị của Ví dụ 2.61 thành hình tròn màu đỏ viền đen lớn.

Giải: Lệnh điều chỉnh như sau:

```
plot(X,y,'o','MarkerEdgeColor','k',...
      'MarkerFaceColor','r',...
      'MarkerSize',8);
```



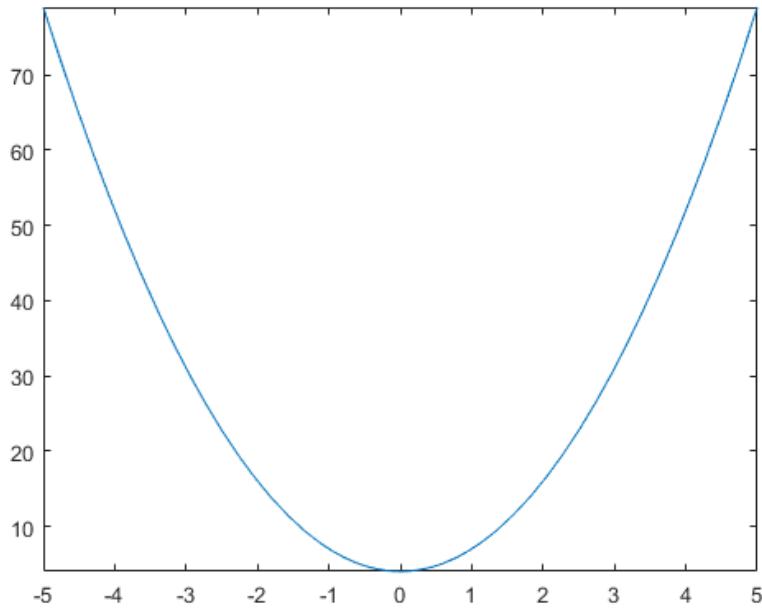
Hình 2.5: Hình vẽ của Ví dụ 2.55.

Ví dụ 2.63

Sử dụng hàm fplot() vẽ đồ thị của hàm $3x^2 + 4$.

Giải: Ta sẽ vẽ đồ thị với trục x trong khoảng từ (-5,5).

```
fun=@(x) 3*x.^2 + 4;
fplot(fun,[-5 5]);
set(gcf,'color','white');
```



Hình 2.6: Đồ thị của hàm khi vẽ với hàm fplot()

2.2.1.2 Đặt tiêu đề và đặt tên cho các trục đồ thị

Chúng ta có thể đặt tên cho đồ thị, trục đồ thị và đồng thời điều chỉnh vùng hiển thị của đồ thị với các lệnh axis(), xlabel(), ylabel() và title() với cú pháp như sau.

Cú pháp:

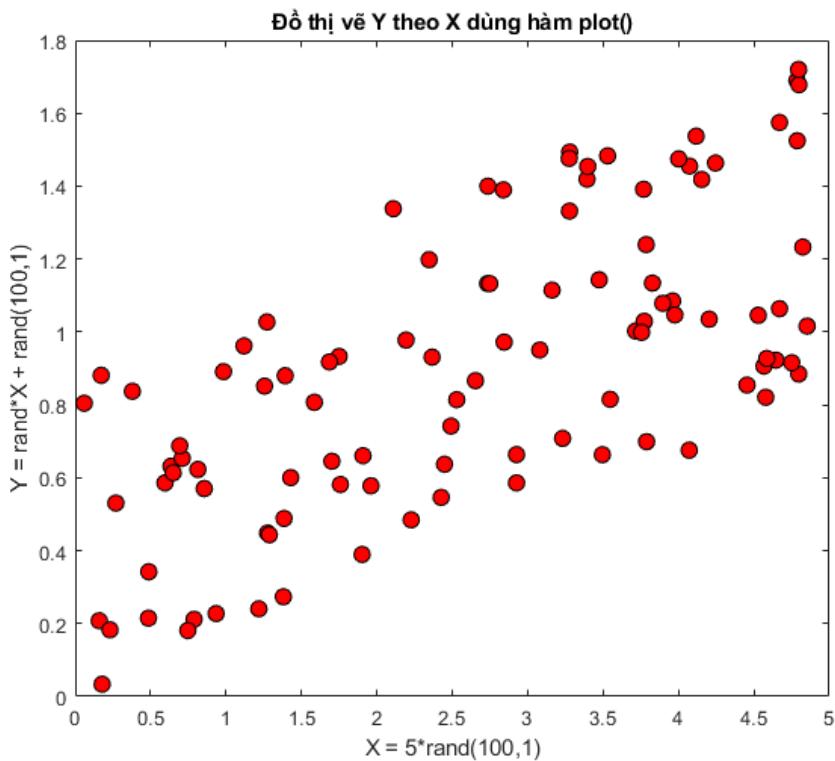
```
axis([-1,6,-1,5]);
xlabel('x');
ylabel('y');
title('Tên của đồ thị');
```

Ví dụ 2.64

Đặt tên cho đồ thị của Ví dụ 2.63 với tên là “Đồ thị vẽ Y theo X dùng hàm plot()”.

Giải: Chúng ta thực hiện lệnh như sau:

```
title('Đồ thị vẽ Y theo X dùng hàm plot()');
```



Hình 2.7: Minh họa đặt tên cho đồ thị.

Bên cạnh hàm `plot()` để vẽ hàm 2D với hai trục trên đồ thị đều tuyến tính, Matlab còn cung cấp cho chúng ta các hàm `semilogx()`, `semilogy()` và `loglog()` để vẽ đồ thị dạng log cơ số 10 cho trục x, trục y và cho cả hai trục x và y. Khi trình diễn kết quả hiệu năng của các hệ thống truyền thông, đặc biệt là ở lớp vật lý như tỷ lệ lỗi bit, tỷ lệ lỗi symbol và xác suất dừng hệ thống, chúng ta hay sử dụng hàm `semilogy()`. Tuy nhiên, khi chúng ta vẽ dung lượng kênh truyền thì không dùng đồ thị theo dạng log mà theo dạng tuyến tính cả hai trục.

Ví dụ 2.65

Biết rằng xác suất dừng của hệ thống SISO trên kênh truyền fading Rayleigh là $OP = 1 - \exp\left(-\frac{\gamma_{th}}{\bar{\gamma}}\right)$ với $\gamma_{th} = 1$ là ngưỡng dừng và $\bar{\gamma}$ là tỷ lệ tín hiệu trên nhiễu trung bình.

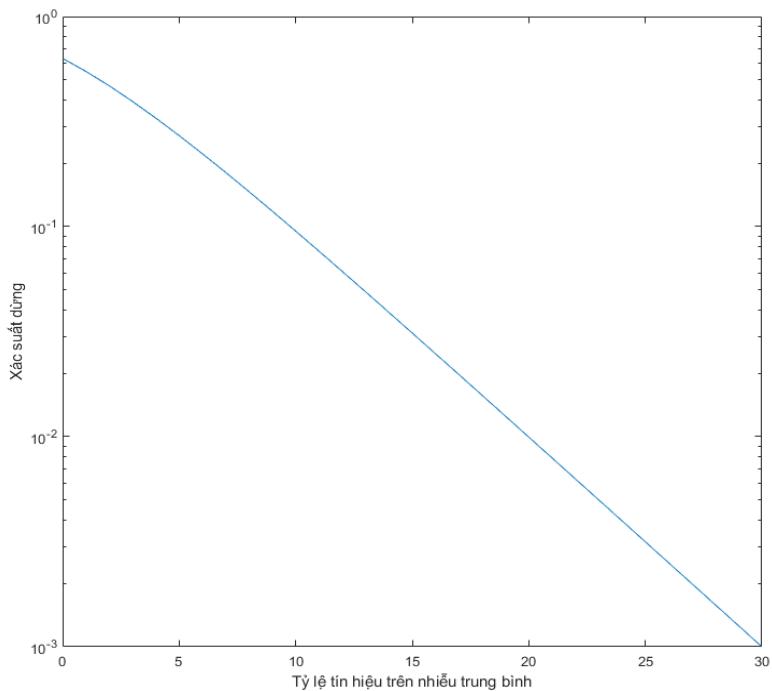
Hãy vẽ đồ thị của OP theo tỷ lệ tín hiệu trên nhiễu trung bình từ 0 đến 30 dB.

Giải: Chúng ta thực hiện đoạn mã sau ở cửa sổ lệnh.

```

SNRdB = 0:30;
SNR = 10.^ (SNRdB/10);
gth = 1;
OP = 1 - exp(-gth./SNR);
semilogy(SNRdB,OP);
xlabel('Tỷ lệ tín hiệu trên nhiễu trung bình');
ylabel('Xác suất dừng hệ thống');

```



Hình 2.8: Vẽ đồ thị xác suất đúng hệ thống sử dụng hàm semilogy().

2.2.1.3 Vẽ trên cùng một đồ thị và ghi chú đồ thị

Để thực hiện so sánh kết quả mô phỏng hay kết quả lý thuyết, chúng ta thông thường biểu diễn kết quả trên cùng một trực đồ thị và thực hiện ghi chú đồ thị để phân biệt. Để vẽ trên cùng một đồ thị, chúng ta có hai cách:

- Cách 1: Đưa dữ liệu của hai hay nhiều đường đồ thị là tham số đầu vào của hàm plot.
- Cách 2: Vẽ tuần tự từng đường đồ thị và sử dụng hàm hold on.

Việc thực hiện ghi chú đồ thị thực hiện thông qua hàm **legend()**. Phần ghi chú cho đồ thị là phần rất quan trọng, chúng ta có thể thực hiện điều chỉnh nội dung phần ghi chú cùng với vị trí của phần ghi chú thông qua các tham số đầu vào của hàm **legend()**. Muốn điều chỉnh vị trí của phần ghi chú trên đồ thị, chúng ta thiết lập tham số “Location” theo các giá trị như Bảng 2.5.

Bảng 2.5: Bảng tham số vị trí cho hàm legend().

TT	Giá trị thiết lập	Viết tắt	Vị trí trên đồ thị
1	'north'	'n'	Trong đồ thị, hướng 12h
2	'south'	's'	Trong đồ thị, hướng 6h
3	'east'	'e'	Trong đồ thị, hướng 3h
4	'west'	'w'	Trong đồ thị, hướng 9h
5	'northeast'	'ne'	Trong đồ thị, hướng 13h
6	'northwest'	'nw'	Trong đồ thị, hướng 11h
7	'southeast'	'se'	Trong đồ thị, hướng 5h

8	'southwest'	'sw'	Trong đồ thị, hướng 7h
9	'northoutside'	'no'	Ngoài đồ thị, hướng 12h
10	'southoutside'	'so'	Ngoài đồ thị, hướng 6h
11	'eastoutside'	'eo'	Ngoài đồ thị, hướng 3h
12	'westoutside'	'wo'	Ngoài đồ thị, hướng 9h
13	'northeastoutside'	'ne'	Ngoài đồ thị, hướng 13h
14	'northwestoutside'	'nw'	Ngoài đồ thị, hướng 11h
15	'southeastoutside'	'se'	Ngoài đồ thị, hướng 5h
16	'southwestoutside'	'sw'	Ngoài đồ thị, hướng 7h
17	'best'	'b'	Trong đồ thị, Matlab tự chọn vị trí tốt nhất, không che các đường vẽ
18	'bestoutside'	'bo'	Ngoài đồ thị, Matlab tự chọn vị trí tốt nhất

Hàm **grid on/off** cho phép chúng ta bật/tắt lưới trên đồ thị. Trong một số trường hợp, dùng lưới sẽ rất tốt, cho phép chúng ta có thể ước lượng giá trị trên đồ thị, nhưng trong một số trường hợp khác, ví dụ như số lượng đường biểu diễn trên đồ thị lớn, thì bật lưới sẽ gây rối đồ thị.

Để biểu diễn các thanh lỗi của đồ thị, chúng ta dùng hàm **errorbar()**.

Ví dụ 2.66

Vẽ hàm sin và cos trong hai chu kỳ và vẽ thanh lỗi cho hàm sin.

Giải: Chúng ta thực hiện trên trình biên soạn của Matlab như sau:

```

N = 50;
x = 0:2*pi/N:4*pi;
f = @sin;
g = @cos;

y1 = f(x);
y2 = g(x);

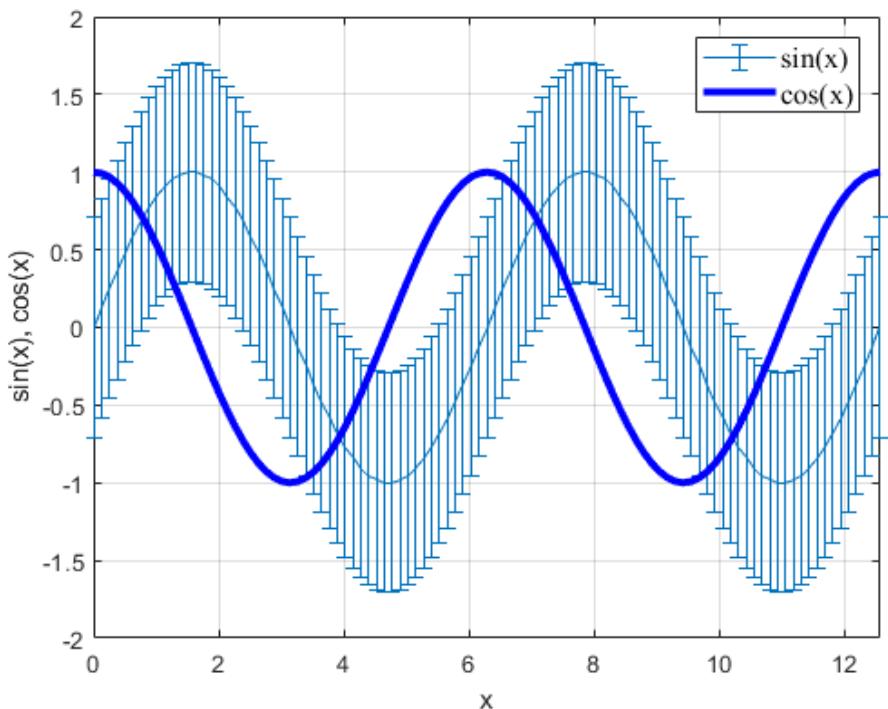
e1 = std(y1)*ones(size(x));
e2 = std(y2)*ones(size(x));

plot(x,y1, '--r','LineWidth',3);
errorbar(x,y1,e1);
xlabel('x');
ylabel('sin(x), cos(x)');
grid on

hold on
plot(x,y2, '-b','LineWidth',3);
axis([0,4*pi,-2,2]);

legend('sin(x)','cos(x)', 'FontSize',12);

```



Hình 2.9: Kết quả của Ví dụ 2.58.

2.2.1.4 Chia nhỏ đồ thị và đồ thị con

Trong một số tình huống chúng ta muốn biểu diễn nhiều hình và mỗi hình trên một đồ thị con. Matlab cho phép chúng ta thực hiện điều này với **subplot()**, cú pháp như sau:

Cú pháp:

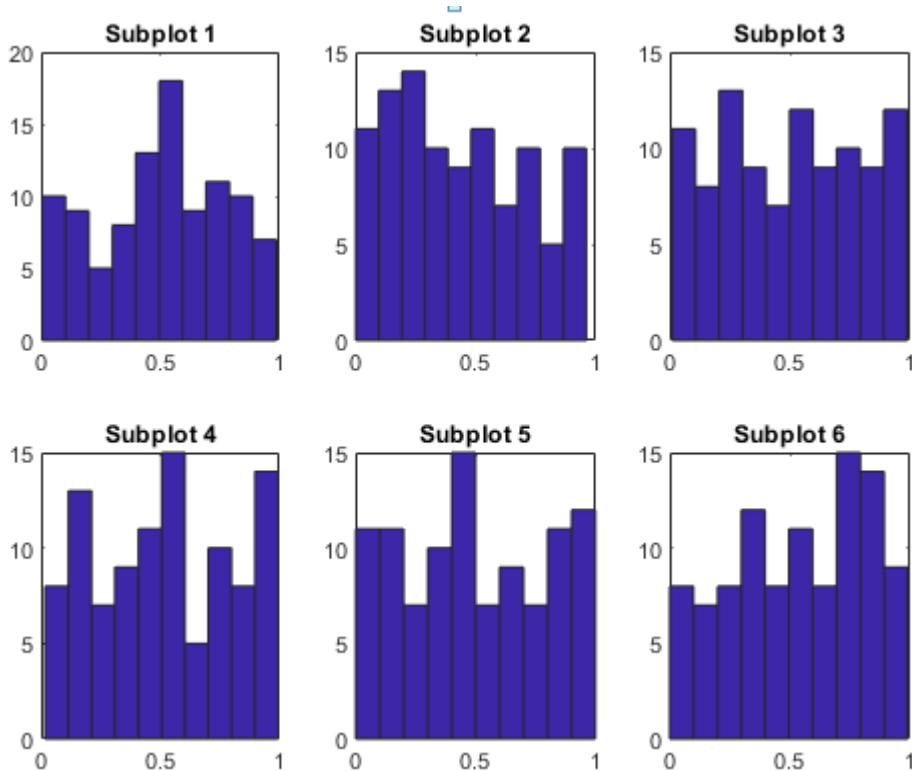
```
subplot(nr,nc,i)
% nr: tổng số lượng cột
% nc: tổng số lượng hàng
% i: chỉ số của đồ thị con theo quy định từ trái sang phải, từ trên xuống dưới.
```

Ví dụ 2.67

Vẽ 6 đồ thị trong 2 dòng, mỗi đồ thị là biểu đồ tần suất của các biến ngẫu nhiên $\text{rand}(1,100)$.

Giải: 6 đồ thị trong hai dòng nghĩa là $\text{nr} = 2$ và $\text{nc} = 3$.

```
for i=1:6
    subplot(2,3,i);
    hist(rand(1,100),10);
    title(sprintf('Subplot %d', i));
end
```



Hình 2.10: Kết quả của Ví dụ 2.59.

2.2.1.5 Cấu hình đồ thị

Trong một số trường hợp, chúng ta muốn điều chỉnh các tính chất của đồ thị. Có hai cách để thực hiện, một là thực hiện bằng cách sử dụng các hàm, ví dụ như hàm `get()`, `set()`, `handle()`, hay thực hiện bằng cách tương tác trực tiếp trên đồ thị bằng cách mở “Open Property Inspector” như Hình 2.11.



Hình 2.11: Nút cuối cùng bên phải là Open Property Inspector.

2.2.1.6 Biểu đồ tần suất

Histograms là biểu đồ tần suất trong tiếng Việt, là một công cụ hữu ích để vẽ hàm mật độ phân bố xác suất của biến ngẫu nhiên. Có hai cách để vẽ biểu đồ tần suất: tự lập trình theo định nghĩa hay sử dụng hàm có sẵn của Matlab.

Nếu theo định nghĩa, giả sử chúng ta có biến ngẫu nhiên R , có chiều dài là N . Việc vẽ biểu đồ tần suất theo các bước như sau:

- Bước 1: tìm giá trị nhỏ nhất và lớn nhất của R
- Bước 2: xác định phân vùng trục R của biểu đồ tần suất.
- Bước 3: tính số lượng giá trị của R trong từng phân vùng.
- Bước 4: vẽ biểu đồ tần suất bằng cách lấy số lượng của giá trị trong từng phân vùng chia cho chiều dài của R .
- Bước 5 (nếu có): vẽ biểu đồ hàm mật độ phân bố xác suất của R kiểm chứng kết quả ở Bước 4.

Ví dụ 2.68

Thực hiện vẽ biểu đồ tần suất của biến ngẫu nhiên phân bố Gauss có trung bình bằng 3 và phương sai bằng 1.

Giải: Chúng ta sẽ sử dụng hàm randn() để tạo biến ngẫu nhiên phân bố chuẩn, sau đó sẽ nhân với độ lệch chuẩn và cộng với giá trị trung bình để tạo ra phân bố Gauss như mong muốn. Sau đó, thực hiện giải thuật 5 bước để vẽ biểu đồ tần suất như sau.

```
% Khởi tạo biến ngẫu nhiên R
N = 10^6;
muR = 3;
varR = 1;
R = muR + sqrt(varR).*randn(1,N);

% Bước 1:
xMin = min(R);
xMax = max(R);

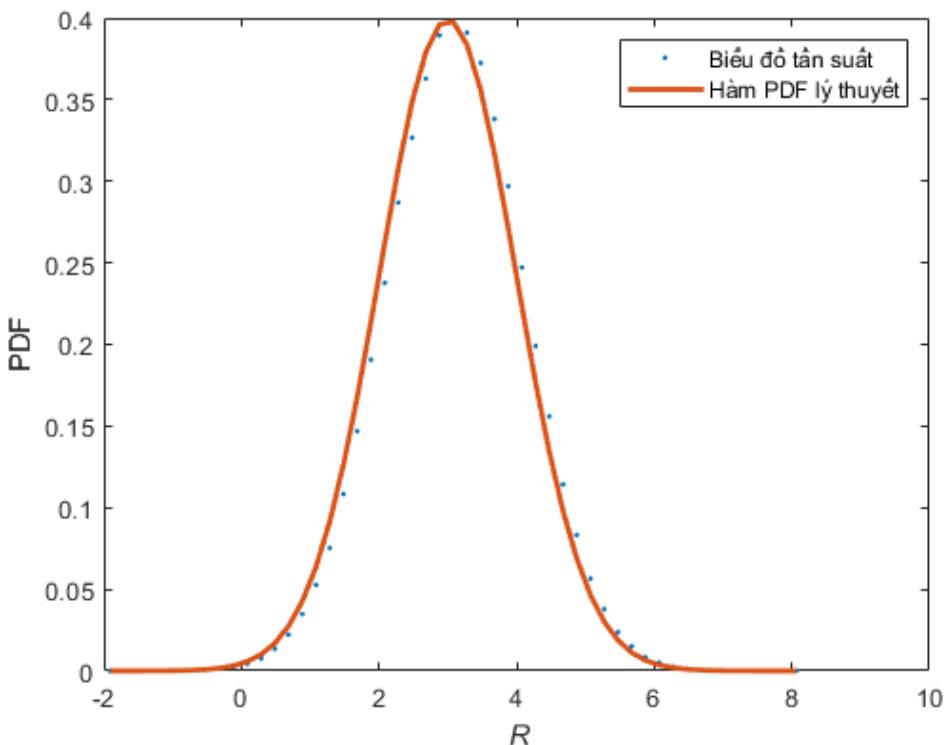
% Bước 2
NoB = 50;
dx = (xMax - xMin)/NoB;

% Bước 3
x = xMin:dx:xMax;

% Bước 4
PDF_s = zeros(size(x));
for idx = 2:length(x)
    PDF_s(idx) = PDF_s(idx) + sum((R >= x(idx-1))&(R < x(idx)))/(dx*N);
end

% Bước 5
PDF_t = 1./sqrt(2*pi*varR).*exp(-(x - muR).^2./(2*varR));

% Vẽ đồ thị
plot(x,PDF_s,'.',x,PDF_t,'-','linewidth',2);
xlabel('t R');
ylabel('PDF');
legend('Biểu đồ tần suất','Hàm PDF lý thuyết');
set(gcf,'color','white');
```



Hình 2.12: Biểu đồ tần suất và hàm PDF theo giải thuật 5 bước.

Cách thứ 2 là sử dụng hàm có sẵn của Matlab, đó là hàm histogram().

Cú pháp:

histogram(X,M)

với X là vector cần vẽ biểu đồ tần suất và M nếu là một số là số lượng mốc vẽ trên đồ thị.

Ví dụ 2.69

Cho điểm của 10 học sinh như sau 4, 5, 7, 9, 2 4, 9, 10, 10, 2. Hãy vẽ đồ thị tần suất theo điểm từ 0 đến 10.

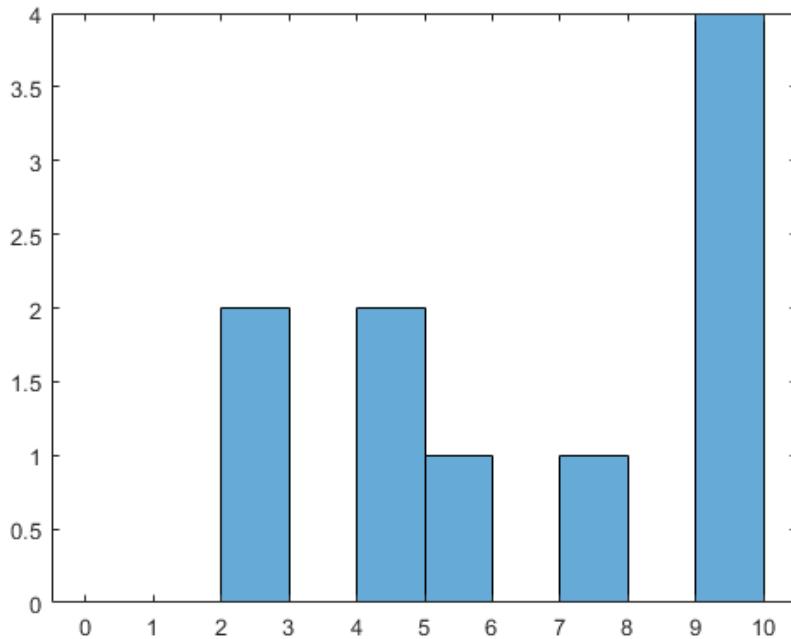
Giải: Chúng ta dùng hàm histogram() để vẽ đồ thị tần suất.

```
> A=[4, 5, 7, 9, 2 4, 9, 10, 10, 2]
```

```
A =
```

```
4   5   7   9   2   4   9   10  10   2
```

```
>> M=0:10;
>> histogram(A,M)
```



Hình 2.13: Đồ thị tần suất của Ví dụ 2.69.

2.2.1.7 Vẽ hàm mật độ phân bố xác suất và hàm phân bố xác suất tích lũy

Matlab hỗ trợ hai hàm pdf() và cdf() để tính giá trị hàm mật độ phân bố xác suất và hàm phân bố xác suất tích lũy. Cú pháp của hai hàm này như sau:

Cú pháp:

`Y = pdf(NAME,X,A)`

% NAME là tên của phân phối, cho ở Bảng 2.6.

% X là điểm cần tính giá trị pdf

% A là tham số của phân phối. Nếu phân phối có nhiều hơn một tham số, các tham số có thể tiếp nối sau A.

Cú pháp:

`Y = cdf(NAME,X,A)`

% NAME là tên của phân phối, cho ở Bảng 2.6.

% X là giá trị cần tính giá trị cdf

% A là tham số của phân phối. Nếu phân phối có nhiều hơn một tham số, các tham số có thể tiếp nối sau A.

Malab không hỗ trợ tất cả các phân phối xác suất mà chỉ hỗ trợ các phân phối xác suất cho trong Bảng 2.6.

Bảng 2.6: Danh mục các phân bố hỗ trợ bởi hàm $cdf()$ và hàm $pdf()$.

TT	Tham số tên phân bô chuẩn hóa NAME	Tên phân bô	Tham số đầu vào A	Tham số đầu vào B
1	'Beta'	Phân phối Beta	a first shape parameter	b second shape parameter
2	'Binomial'	Phân phối Binomial	n number of trials	p probability of success for each trial
3	'BirnbaumSaunders'	Phân phối Birnbaum-Saunders	β scale parameter	γ shape parameter
4	'Burr'	Phân phối Burr Type XII	α scale parameter	c first shape parameter
5	'Chisquare'	Phân phối Chi-Square	v degrees of freedom	—
6	'Exponential'	Phân phối Exponential	μ mean	—
7	'Extreme Value'	Phân phối Extreme Value	μ location parameter	σ scale parameter
8	'F'	Phân phối F	v1 numerator degrees of freedom	v2 denominator degrees of freedom
9	'Gamma'	Phân phối Gamma	a shape parameter	b scale parameter
10	'Generalized Extreme Value'	Phân phối Generalized Extreme Value	k shape parameter	σ scale parameter
11	'Generalized Pareto'	Phân phối Generalized Pareto	k tail index (shape) parameter	σ scale parameter
12	'Geometric'	Phân phối Geometric	p probability parameter	—
13	'HalfNormal'	Phân phối Half-Normal	μ location parameter	σ scale parameter
14	'Hypergeometric'	Phân phối Hypergeometric	m size of the population	k number of items with the desired characteristic in the population
15	'InverseGaussian'	Phân phối Inverse Gaussian	μ scale parameter	λ shape parameter
16	'Logistic'	Phân phối Logistic	μ mean	σ scale parameter
17	'LogLogistic'	Phân phối Loglogistic	μ mean of logarithmic values	σ scale parameter of logarithmic values

18	'Lognormal'	Phân phối Lognormal	μ mean of logarithmic values	σ standard deviation of logarithmic values
19	'Nakagami'	Phân phối Nakagami	μ shape parameter	ω scale parameter
20	'Negative Binomial'	Phân phối Negative Binomial	r number of successes	p probability of success in a single trial
21	'Noncentral F'	Phân phối Noncentral F	v1 numerator degrees of freedom	v2 denominator degrees of freedom
22	'Noncentral t'	Phân phối Noncentral t	v degrees of freedom	δ noncentrality parameter
23	'Noncentral Chi-square'	Phân phối Noncentral Chi-Square	v degrees of freedom	δ noncentrality parameter
24	'Normal'	Phân phối Normal	μ mean	σ standard deviation
25	'Poisson'	Phân phối Poisson	λ mean	—
26	'Rayleigh'	Phân phối Rayleigh	b scale parameter	—
27	'Rician'	Phân phối Rician	s noncentrality parameter	σ scale parameter
28	'Stable'	Phân phối Stable	α first shape parameter	β second shape parameter
29	'T'	Phân phối Student's t	v degrees of freedom	—
30	'tLocationScale'	Phân phối t Location-Scale	μ location parameter	σ scale parameter
31	'Uniform'	Phân phối Uniform (Continuous)	a lower endpoint (minimum)	b upper endpoint (maximum)
32	'Discrete Uniform'	Phân phối Uniform (Discrete)	n maximum observable value	—
33	'Weibull'	Phân phối Weibull	a scale parameter	b shape parameter

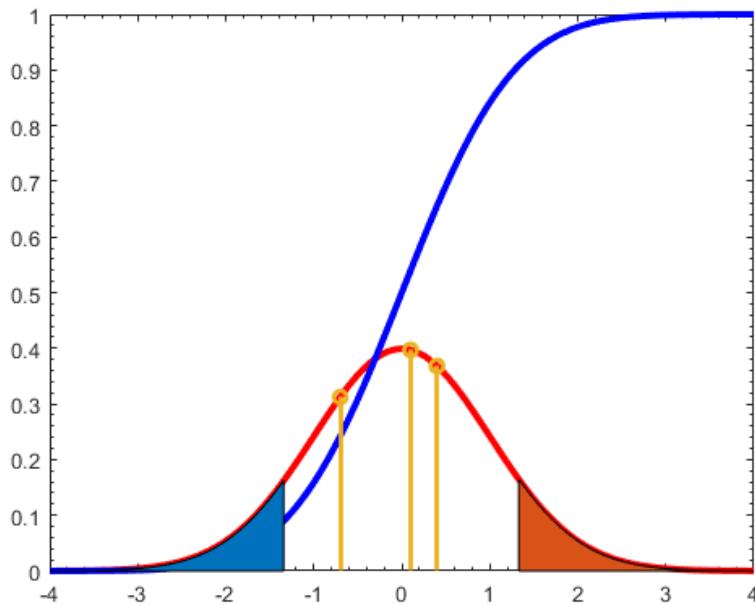
Các ví dụ sau sẽ trình bày cách thức vẽ hàm PDF và CDF của một biến ngẫu nhiên. Giả sử chúng ta xem xét với biến phân bố Gauss ở Ví dụ 2.70.

Ví dụ 2.70

- Vẽ hàm PDF và CDF của phân bố chuẩn.
- Vẽ các vùng diện tích bên dưới hàm PDF.
- Vẽ đường chia cắt các vùng diện tích.

Giải: Để tô các vùng dưới đồ thị, chúng ta dùng hàm area(). Để vẽ các đường thẳng phân cách giữa các vùng, chúng ta dùng hàm stem().

```
figure;
domain = -4:0.01:4;
plot(domain,normpdf(domain),'r','LineWidth',3);
hold on
plot(domain,normcdf(domain),'b','LineWidth',3);
%
hold on
% Vẽ các vùng
L = fix(length(domain))/3;
R = 2*fix(length(domain))/3;
area(domain(1:L),normpdf(domain(1:L)));
area(domain(R:end),normpdf(domain(R:end)));
% Vẽ các đường
stem([-0.7,0.1,0.4],normpdf([-0.7,0.1,0.4]),'LineWidth',2);
% Điều chỉnh đồ thị
axis([-4,4,0,0.5]);
set(gca,'XMinorTick','on');
set(gca,'YMinorTick','on');
```



Hình 2.14: Hàm PDF và CDF của phân bố chuẩn.

Ví dụ 2.71

Hãy điều chỉnh chương trình Matlab của Ví dụ 2.70 và vẽ hàm CDF và PDF của phân bố hàm mũ có tham số bằng 2.

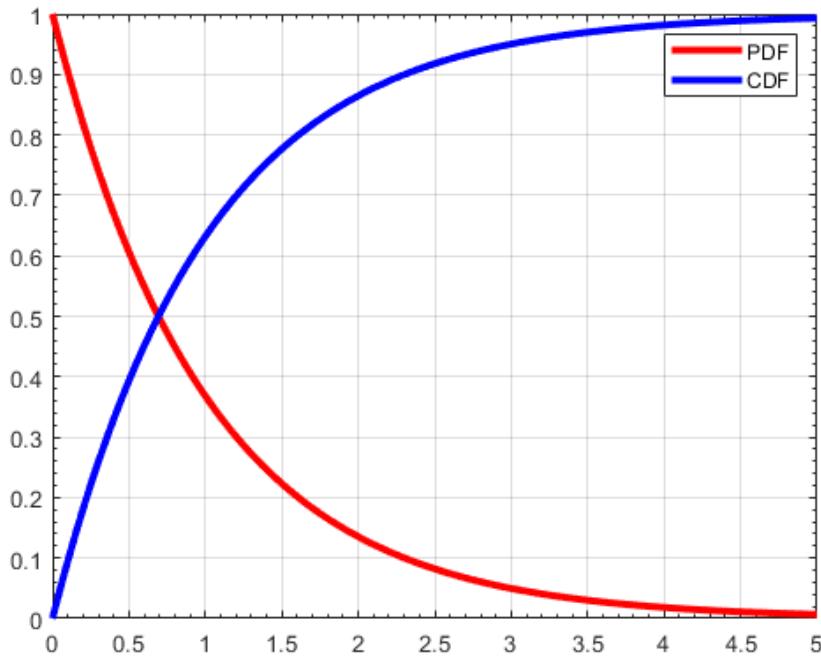
Giải: Để vẽ hàm CDF và PDF của phân bố mũ, chúng ta sử dụng hàm expcdf() và exppdf().

```
figure;
domain = 0:0.01:5;
plot(domain,exppdf(domain),'r','LineWidth',3);
```

```

hold on
plot(domain,expcdf(domain),'-b','LineWidth',3);
% Điều chỉnh đồ thị
set(gca,'XMinorTick','on');
set(gca,'YMinorTick','on');
grid on
legend('PDF','CDF');
set(gcf,'color','white');

```



Hình 2.15: Hàm PDF và CDF của phân bố hàm mũ.

2.2.1.8 Đọc một file ảnh và trình chiếu

Matlab cung cấp hàm **imagesc()** để trình chiếu một ma trận ảnh. Nếu có Toolbox image processing, ta có thể dùng hàm **imshow()**. Để đọc và chuyển một ảnh thành ma trận điểm ảnh, ta dùng hàm **imread()**.

Ví dụ 2.72

Biết rằng mandrill một file lưu ma trận ảnh X có sẵn trong Matlab. Hãy trình chiếu ma trận X chưa trong mandrill theo các mức xám.

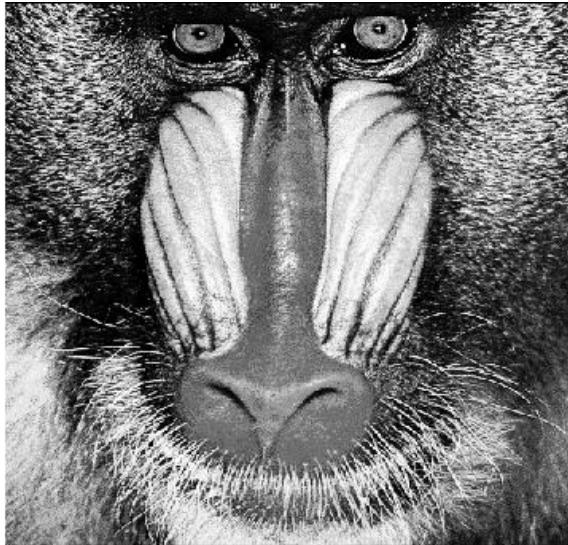
Giải: Chúng ta lần lượt dùng các hàm **load()**, **figure()**, **imagesc()**, **axis()** và hàm **colormap()**.

```

% Đọc ma trận ảnh X từ file mandrill vào workspace
load mandrill
% Khởi tạo đồ thị và trình chiếu ảnh từ ma trận X
figure; imagesc(X);
% Điều chỉnh đồ thị cho phù hợp với ảnh
axis image
% Điều chỉnh mức xám

```

```
map = colormap(gray);
% Tắt đồ thị
axis off
```



Hình 2.16: Hình m/c xám dùng hàm imagesc.

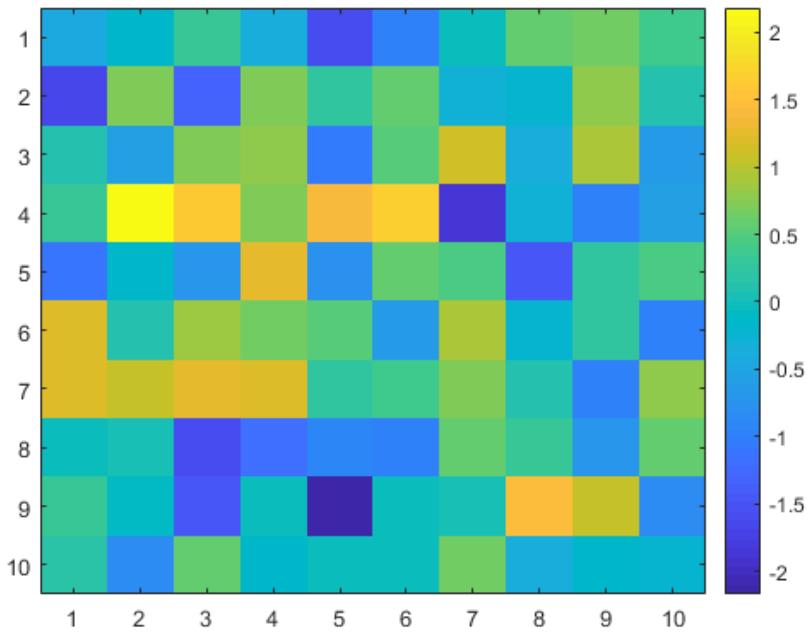
Trong một số trường hợp, chúng ta có thể dùng hàm imagesc() để vẽ biểu đồ nhiệt của ma trận 2 chiều. Biểu đồ nhiệt dùng để biểu diễn đồ họa của dữ liệu trong đó các giá trị riêng lẻ có trong ma trận được biểu diễn dưới dạng màu. Một ứng dụng tiêu biểu của đồ thị nhiệt trong viễn thông là để vẽ vùng phủ sóng của một trạm phát.

Ví dụ 2.73

Vẽ biểu đồ nhiệt của ma trận ngẫu nhiên theo phân bố chuẩn 10×10 .

Giải: Ta dùng hàm randn() để tạo ma trận ngẫu nhiên theo phân bố chuẩn.

```
figure;
rand('twister',1);
X = randn(10,10);
imagesc(X);
colorbar
```



Hình 2.17: Đồ thị nhiệt của $\text{randn}(10,10)$.

2.2.1.9 Hàm lưu trữ và in ấn

Để lưu trữ kết quả mô phỏng, đặc biệt là các mô phỏng tồn tại nhiều thời gian để thực thi, chúng ta có thể dùng hàm **print()** và hàm **save()** trong các chương trình mô phỏng Matlab.

Cú pháp:

```
print(filename, formattype)
% filename là tên file muốn lưu
% formattype: là định dạng muốn lưu file, ví dụ PDF ('-dpdf'), encapsulated
PostScript ('-depsc'), JPEG ('-djpeg'), PNG ('-dpng')
```

Cú pháp:

```
save(FILENAME):
% lưu toàn bộ biến ở Workspace thành file có tên là FILENAME theo định dạng
.mat của Matlab.
save(FILENAME,VARIABLES)
% lưu các biến VARIABLES vào file có tên FILENAME
```

2.2.1.10 LaTeX trong Matlab

Matlab có hỗ trợ LaTeX bao gồm các lệnh bô nghĩa ở Bảng 2.7 và các ký tự đặc biệt ở Bảng 2.8.

Bảng 2.7: Bảng các lệnh bỏ nghĩa của LaTeX trong Matlab.

Lệnh bỏ nghĩa	Chú thích	Ví dụ
<code>^{ }</code>	Chỉ số trên	'text^{superscript}'
<code>_ { }</code>	Chỉ số dưới	'text_{subscript}'
<code>\bf</code>	In đậm	'\bf text'
<code>\it</code>	In nghiêng	'\it text'
<code>\sl</code>	In nghiêng, tương tự <code>\it</code>	'\sl text'
<code>\rm</code>	Font bình thường	'\rm text'
<code>\fontname{ }</code>	Điều chỉnh loại font	'\fontname{Courier} text'
<code>\fontsize{ }</code>	Điều chỉnh kích thước font với đơn vị là point units.	'\fontsize{15} text'
<code>\color{ }</code>	Điều chỉnh màu của font với các màu sau: red, green, yellow, magenta, blue, black, white, gray, darkGreen, orange, hay lightBlue.	'\color{magenta} text'
<code>\color[rgb]{specifier}</code>	Điều chỉnh màu theo hệ số màu RGB	'\color[rgb]{0,0.5,0.5} text'

Bảng 2.8: Các ký tự đặc biệt của LaTeX trong Matlab.

Lệnh LaTeX	Ký tự	Lệnh LaTeX	Ký tự	Lệnh LaTeX	Ký tự
<code>\alpha</code>	A	<code>\upsilon</code>	Y	<code>\sim</code>	~
<code>\angle</code>	∠	<code>\phi</code>	φ	<code>\leq</code>	≤
<code>\ast</code>	*	<code>\chi</code>	χ	<code>\infty</code>	∞
<code>\beta</code>	β	<code>\psi</code>	ψ	<code>\clubsuit</code>	♣
<code>\gamma</code>	γ	<code>\omega</code>	ω	<code>\diamondsuit</code>	♦
<code>\delta</code>	δ	<code>\Gamma</code>	Γ	<code>\heartsuit</code>	♥
<code>\epsilon</code>	ε	<code>\Delta</code>	Δ	<code>\spadesuit</code>	♠
<code>\zeta</code>	ζ	<code>\Theta</code>	Θ	<code>\leftrightarrow</code>	↔
<code>\eta</code>	η	<code>\Lambda</code>	Λ	<code>\leftarrow</code>	←
<code>\theta</code>	θ	<code>\Xi</code>	Ξ	<code>\Leftarrow</code>	⇐
<code>\vartheta</code>	ϑ	<code>\Pi</code>	Π	<code>\uparrow</code>	↑
<code>\iota</code>	ι	<code>\Sigma</code>	Σ	<code>\rightarrow</code>	→
<code>\kappa</code>	κ	<code>\Upsilon</code>	Υ	<code>\Rightarrow</code>	⇒
<code>\lambda</code>	λ	<code>\Phi</code>	Φ	<code>\downarrow</code>	↓
<code>\mu</code>	μ	<code>\Psi</code>	Ψ	<code>\circ</code>	◦
<code>\nu</code>	ν	<code>\Omega</code>	Ω	<code>\pm</code>	±
<code>\xi</code>	ξ	<code>\forall</code>	∀	<code>\geq</code>	≥
<code>\pi</code>	π	<code>\exists</code>	∃	<code>\propto</code>	∞
<code>\rho</code>	ρ	<code>\ni</code>	∋	<code>\partial</code>	∂
<code>\sigma</code>	σ	<code>\cong</code>	≈	<code>\bullet</code>	•

\varsigma	ς	\approx	\div	\div
\tau	τ	\Re	\neq	\neq
\equiv	\equiv	\oplus	\oplus	\aleph
\Im	\Im	\cup	\cup	\wp
\otimes	\otimes	\subsetneq	\subseteq	\oslash
\cap	\cap	\in	\Subset	\supseteq
\supset	\supset	\lceil	\subset	\subset
\int	\int	\cdot	\circ	\circ
\rfloor	\rfloor	\neg	\neg	\nabla
\lfloor	\lfloor	\times	x	\ldots
\perp	\perp	\surd	\sqrt	\prime
\wedge	\wedge	\varpi	\omega	\emptyset
\rceil	\rceil	\rangle	\mid	
\vee	\vee	\langle	\copyright	\circledcirc

Để sử dụng chế độ LaTeX, sử dụng ký hiệu \$\$ xung quanh mã LaTeX. Lưu ý rằng khi sử dụng chế độ LaTeX, Matlab sử dụng font chữ mặc định và các thuộc tính như FontName, FontWeight và FontAngle trở nên không có hiệu lực.

Ví dụ 2.74

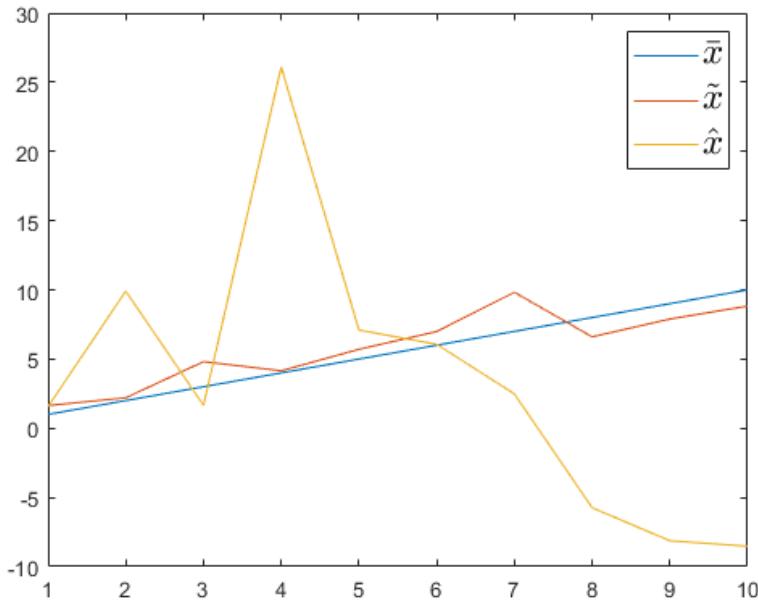
Thực hiện dòng lệnh sau và quan sát kết quả mà Matlab trả về:

```

x = 1:10;
n = randn(size(x));
x1 = x + n;
x2 = x./n;
plot(x,x,x1,x,x2);
leg1 = legend('$\bar{x}$','$\tilde{x}$','$\hat{x}$');
set(leg1,'Interpreter','latex');
set(leg1,'FontSize',17);
set(gcf,'color','w');

```

Giải: Sau khi thực hiện các dòng lệnh trên, chúng ta có Hình 2.18.



Hình 2.18: Hình từ Ví dụ 2.74.

Ví dụ 2.75

Sử dụng hàm `latex()` để trả về mã LaTeX cho biểu thức của các biến symbolic sau

a. $x^2 + 1/x$

b. $\sin(\pi*x) + \phi$.

Giải: Điểm lưu ý là chúng ta cần khai báo biến symbolic cho x và ϕ trước khi thực hiện lệnh `latex()`.

```

syms x phi
chr = latex(x^2 + 1/x)
chr = latex(sin(pi*x) + phi)
chr =
    '\frac{2}{x}+x^2'

chr =
    '\phi+\sin\left(\pi\,x\right)'

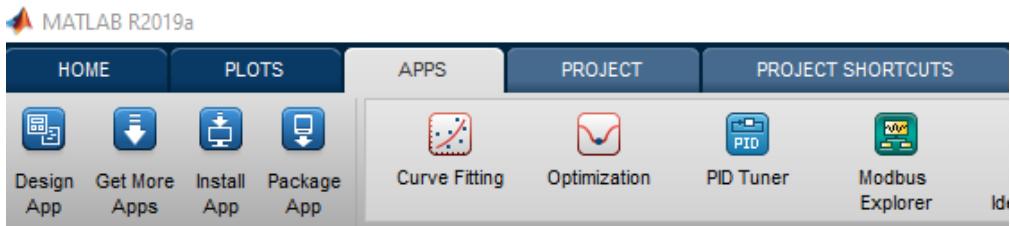
```

2.3 Lập trình giao diện và ứng dụng trong Matlab

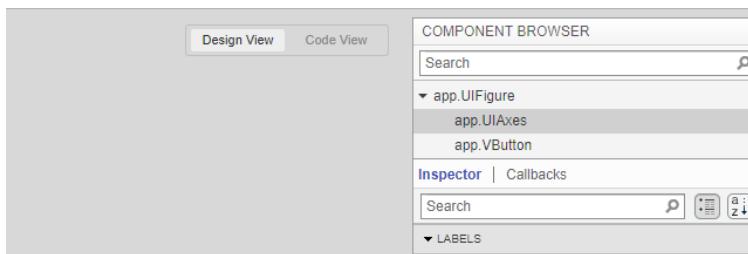
Matlab hỗ trợ hai cách để lập trình giao diện:

- App Designer: Đây là cách lập trình giao diện mới, hỗ trợ nhiều công cụ và chức năng hơn và đặc biệt là có hỗ trợ biên dịch với Matlab compiler. Để khởi động App Designer, chúng ta chọn TAB “Design App” như Hình 2.19.

- GUIDE và Programmatic Workflow: là phiên bản cũ của App Designer có thể phù hợp với mọi phiên bản Matlab. Để khởi tạo GUIDE, chúng ta gõ “guide” ở cửa sổ lệnh.



Hình 2.19: Giao diện mở phần lập trình giao diện “App Designer”.



Hình 2.20: Tab chọn “Design View” hay “Code View”.

Để đơn giản, phần này chỉ trình bày cách lập trình trên App Designer. Sau khi vào giao diện chính của “App Designer”, chúng ta sẽ có hai TAB để làm việc như Hình 2.20, đó là:

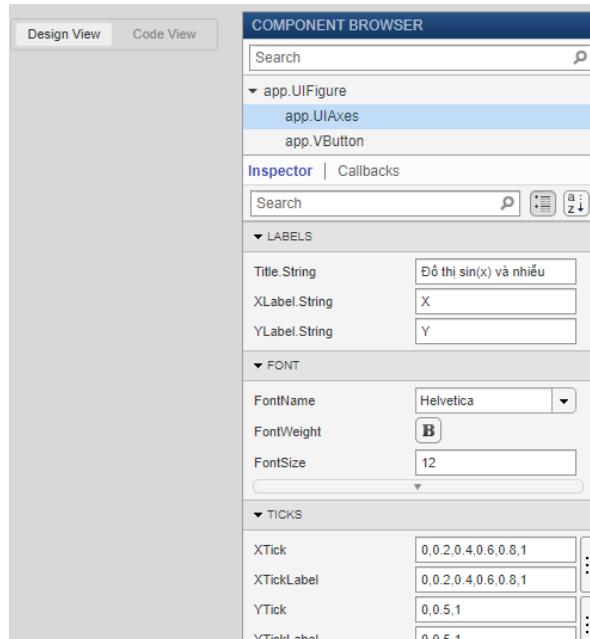
- “Design View”: giao diện để chúng ta thiết kế giao diện cho ứng dụng dùng phương pháp kéo và thả, toàn bộ thư viện các thành phần (components) ở cột bên trái.
- “Code View”: giao diện để lập trình các chức năng cho ứng dụng. Phần lập trình được sẽ có màu sáng và điều chỉnh được, phần không lập trình và điều chỉnh được sẽ có màu tối.

Bảng 2.9: Bảng các nhóm thành phần hỗ trợ trong App Designer.

Các nhóm thành phần	TT	Thành phần
Common Components	1	Axes
	2	Button
	3	Check Box
	4	Date Picker
	5	Drop Down
	6	Edit Field (Numeric)
	7	Edit Field (Text)
	8	Image
	9	Label
	10	List box
	11	Radio Button Group
	12	Slider

	13	Spinner
	14	State Button
	15	Table
	16	Text Area
	17	Toggle Button Group
	18	Tree
Containers and Figure Tools	1	Grid Layout
	2	Panel
	3	Tab Group
	4	Menu Bar
Instrumentation	1	Gauge
	2	90 Degree Gauge
	3	Linear Gauge
	4	Semicircular Gauge
	5	Knob
	6	Discrete Knob
	7	Lamp
	8	Switch
	9	Rocker Switch
	10	Toggle Switch

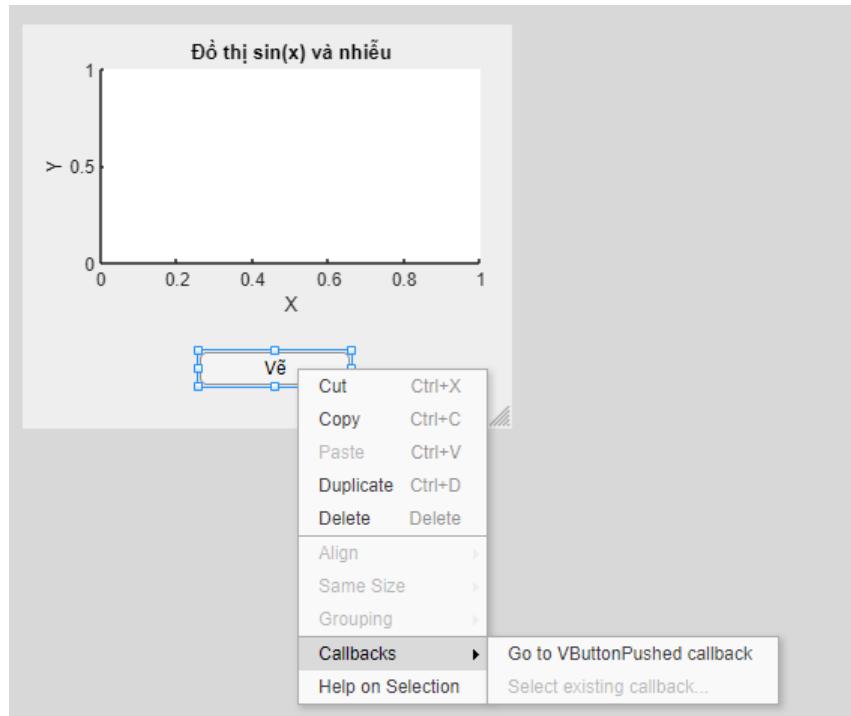
Để thay đổi các đặc tính (property) của các thành phần trong giao diện, chúng ta tham chiếu tới cột “Component Browser” rồi lựa chọn “Component”, tiếp theo là “Property” mà chúng ta muốn điều chỉnh, như ở Hình 2.21.



Hình 2.21: Giao diện Component Browser.

Sau khi thiết kế xong giao diện ở mục “Desgin View”, chúng ta sẽ lập trình chức năng cho giao diện bằng một trong hai cách sau:

- Cách 1: Nhập phải vào đối tượng và chọn “Callbacks” và chọn “Goto...” như Hình 2.22.
- Cách 2: Chuyển qua TAB “Code View” để lập trình chức năng cho giao diện.



Hình 2.22: Gọi hàm callback.

Việc lập trình trong hàm callback tương tự như lập trình Matlab, chỉ lưu ý là chúng ta tương tác với các đối tượng thông qua quai (handles) của đối tượng đó. Từ quai của đối tượng, chúng ta có thể gán, điều chỉnh tính chất của đối tượng và có thể thực thi các hàm của đối tượng.

Ví dụ 2.76

Lập trình giao diện vẽ tín hiệu hình sin và nhiễu trắng cộng trong hai chu kỳ.

- a. Đồ thị có tên là “Đồ thị sin(x) cộng nhiễu”.
- b. Sử dụng slider để điều chỉnh biên độ nhiễu và vẽ tín hiệu.

Giải: Ta thực hiện các bước như sau:

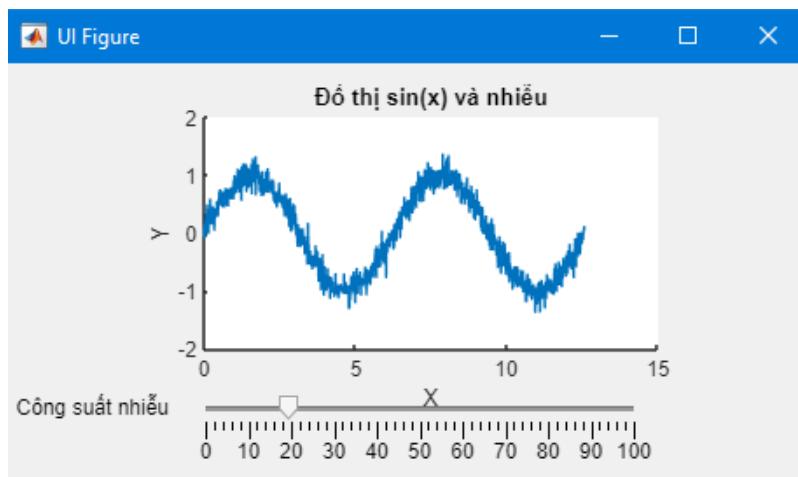
- Bước 1: Chúng ta thiết kế giao diện gồm thành phần: Axes và Slider.
 - + Đối với Axes vào mục Title: String điều chỉnh “Đồ thị sin(x) cộng nhiễu”.
 - + Đối với Slider, thay đổi Label thành “Công suất nhiễu”.
- Bước 2: Vào mục Callbacks, cập nhật như sau:

```

function CngsutnhieuSliderValueChanged(app, event)
x=0:0.01:4*pi;
y = sin(x);
nRatio = app.CngsutnhieuSlider.Value/100;
y = y + std(y).*nRatio.*randn(size(x));
plot(app.UIAxes,x,y);
end

% Value changing function: CngsutnhieuSlider
function CngsutnhieuSliderValueChanging(app, event)
changingValue = event.Value;
x=0:0.01:4*pi;
y = sin(x);
nRatio = changingValue/100;
y = y + std(y).*nRatio.*randn(size(x));
plot(app.UIAxes,x,y);
end

```

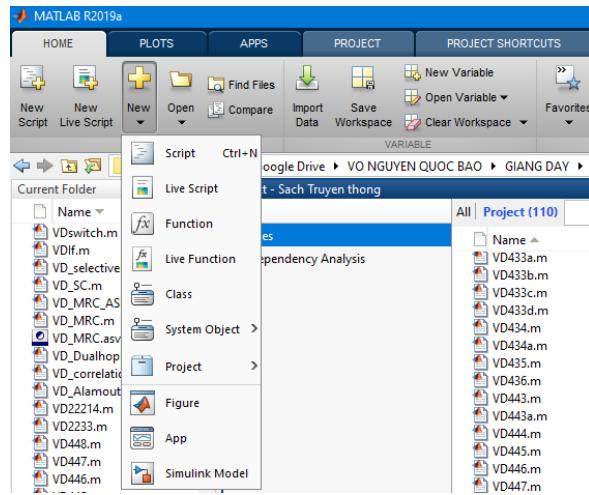


Hình 2.23: Giao diện của Ví dụ 2.76.

2.4 Cách thức lập trình trong Matlab

Do Matlab là ngôn ngữ lập trình định hướng ma trận, nên chúng ta khi lập trình phải tận dụng thế mạnh của Matlab khi xây dựng giải thuật. Việc sử dụng tư duy lập trình của ngôn ngữ khác, ví dụ như C hoặc Pascal vào trong lập trình Matlab sẽ làm tăng thời gian chạy chương trình và chương trình phức tạp hơn. Có hai cách đơn giản để thực hiện một giải thuật trong Matlab:

- Cách 1: Trực tiếp từng dòng lệnh ở cửa sổ lệnh. Cách này có ưu điểm là dễ tương tác và cho ngay kết quả trên cửa sổ lệnh. Phù hợp với các bài toán chỉ có 1 hoặc 2 dòng lệnh.
- Cách 2: Thực hiện thông qua Script hay live Script, lưu tương ứng ở dạng .m hay .mlx. Script (hay live Script), là tập hợp các dòng lệnh Matlab và có thể gọi hàm (function) của Matlab. Điểm khác của Script với function là thực hiện các dòng lệnh Matlab mà không có tham số đầu vào và giá trị trả về.



Hình 2.24: Cách thức mở một script hay live Script trên giao diện chính của Matlab bằng cách chọn "New Script" hoặc "New Live Script" hay từ "New".

Bên cạnh hai cách trên, chúng ta có thể sử dụng Simulink, lập trình giao diện, hay Apps để giải một bài toán bằng Matlab.

Một điểm lưu ý quan trọng là gần như tất cả các hàm của Matlab đều có khả năng xử lý trên ma trận và vector. Việc tự duy theo ngôn ngữ C, nghĩa là xử lý trên từng phần tử của ma trận, sẽ làm tăng thời gian chạy chương trình trên Matlab. Sau đây là một số ví dụ làm rõ hai cách lập trình và cách thức lập trình tận dụng thế mạnh của Matlab.

Ví dụ 2.77

Sử dụng Matlab tính tổng N số tự nhiên đầu tiên.

- Tự duy theo ngôn ngữ cấp thấp.
- Tự duy theo Matlab.

Giải: Bên dưới là chương trình Matlab của Ví dụ 2.77. Ta nhận thấy rằng kết quả của hai phương pháp là như nhau, tuy nhiên với cách tận dụng ưu thế của Matlab, chúng ta chỉ mất có 1 dòng lệnh so với 5 dòng lệnh của phương pháp lập trình theo ngôn ngữ C.

```
% Số lượng số tự nhiên
N = 10;

% Theo C
Tong = 0;
for n=1:N
    Tong = Tong + n;
end
disp(strcat('Tổng của 10 số tự nhiên đầu tiên theo lập trình C: ', num2str(Tong)));

% Theo Matlab
Tong = sum(1:N);
disp(strcat('Tổng 10 số tự nhiên theo lập trình Matlab: ', num2str(Tong)));
```

Chúng ta có kết quả như sau:

```
>>
Tổng của 10 số tự nhiên đầu tiên theo lập trình C là:55
Tổng 10 số tự nhiên theo lập trình Matlab là:55
```

Ví dụ 2.78

Tạo ngẫu nhiên một cột điểm sinh viên theo phân bố đều từ 0 đến 10, sau đó tính tổng số lượng sinh viên đậu môn học (≥ 5).

Giải:

```
% Số lượng sinh viên
N = 50;

% Khởi tạo bảng điểm
Diem = randi([0 10],1,N);

% Theo C
NoP=0;
for idx = 1: length(Diem)
    if Diem(idx) >= 5
        NoP = NoP + 1;
    end
end
disp(strcat('Số lượng sinh viên đậu theo lập trình C =',num2str(NoP)));

% Theo Matlab
NoP = sum(Diem >= 5);
disp(strcat('Số lượng sinh viên đậu theo lập trình Matlab =',num2str(NoP)));
```

Ví dụ 2.79

So sánh thời gian giải thuật chạy của ví dụ trên sử dụng hàm tic và toc.

Giải: Hàm tic() phải sử dụng trước hàm toc().

```
% Số lượng sinh viên
N = 10^7;

% Khởi tạo bảng điểm
Diem = randi([0 10],1,N);

% Theo Matlab
tstart = tic;
NoP=0;
for idx = 1: length(Diem)
    if Diem(idx) >= 5
        NoP = NoP + 1;
    end
end
disp(strcat('Số lượng sinh viên đậu theo C=',num2str(NoP)));
disp(strcat('Thời gian chạy theo C = ',num2str(toc(tstart))));

% Theo Matlab
tstart = tic;
```

```
NoP = sum(Diem >= 5);
disp(strcat('Số lượng sinh viên đậu theo Matlab =',num2str(NoP)));
disp(strcat('Thời gian chạy theo Matlab =',num2str(toc(tstart))));
```

Kết quả của giải thuật trên màn hình cửa sổ lệnh của Matlab như sau:

```
>>
Số lượng sinh viên đậu theo C =5451212
Thời gian chạy theo C = 0.42113
Số lượng sinh viên đậu theo Matlab =5451212
Thời gian chạy theo Matlab = 0.1102
```

Theo kết quả trên, ta thấy rằng sử dụng hàm của Matlab cho phép chúng ta tiết kiệm được gần gấp 4 lần thời gian chạy.

2.5 Giới thiệu các toolbox Matlab quan trọng khác

Phần lõi của Matlab cung cấp các hàm tính toán, lập trình và đồ họa cơ bản. Matlab còn cung cấp các Toolbox - có thể tạm gọi là các hộp công cụ - bao gồm các hàm có chức năng đặc biệt giúp chúng ta có thể xử lý các tác vụ cụ thể. Bảng 2.10 trình bày một số Toolbox quan trọng cho sinh viên và kỹ sư ngành điện tử viễn thông.

Bảng 2.10: Các Toolbox Matlab quan trọng.

TT	Tên toolbox	Tên tiếng Anh	Mô tả
1	Truyền thông	Communications Toolbox	Cung cấp các hàm để thiết kế và mô phỏng lớp vật lý của hệ thống truyền thông bao gồm: mã hóa nguồn, mã hóa sửa lỗi, kênh truyền, điều chế/giải điều chế, MIMO và OFDM
2	Xử lý tín hiệu	Signal Processing Toolbox	Cung cấp các hàm để phân tích, tiền xử lý và rút trích đặc trưng từ những tín hiệu. Toolbox cũng có các công cụ để thiết kế bộ lọc, phân tích tín hiệu và nhiều chức năng và hàm xử lý tín hiệu khác
3	Xử lý ảnh	Image Processing	Cung cấp các giải thuật và ứng dụng xử lý ảnh
4	LTE	LTE	Cung cấp các hàm để mô phỏng lớp vật lý của hệ thống LTE, LTE Advanced, LTE Advanced Pro
5	Wifi	Wifi	Cung cấp các hàm để thiết kế, mô phỏng, phân tích và kiểm chứng hệ thống WLAN không dây.

2.6 Bài tập

- Viết chương trình nhập một số bằng số và trả về số bằng chữ. Ví dụ nhập $N = 111$ trả về một trăm mười một.

2. Viết một chương trình nhập vào số bằng chữ và trả về số bằng số.
3. Viết chương trình chuyển đổi số nguyên dương thành số roman, ví dụ: nhập 10 trả về X.
4. Cho một mảng cell lưu trữ họ và tên các sinh viên, ví dụ fullname = {'Nguyen Van Tí', 'Tran Van Tho', 'Nguyen Canh Chan', 'Tran Luu Nam', 'Tan Hiep Phap'}. Hãy lập trình để trả về danh sách xếp theo thứ tự abc theo tên.
5. Nhập vào một chuỗi có ký tự và số, hãy thực hiện cộng và trả về tổng của các số. Ví dụ: 'có 2 chim xanh và 4 chim đỏ trên cành' trả về 6.
6. Cho một vector chứa một chuỗi số, hãy trả về một chuỗi số khác đếm các phần tử có trong chuỗi số. Ví dụ $x = [5, 5, 2, 1, 1, 1, 1, 3]$, trả về $y = [2, 5, 1, 2, 4, 1, 1, 3]$.
7. Cho vector A, tìm các phần tử đều xuất hiện ở các hàng.
8. Cho cấu trúc dữ liệu s. Hãy tìm dữ liệu bị mất trong cấu trúc, thể hiện bằng con số 999 và phục hồi bằng dữ liệu nội suy là trung bình của hai giá trị liền kề. Ví dụ:

$s = \{ \dots$

```
'Day Temp'
' 1 -5'
' 2 19'
' 3 1'
' 4 9999'
' 5 3'};


```

Giá trị trả về là $t = [-5 19 1 2 3]$;

Hãy áp dụng giải thuật cho dữ liệu thu thập thực tế tại:

https://www.ndbc.noaa.gov/view_text_file.php?filename=41nt1h2008.txt.gz&dir=database/historical/stdmet/

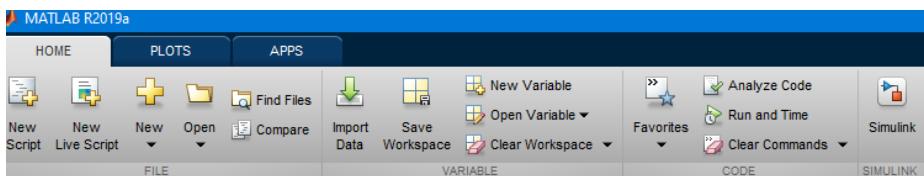
9. Cho một ma trận có kích thước $m \times n$, trả về chỉ số hàng của những hàng có cùng dạng với hàng số 1, ví dụ: Input $a = [1 2 3 0; 5 6 7 9; 2 7 8 7]$ Output b is 3. Dạng của hàng 1 là tăng (1-2), tăng (2-3), giảm (3-0).
10. Cho ma trận n hàng và 2 cột, chứa tọa độ xOy của n điểm. Hãy tìm hai điểm có khoảng cách xa nhất và trả về chỉ số hàng của hai điểm đó.
11. Bài toán tính số bước Kaprekar. Hằng số Kaprekar có giá trị là 6174. Cho một số nguyên dương bất kỳ nhỏ hơn 10000, ta có thể tiến tới hằng số Kaprekar như sau: Sắp xếp các con số của số nhập vào theo thứ tự tăng dần và giảm dần và sau đó trừ lấy chênh lệch. Lặp lại quá trình này cho đến khi mức chênh lệch bằng hằng số Kaprekar. Ví dụ, $n = 2376$, Bước 1: các con số theo thứ tự tăng dần: 7632, các con số theo thứ tự giảm dần: 2367, độ chênh lệch: $7632 - 2367 = 5265$. Bước 2: $n = 5265$, các con số theo thứ tự tăng dần: 6552, các con số theo thứ tự giảm dần: 2556, độ chênh lệch: 2996. Tiếp tục ở Bước 3...

12. Cho một chuỗi số, trả về chuỗi các số không trùng lặp. Ví dụ $x = [1 1 2 3 3]$, trả về $y = [1 2 3]$.
13. Cho một chuỗi ký tự, trả về chuỗi ký tự không trùng lặp.
14. Cho một chuỗi có chứa các dấu (), trả về kết quả là số dấu () có cân bằng không?
15. Cho một vector số, trả về xác suất xuất hiện của các phần tử trong vector.
16. Bài toán Goldbach chỉ ra rằng một số nguyên bất kỳ N sẽ là tổng của hai số nguyên tố p_1 và p_2 . Nhập vào số nguyên bất kỳ và trả về các cặp số p_1 và p_2 .
17. Nhập vào một chuỗi ký tự bất kỳ và tiến hành mã hóa bằng cách dịch về phía phải 13 vị trí trong bảng mã ASCII.
18. Cho một số dương bất kỳ, hãy kiểm tra số này có cân bằng hay không biết rằng một số được gọi là cân bằng nếu tổng các số bên trái bằng tổng các số bên phải. Ví dụ: 13722 là số cân bằng vì $1+3=2+2$.
19. Nhập vào một vector có một số phần tử là NaN. Hãy thay thế các phần tử NaN trong vector bằng giá trị liền trước.
20. Nhập vào một ma trận bất kỳ, kiểm tra và thực hiện xóa hàng nếu hàng có chứa giá trị NaN.
21. Nhập vào một vector số nguyên, trả về giá trị logic là true nếu tồn tại một số là bình phương của một số khác trong vector. Ngược lại là 0.
22. Nhập vào một câu và loại bỏ tất cả các nguyên âm trong câu.
23. Cho số n , hãy tính tổng các con số tạo nên bình phương của n . Ví dụ, $n=5$, thì $n^2 = 25$, kết quả trả về là 7.
24. Cho một ma trận bất kỳ, hãy hoán chuyển cột cuối với cột một.
25. Cho ma trận bất kỳ, hãy tìm các phần tử nhỏ hơn trung bình của ma trận và thay thế bằng giá trị lớn nhất trong ma trận.
26. Cho một vector bất kỳ, trả về 1 nếu vector đồng biến, 2 nếu nghịch biến và 0 nếu không đồng biến cũng như nghịch biến.
27. Tính tổng của n số Fibonacci đầu tiên.
28. Hãy tạo ra bảng cửu chương từ 1 đến 20.
29. Viết chương trình có giao diện giải phương trình bậc 2 tổng quát.
30. Hãy sử dụng hàm profile cho Bài tập 27 và 28.

CHƯƠNG 3: GIỚI THIỆU VỀ SIMULINK

3.1 Giới thiệu

Simulink là một công cụ đồ họa mở rộng trong Matlab để mô hình hóa và mô phỏng hệ thống. Một trong những lợi thế chính của Simulink là khả năng tương tác, điều chỉnh và cài đặt qua các khối chức năng mà không phải thực hiện với các dòng lệnh.



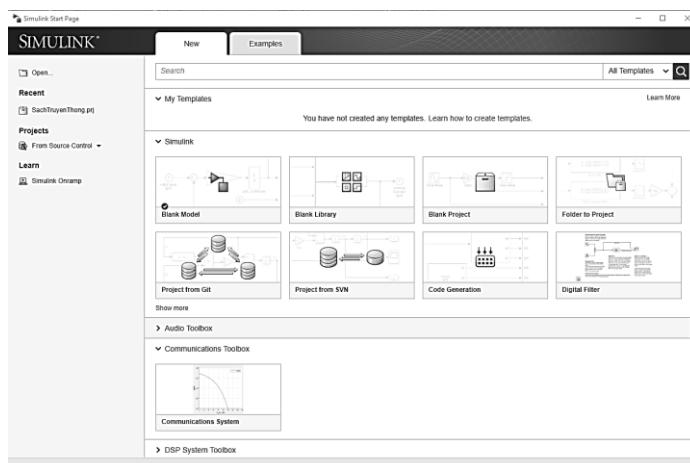
Hình 3.1: Khởi động Simulink ở trên giao diện chính (nút bên phải).

Môi trường Simulink cho phép chúng ta thực hiện “gấp và thả” các khối chức năng để xây dựng mô hình mô phỏng. Nhiều khối chức năng và/hoặc phần tử của sơ đồ khối đã được Simulink hỗ trợ sẵn, như các hàm truyền, toán tử cơ bản,..., cũng như các thiết bị vào/ra như bộ tạo hàm và máy hiện sóng. Simulink được tích hợp với Matlab và các dữ liệu giữa Simulink và Matlab có thể dễ dàng chuyển đổi với nhau.

3.2 Simulink

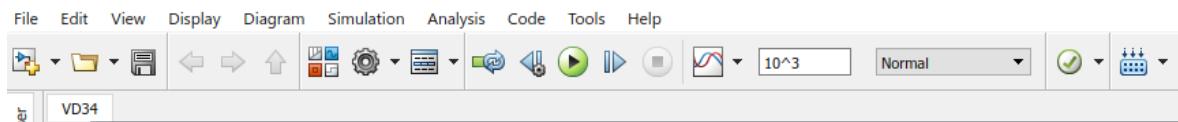
3.2.1 Khởi động Simulink

Chúng ta có thể khởi động Simulink bằng hai cách: nhập lệnh “Simulink” trực tiếp ở cửa sổ lệnh hay là nhấn vào nút Simulink ở giao diện chính của Matlab như Hình 3.1.



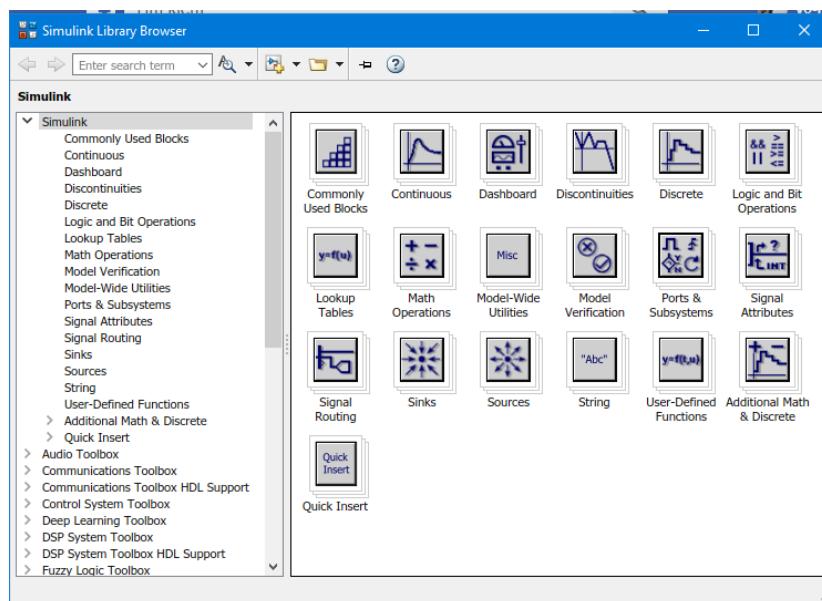
Hình 3.2: Giao diện chính của Simulink.

Trang bắt đầu của Simulink trình bày như Hình 3.2 nếu Matlab có cài đặt Simulink. Từ giao diện chính, chúng ta có thể bắt đầu với một mô hình mới hay mô hình đang triển khai. Để làm việc với Simulink, chúng ta cần biết về thanh công cụ Simulink như Hình 3.3 bao gồm 3 chức năng chính là: (i) Cài đặt mô hình, (ii) Chạy mô hình và (iii) Xây dựng mô hình.



Hình 3.3: Thanh công cụ của Simulink với nút đầu tiên là nút Library Browser để mở thư viện Simulink.

3.2.2 Các thành phần cơ bản của Simulink



Hình 3.4: Các khối thư viện cơ bản của Simulink.

Bảng 3.1: Thư viện khối của Simulink.

TT	Tên thư viện	Thuyết minh
1	Continuous	Các khối chức năng cho hàm liên tục, ví dụ như đạo hàm hay tích phân
2	Dashboard	Các khối chỉ thị hay điều khiển để tương tác với mô phỏng
3	Discontinuities	Khối chức năng cho hàm không liên tục như hàm bão hòa
4	Discrete	Khối các hàm chức năng cho thời gian rời rạc, ví dụ như khối trễ
5	Logic and Bit Operations	Các khối toán tử cho bit hoặc toán tử logic
6	Lookup Tables	Các khối bảng truy vấn như hàm cos hay sin

7	Math Operations	Các khối thực hiện chức năng toán học như độ lợi, nhân hay tổng
8	Model Verification	Các khối để tự kiểm chứng mô hình như kiểm chứng giá trị đầu vào
9	Model-Wide Utilities	Các khối cho mô hình
10	Ports and Subsystems	Các khối liên quan đến các hệ thống con
11	Signal Attributes	Các khối xử lý tính chất tín hiệu như chuyển đổi kiểu dữ liệu
12	Signal Routing	Các khối định tuyến tín hiệu
13	Sinks	Các khối đích
14	Sources	Các khối nguồn
15	String	Khối xử lý chuỗi
16	User-Defined Functions	Khối chức năng người dùng tự định nghĩa

Mô hình Simulink bao gồm: các khối (block) được chứa trong các thư viện, các đường kết nối (lines) và các chú thích (annotation).

- Các khối: có thể là các hàm toán học, có thể có nhiều biến vào và nhiều biến ra, thực hiện chức năng cụ thể, ví dụ như: tạo, điều chỉnh, biến đổi,..., trên tín hiệu. Các khối được phân loại trong thư viện theo đặc tính hay chức năng của các khối. Hình 3.4 và Bảng 3.3 liệt kê các khối thư viện của Simulink.
- Các đường tín hiệu: là các kết nối giữa các khối, truyền tín hiệu hay giá trị giữa các khối. Tín hiệu có nhiều kiểu dữ liệu khác nhau: kiểu số, vector hay ma trận.
- Các chú thích: bao gồm dạng hình ảnh hay dạng văn bản được bổ sung vào mô hình nhằm thuyết minh cho mô hình mô phỏng.

3.2.3 Các bước xây dựng mô hình Simulink

Để bắt đầu xây dựng một mô hình Simulink, chúng ta thực hiện các bước sau:

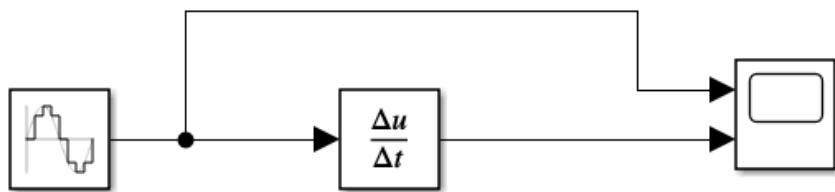
- Bước 1: Mở thư viện Simulink bằng cách nhấn vào nút “Library Browser” như Hình 3.3;
- Bước 2: Dùng phương pháp “gấp và thả” các khối ở “Simulink Library Browser” vào mô hình;
- Bước 3: Kết nối các khối ở Bước 2 theo mô hình đã thiết kế;
- Bước 4: Điều chỉnh các tham số của các khối theo thiết kế;
- Bước 5: Điều chỉnh tham số chạy và chạy chương trình.

Ví dụ 3.1

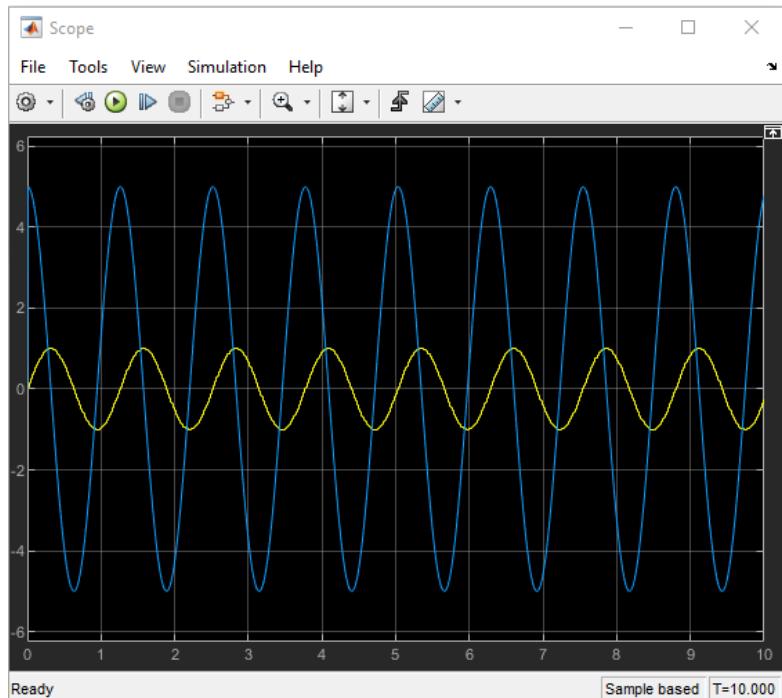
Tạo tín hiệu sin và cho qua khối đạo hàm. Xuất cả hai tín hiệu lên máy hiện sóng.

Giải: Chúng ta thực hiện các bước tương tự như trên:

- Bước 1: Xây dựng mô hình bằng cách chọn các khối
 - + Khối “Sine Wave” ở “Source”
 - + Khối “Derivative” ở “Continuous”
 - + Khối “Scope” ở “Sinks”
- Bước 2: Kết nối các khối;
- Bước 3: Điều chỉnh thông số các khối;
- Bước 4: Điều chỉnh tham số mô phỏng của mô hình cho phù hợp và thực hiện chạy mô phỏng.



Hình 3.5: Mô hình của Ví dụ 3.1.



Hình 3.6: Tín hiệu ngõ ra của Ví dụ 3.1

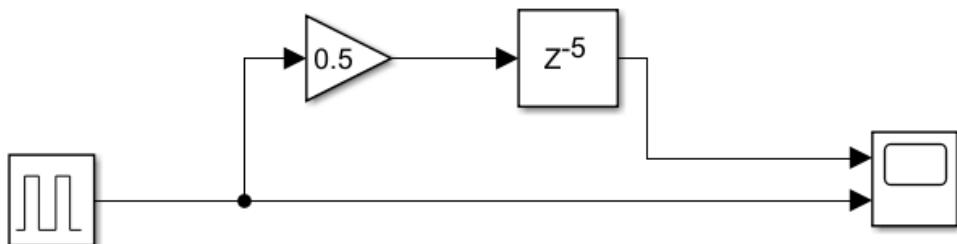
Ví dụ 3.2

Tạo một tín hiệu xung vuông có biên độ 1V, thực hiện nhân với bộ suy hao 1/2 biên độ và cho qua bộ trễ 1/2 chu kỳ. Xuất cả hai tín hiệu ra máy hiện sóng để quan sát.

Giải: Chúng ta chọn các khối như Bảng 3.2 và kết nối các khối như Hình 3.7.

Bảng 3.2: Danh mục các khối cho Ví dụ 3.2.

TT	Khối	Thư viện
1	Pulse Generator	Sources
2	Gain	Commonly Used Blocks
3	Delay	Commonly Used Blocks
4	Scope	Sinks

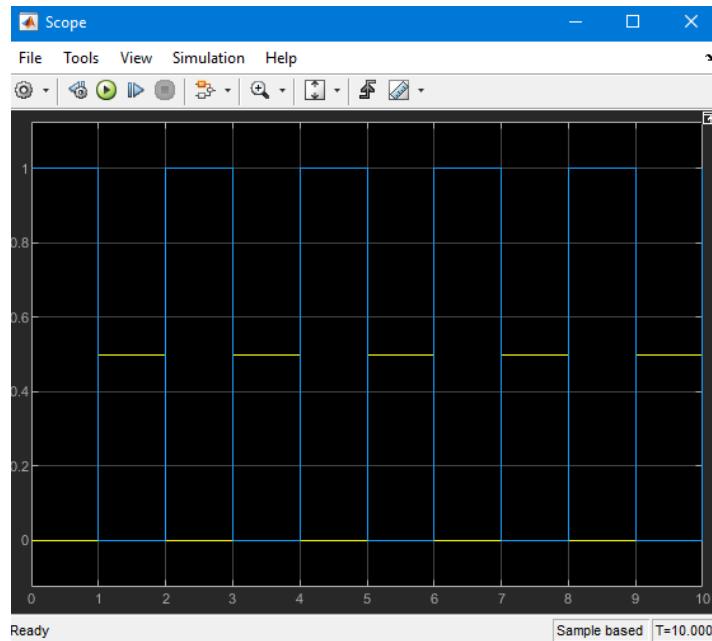


Hình 3.7: Sơ đồ khối cho Ví dụ 3.2.

Sau khi kết nối các khối, chúng ta lần lượt thiết lập thông số cho các khối để đáp ứng yêu cầu của bài toán. Chúng ta thiết lập các khối tuần tự như sau:

- Khối Pulse Generator
 - + Amplitude: 1
 - + Periods (number of samples): 10
 - + Pulse width (number of samples): 5
 - + Phase delay (number of samples): 0
- Khối Gain
 - + Gain: 0.5
- Khối Delay:
 - + Delay length (Dialog): 5
 - + Initial condition: 0.0
 - + Input processing: Elements as channels (samples based)

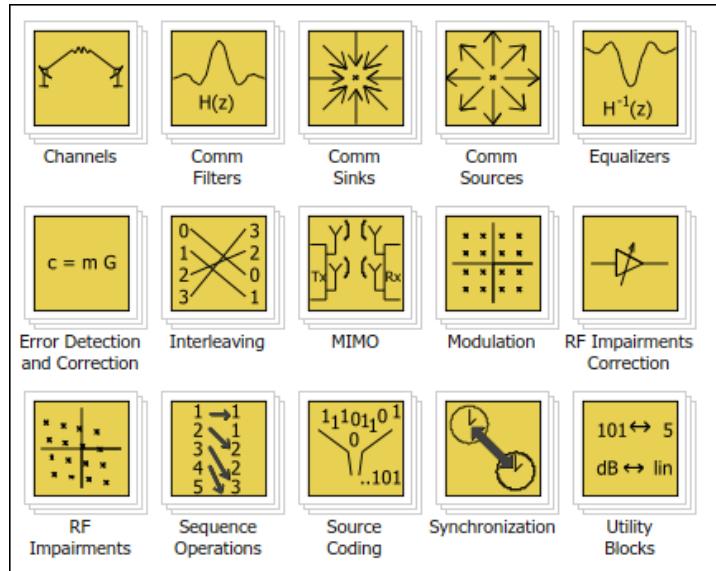
Bước tiếp theo là thiết lập thông số mô phỏng cho hệ thống, chọn Simulation stop time bằng 100 (samples) và thực hiện lệnh chạy. Kết quả là Hình 3.8.



Hình 3.8: Tín hiệu ở ngõ ra của Ví dụ 3.2.

3.3 Simulink Communication Toolbox

Communications Toolbox trong Simulink về cơ bản tổ chức giống như Communications Toolbox của Matlab. Chúng ta có thể sử dụng Simulink để mô phỏng các chức năng của hệ thống truyền thông, ví dụ như mã hóa nguồn, điều chế/giải điều chế, mô phỏng kênh truyền, cũng như thiết kế và phân tích hiệu năng các hệ thống thông tin số và thông tin vô tuyến.



Hình 3.9: Các khối của Communications Toolbox trong Simulink.

Communications Toolbox tổ chức thành các thư viện như

Bảng 3.3.

Bảng 3.3: Các khối thư viện của Simulink Communication Toolbox.

TT	Thư viện khối	Thuyết minh
1	Channels	Bao gồm các khối kênh truyền: nhiễu trắng, nhị phân đối xứng, MIMO và SISO fading.
2	Comm Filter	Bao gồm các khối bộ lọc.
3	Comm Sinks	Bao gồm các khối nhận và phân tích dữ liệu của hệ thống truyền thông, ví dụ như: khối biểu đồ chòm sao, khối tính tỷ lệ lỗi bit, khối biểu đồ mắt.
4	Comm Sources	Chia ra làm hai loại. Loại 1 là các khối tạo tín hiệu ngẫu nhiên. Loại 2 là các khối tạo mã.
5	Equalizers	Bao gồm các khối cân bằng: hồi tiếp, tuyển tính và MLSE.
6	Error Detection and Correction	Khối mã hóa: mã hóa khối, chập và CRC.
7	Interleaving	Khối trộn dữ liệu.
8	MIMO	Khối MIMO bao gồm kênh truyền MIMO và mã không gian thời gian.
9	Modulation	Khối điều chế, chia ra làm 3 mục: điều chế tương tự băng gốc, điều chế tương tự băng dải và điều chế số băng gốc.
10	RF Impairments	Các hiệu ứng của kênh truyền: suy hao, mất cân bằng, không tuyển tính, lệch pha, nhiễu pha và nhiễu nhiệt.
11	RF Impairments Correction	Bù/sửa các hiệu ứng của kênh truyền.
12	Sequence Operations	Các phép toán tuần tự.
13	Source Coding	Mã hóa nguồn tin.
14	Synchronization	Đồng bộ.
15	Utility Blocks	Các khối chức năng hữu dụng khác.

Ví dụ 3.3

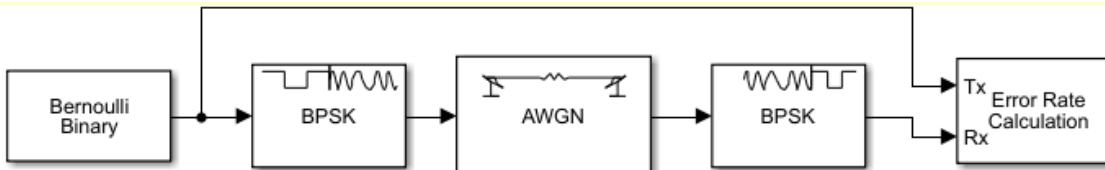
Sử dụng Simulink Communications Toolbox, mô phỏng kênh truyền nhiễu trắng cho điều chế BPSK và so sánh kết quả với lý thuyết.

Giải: Chúng ta xây dựng hệ thống như Bảng 3.4.

Bảng 3.4: Các khối xây dựng cho Ví dụ 3.2.

TT	Khối	Thư viện
1	Bernoulli Binary	Comm Sources/Random Data Sources
2	BPSK Modulator	Modulation/PM/BPSK Modulator Baseband
3	AWGN	Channel

4	BPSK Demodulator	Modulation/PM/BPSK Demodulator Baseband
5	Error Rate Calculation	Comm Sinks



Hình 3.10: Mô hình hệ thống của Ví dụ 3.3.

Với khối kênh truyền nhiễu trắng, chúng ta thiết lập Eb/No (dB) hai giá trị 0 dB và 3 dB, cho mỗi lần chạy. Với “Simulation stop time”, chúng ta chọn 10^6 cho cả hai lần chạy. Với điều chế BPSK ở kênh truyền nhiễu trắng, tỷ lệ lỗi bit là $Q(\sqrt{2\gamma})$, nên ta có kết quả ở Bảng 3.5.

Bảng 3.5: Bảng kết quả của Ví dụ 3.3.

Lần chạy	Eb/No (dB)	Kết quả mô phỏng	Kết quả lý thuyết
1	0	0.0785	0.0786
2	3	0.0229	0.0228

Lưu ý là kết quả mô phỏng của mô hình trong Ví dụ 3.3 được lưu ở Workspace của Matlab với tên biến là ErrorVec. ErrorVec là một vector ba phần tử, phần tử đầu tiên là tỷ số tín hiệu trên nhiễu, phần tử thứ hai là tổng số lỗi và phần tử thứ ba là tổng số bit truyền. Tên biến này có thể thay đổi trong phần Block Parameters của khối Error Rate Calculation.

3.4 Bài tập

- Sử dụng Simulink để thực hiện hệ thống mã hóa nguồn tín hiệu A law cho tín hiệu sin.
- Sử dụng Simulink để thực hiện hệ thống mã hóa nguồn tín hiệu mu-law cho tín hiệu sin.
- Sử dụng Simulink để thực hiện điều chế/giải điều chế tương tự FM băng dài.
- Sử dụng Simulink để thực hiện điều chế/giải điều chế tương tự PM băng dài.
- Sử dụng Simulink để thực hiện điều chế/giải điều chế tương tự SSB AM băng dài.
- Sử dụng Simulink để thực hiện điều chế số MPSK trên kênh truyền nhiễu trắng.
- Sử dụng Simulink để thực hiện điều chế số MPSK trên kênh truyền fading Rayleigh.
- Sử dụng Simulink để thực hiện tính xác suất dừng cho kênh truyền fading Rayleigh.
- Sử dụng Simulink để thực hiện điều chế số MQAM trên kênh truyền nhiễu trắng.
- Sử dụng Simulink để mô phỏng hệ thống mã hóa/giải mã tuyến tính trên
 - Kênh truyền nhị phân đối xứng.

- b. Kênh truyền nhiễu trăng.
11. Sử dụng Simulink để mô phỏng hệ thống mã hóa/giải mã BCH trên
- a. Kênh truyền nhị phân.
 - b. Kênh truyền nhiễu trăng.
12. Sử dụng Simulink để mô phỏng hệ thống mã hóa/giải mã RS trên
- a. Kênh truyền nhị phân.
 - b. Kênh truyền nhiễu trăng.
13. Sử dụng Simulink để mô phỏng hệ thống mã hóa/giải mã chập trên
- a. Kênh truyền nhị phân.
 - b. Kênh truyền nhiễu trăng.
 - c. Kênh truyền fading Rayleigh.
 - d. Kênh truyền fading Rician.

CHƯƠNG 4: MÔ PHỎNG TÍN HIỆU VÀ QUÁ TRÌNH THU PHÁT

4.1 Biến ngẫu nhiên và tạo ra biến ngẫu nhiên

4.1.1 Biến ngẫu nhiên

4.1.1.1 Tạo ra số giả ngẫu nhiên

Để mô phỏng trên máy tính, việc tạo ra số giả ngẫu nhiên là một trong những bước cơ bản đầu tiên. Khi chưa có máy tính, số giả ngẫu nhiên thường được tạo thông qua các phương pháp như quay bánh xe, chia bài, hoặc xoay đĩa. Tuy nhiên, kể từ khi có máy tính thì mô phỏng thường được thực hiện trên máy tính với các giải thuật tạo số giả ngẫu nhiên.

Số giả ngẫu nhiên là một chuỗi số được sinh ra một cách xác định và có đặc tính phân bố đều trong khoảng 0 và 1.

Một trong những giải thuật cơ bản nhất để tạo ra số giả ngẫu nhiên là giải thuật đệ quy với giá trị bắt đầu là x_1 (còn gọi là seed) [3]. Giá trị x_n được tính theo giải thuật sau:

$$x_n = ax_{n-1} \text{ modulo } m \quad (4.1)$$

với a và m là các số nguyên dương cho trước và $n \geq 2$. Khi đó, ta dễ dàng thấy rằng x_n sẽ nhận các giá trị từ 0 đến $m-1$ và x_n/m sẽ là một xấp xỉ tốt của phân bố đều trong khoảng 0 đến 1.

Ví dụ 4.1

Tạo N biến phân bố đều theo giải thuật như trên và so sánh kết quả đạt được với hàm rand() của Matlab.

Giải: Lưu ý là giải thuật tạo biến ngẫu nhiên ở trên là giải thuật đệ quy nên chúng ta không thể áp dụng giải thuật cho tất cả các phần tử trên vector đồng thời.

```
clc;
N = 1000;

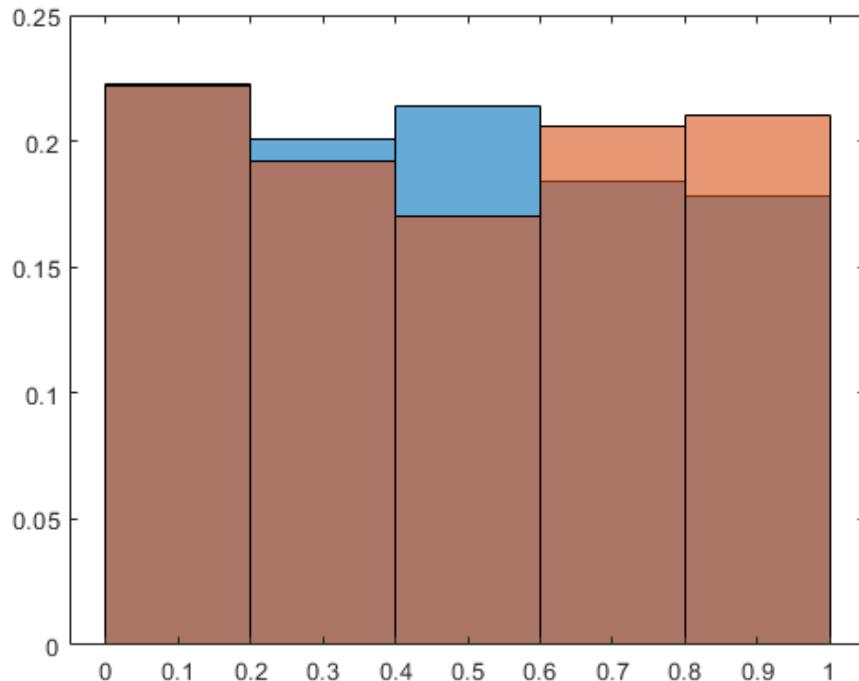
% Giải thuật đệ quy
x = zeros(1,N);
m = 10;
a = 3;
x(1) = 3.4;
for n = 2:N
    x(n) = urand(n,a,m,x(n-1));
end
x=x/m;
```

```

% Số lượng phân đoạn trực x của biểu đồ tần suất
NoB = 5;
histogram(x,NoB,'Normalization','Probability');
hold on
histogram(rand(1,N),NoB,'Normalization','Probability');

function out = urand(n,a,m,x0)
if n == 0
    out = x0;
else
    out = mod(a*urand(n-1,a,m,x0),m);
end
end

```



Hình 4.1: Biểu đồ tần suất của Ví dụ 4.1.

Ví dụ 4.2

Chạy lại Ví dụ 4.1 với 10 phần tử.

- Cho $x_1 = 3.4$, chạy giải thuật nhiều lần, quan sát chuỗi giá trị nhận được.
- Cho $x_1 = x_2 m$ ở câu a), quan sát chuỗi giá trị nhận được và nhận xét.

Giải: Khi $x_1 = 3.4$, thì chuỗi giá trị nhận cho các lần chạy luôn là

```
x =
```

0.3400	0.0600	0.6200	0.2200	0.4600	0.3400	0.5798	0.1950	0.4796	0.9977
--------	--------	--------	--------	--------	--------	--------	--------	--------	--------

Khi $x_1 = x_2, m = 0.6$, thì chuỗi nhận được là

$x =$

0.0600	0.5400	0.5800	0.9800	0.1400	0.0600	0.2200	0.2883	0.2977	0.0886
--------	--------	--------	--------	--------	--------	--------	--------	--------	--------

Nhận xét: Khi cho $x_1 = x_2, m = 0.6$, chúng ta mong đợi là x_2 của câu b) sẽ bằng x_3 của câu a), nhưng kết quả không như mong đợi. Nghĩa là với mỗi một x_1 (seed) khác nhau chúng ta sẽ có một chuỗi khác nhau và ngay cả khi chúng ta bắt được một giá trị bất kỳ trong chuỗi chúng ta cũng không thể tái tạo lại chuỗi ngẫu nhiên.

4.1.1.2 Tạo số ngẫu nhiên rời rạc với hàm xác suất khồi cho trước

Để tạo số ngẫu nhiên rời rạc, ta có 3 phương pháp như sau:

- Phương pháp biến đổi ngược (Inverse transform method).
- Phương pháp chấp nhận – loại trừ (Acceptance – Rejection technique).
- Phương pháp tổng hợp (Composition approach).

4.1.1.2.1 Phương pháp biến đổi ngược

Chúng ta muốn tạo ra biến ngẫu nhiên rời rạc X có hàm khồi xác suất (PMF) là

$$\Pr(X = x_j) = p_j, j = 0, 1, \dots, \sum_j p_j = 1. \quad (4.2)$$

Các bước thực hiện như sau:

- Bước 1: Tạo biến ngẫu nhiên phân bố đều U trong khoảng $(0,1)$

$$X = \begin{cases} x_0, & U < p_0 \\ x_1, & p_0 < U < p_0 + p_1 \\ M & \\ x_j, & \sum_{i=0}^{j-1} p_i < U < \sum_{i=0}^j p_i \\ M & \end{cases}. \quad (4.3)$$

- Bước 2: Cho

$$\Pr(X = x_j) = \Pr\left(\sum_{i=0}^{j-1} p_i \leq U < \sum_{i=0}^j p_i\right) = p_j. \quad (4.4)$$

thì X có hàm khồi xác suất (PMF) như mong muốn.

Ví dụ 4.3

Tạo biến ngẫu nhiên rời rạc X nhận giá trị 1, 2, 3 và 4 có xác suất phân bố tương ứng như sau: $p_1 = 0.2$, $p_2 = 0.15$, $p_3 = 0.25$ và $p_4 = 0.4$.

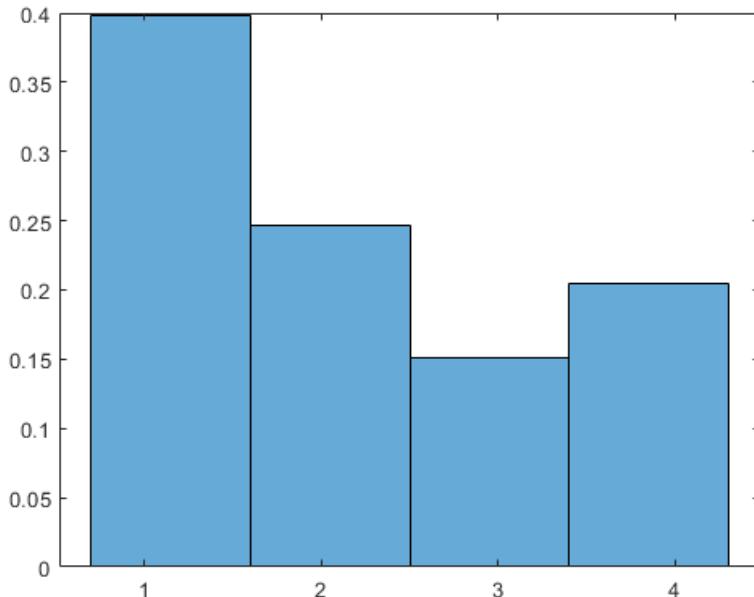
Giải: Chúng ta sử dụng vòng lặp for và hàm histogram() như sau.

```
% Số lượng mẫu của biến ngẫu nhiên
N = 10^4;

% Khởi tạo biến ngẫu nhiên phân bố đều trong khoảng (0,1)
U = rand(1,N);
p = [0.2 0.15 0.25 0.4];

%
pc = cumsum(p);
X = zeros(1,N);
for idx = 1:length(pc)
    X = X + (U <= pc(idx));
End

% Vẽ đồ thị phân bố
histogram(X,4,'Normalization','probability');
xticks([1 2 3 4]);
set(gcf,'color','w');
```



Hình 4.2: Hàm mật độ phân bố xác suất của Ví dụ 4.3.

4.1.1.2.2 Phương pháp chấp nhận - loại trừ (Acceptance - Rejection technique)

Giả sử ta có phương pháp hiệu quả để tạo ra biến ngẫu nhiên có hàm phân bố khối $\{q_j, j \geq 0\}$, khi đó ta có thể tận dụng phương pháp này để tạo ra biến ngẫu nhiên Y có hàm phân bố khối $\{p_j, j \geq 0\}$ bằng cách trước tiên tạo ra biến ngẫu nhiên Y có hàm phân bố khối $\{q_j, j \geq 0\}$

và chấp nhận giá trị với xác suất tỷ lệ với $\frac{p_j}{q_j}$. Cụ thể, gọi c là một hằng số mà:

$$\frac{p_j}{q_j} \leq c \quad \forall j, \quad p_j > 0. \quad (4.5)$$

Giải thuật ba bước của phương pháp chấp nhận và loại trừ như sau:

- Bước 1: Tạo biến ngẫu nhiên Y có hàm phân bố khối $\{q_j, j \geq 0\}$.
- Bước 2: Tạo biến ngẫu nhiên U phân bố đều trong khoảng $[0, 1]$.
- Bước 3: Nếu $U < p_Y/cq_Y$, cho $X = Y$. Nếu không, lặp lại Bước 1.

Ví dụ 4.4

Tạo biến ngẫu nhiên rời rạc có giá trị từ 1 đến 10, với xác suất tương ứng là 0.11, 0.12, 0.09, 0.08, 0.12, 0.10, 0.09, 0.09, 0.10 và 0.10.

Giải: Ta chọn c như sau:

$$c = \max_j \frac{p_j}{q_j} = 1.2. \quad (4.6)$$

- Bước 1: Tạo biến ngẫu nhiên phân bố đều U_1 trong khoảng $(0,1)$ và đặt $Y = \lfloor 10U_1 \rfloor + 1$ với $\lfloor . \rfloor$ là hàm lấy phần nguyên gần nhất bên trái.
- Bước 2: Tạo biến ngẫu nhiên phân bố đều U_2 .
- Bước 3: Nếu $U_2 \leq \frac{p_Y}{0.12}$, cho $X = Y$ và dừng. Ngược lại, quay lại Bước 1.

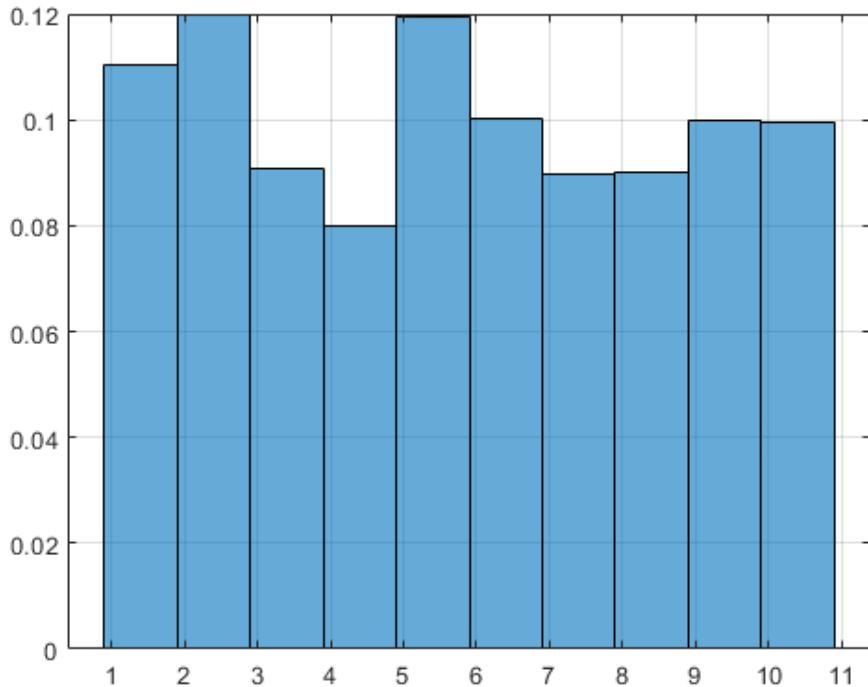
```
% Số lượng mẫu của biến ngẫu nhiên
N = 10^6;

p = [0.11 0.12 0.09 0.08 0.12 0.1 0.09 0.09 0.1 0.1];
c = max(p);
%
% Bước 1
% Khởi tạo biến ngẫu nhiên phân bố đều trong khoảng (0,1)
U1 = rand(1,N);
% Dùng hàm floor() để lấy phần nguyên bên trái
Y = floor(10*U1) + 1;

% Bước 2
U2 = rand(1,N);

% Bước 3
X = zeros(1,N);
for idx = 1:length(p)
    X = Y.*(U2 <= p(Y)/.12);
end
X=X(X>0);
```

```
% Vẽ đồ thị phân bố
histogram(X,10,'Normalization','probability');
grid on
set(gcf,'color','white');
```



Hình 4.3: Đồ thị tần suất của biến ngẫu nhiên sinh ra theo phương pháp chấp nhận loại trừ.

4.1.1.2.3 Phương pháp tổng hợp (Composition approach)

Phương pháp tổng hợp là phương pháp tạo một biến ngẫu nhiên rời rạc khi đã biết trước hai biến ngẫu nhiên X_1 với hàm phân bố khói $\{p_j^{(1)}, j \geq 0\}$ và biến ngẫu nhiên X_2 với hàm phân bố khói $\{p_j^{(2)}, j \geq 0\}$. Biến ngẫu nhiên X có hàm phân bố khói

$$\Pr(X = j) = \alpha p_j^{(1)} + (1 - \alpha)p_j^{(2)}, \quad j \geq 0 \quad (4.7)$$

được tạo như sau:

$$X = \begin{cases} X_1, & \alpha \\ X_2, & 1 - \alpha \end{cases}. \quad (4.8)$$

Ví dụ 4.5

Hãy tạo biến ngẫu nhiên Y có hàm phân phối khói như sau:

$$\Pr(Y = j) = \begin{cases} 0.05, & j = 1, 2, 3, 4, 5 \\ 0.15, & j = 6, 7, 8, 9, 10 \end{cases}. \quad (4.9)$$

Giải: Ta chuyển đổi bài toán hiện tại thành bài toán với hai biến ngẫu nhiên X_1 và X_2 bằng cách để ý rằng $p_j = 0.5 p_j^{(1)} + 0.5 p_j^{(2)}$, $j \geq 0$, do đó ta chọn

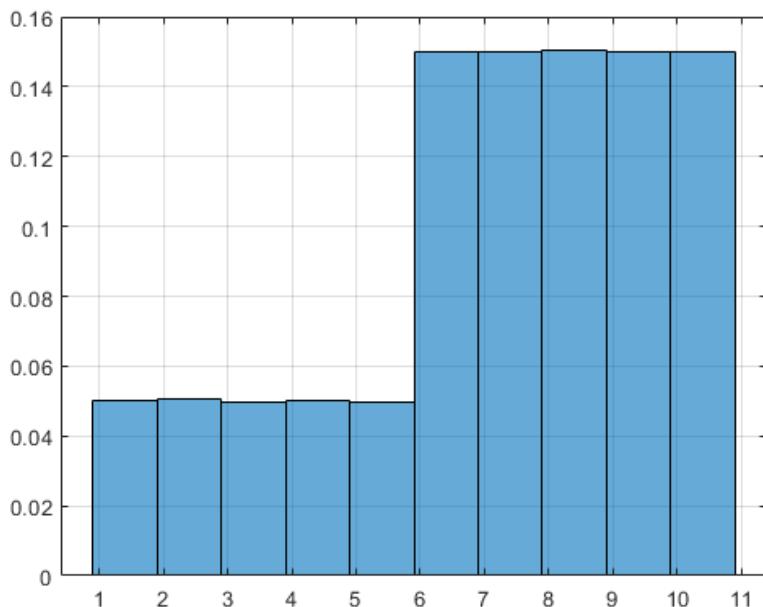
$$p_j^{(1)} = 0.1, \quad j = 1, \dots, 10, \quad (4.10)$$

$$p_j^{(2)} = \begin{cases} 0 & j = 1, \dots, 5 \\ 0.2 & j = 6, 7, 8, 9, 10 \end{cases}. \quad (4.11)$$

Giải thuật để tạo ra Y như sau:

- Bước 1: Tạo biến ngẫu nhiên U_1 .
- Bước 2: Tạo biến ngẫu nhiên U_2 .
- Bước 3: Nếu $U_1 < 0.5$, cho $Y = \lfloor 10U_2 \rfloor + 1$, ngược lại, $Y = \lfloor 5U_2 \rfloor + 6$.

```
% Khởi tạo
N = 10^6;
% Bước 1
U1 = rand(1,N);
% Bước 2
U2 = rand(1,N);
% Bước 3
Y = (U1 < 0.5).*floor(10*U2) + 1 + (U1 >= 0.5).*floor(5*U2) + 6;
% Vẽ biểu đồ tần suất
histogram(Y,10,'Normalization','probability');
grid on
set(gcf,'color','white');
```



Hình 4.4: Đồ thị tần suất của biến ngẫu nhiên trong Ví dụ 4.5.

4.1.1.3 Tạo số ngẫu nhiên liên tục cho hàm ngẫu nhiên

Để tạo biến ngẫu nhiên liên tục, chúng ta có hai phương pháp sau:

- Giải thuật biến đổi ngược (Inverse Transform Algorithm).
- Giải thuật loại bỏ (Rejection method).

4.1.1.3.1 Phương pháp biến đổi ngược

Cho U là biến ngẫu nhiên phân bố đều trong khoảng $(0,1)$. Với bất kỳ hàm phân phối F , biến ngẫu nhiên X được định nghĩa như sau [3]:

$$X = F^{-1}(U) \quad (4.12)$$

sẽ có phân phối F với F^{-1} là hàm ngược của hàm F .

Ví dụ 4.6

Hãy tạo biến ngẫu nhiên có phân bố mũ với hàm phân phối xác suất tích lũy như sau:
 $F(x) = 1 - e^{-x}$.

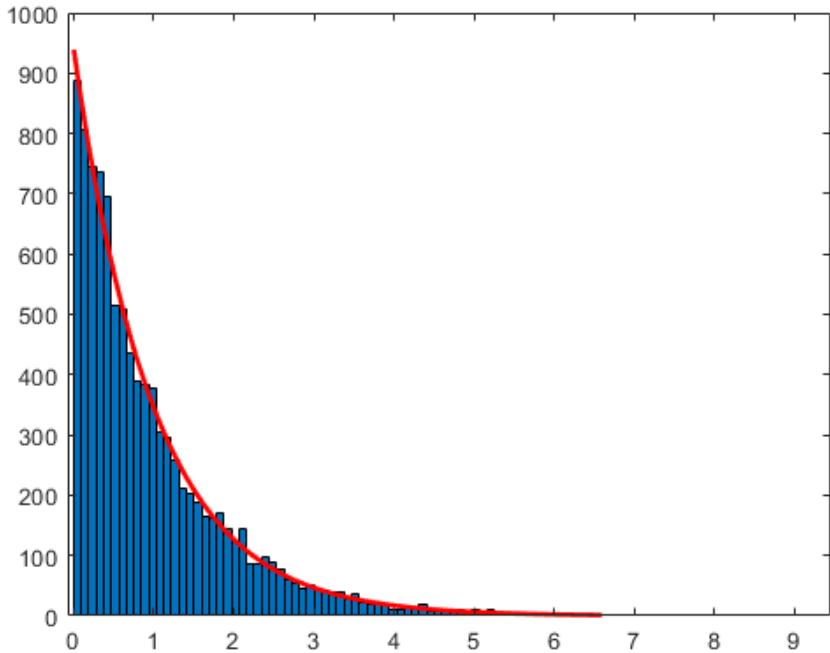
Giải: Phương pháp biến đổi ngược có một nhược điểm là chỉ áp dụng được khi hàm ngược của hàm phân bố xác suất tích lũy tồn tại. Từ $F(x) = 1 - e^{-x} = u$, ta có $x = -\log(1-u)$.

Giải thuật gồm ba bước sau:

- Bước 1: Tạo biến ngẫu nhiên U phân bố đều trong khoảng $(0,1)$.
- Bước 2: Tính $X = -\log(1-U)$.
- Bước 3: Lặp lại hai bước trên khi đủ số mẫu cần thiết.

Chương trình Matlab thực hiện phương pháp biến đổi ngược với giải thuật ba bước như sau:

```
% Số lượng mẫu cần tạo
N = 10^4;
% Tạo biến phân bố đều
U = rand(1,N);
% Tạo biến phân bố theo hàm CDF
X = -log(1-U);
% Vẽ hàm mật độ phân phối để kiểm chứng
histfit(X,100,'exp')
```



Hình 4.5 Hình biểu đồ tần suất của biến ngẫu nhiên tạo bằng giải thuật biến đổi ngược.

4.1.1.3.2 Giải thuật loại bỏ

Giả sử ta có thể tạo ra biến ngẫu nhiên có hàm mật độ xác suất $g(x)$, giải thuật loại bỏ cho phép tạo ra biến ngẫu nhiên Y có hàm mật độ xác suất $f(x)$ từ $g(x)$, cụ thể là

$$\frac{f(y)}{g(y)} \leq c \quad \forall y. \quad (4.13)$$

Giải thuật loại bỏ có thể tóm tắt trong ba bước như sau:

- Bước 1: Tạo biến ngẫu nhiên Y có mật độ phân bố $g(x)$.
- Bước 2: Tạo ra biến ngẫu nhiên X có mật độ phân bố $f(x)$.
- Bước 3: Nếu $U \leq \frac{f(Y)}{cg(Y)}$, cho $X = Y$. Ngược lại, quay lại Bước 1.

Nhìn vào giải thuật, chúng ta cần phải xác định giá trị c , cụ thể:

$$c = \max_x \frac{f(x)}{g(x)}. \quad (4.14)$$

Ví dụ 4.7

Tạo biến ngẫu nhiên có hàm mật độ xác suất $f(x) = 20x(1-x)^3$, $0 < x < 1$.

Giải: Chọn $g(x)$ là phân bố đều trong khoảng $(0,1)$ nên $g(x) = 1$, $0 < x < 1$, nên

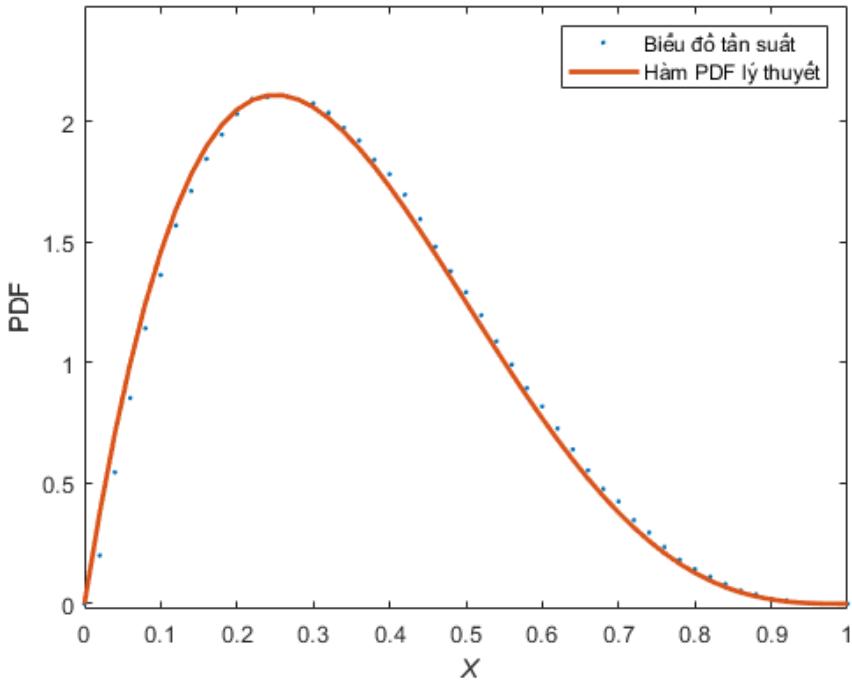
$$h(x) = \frac{f(x)}{g(x)} = 20x(1-x)^3, \quad 0 < x < 1. \quad (4.15)$$

Dễ dàng thấy rằng

$$h'(x) = 20[(1-x)^3 - 3x(1-x)^2] \quad (4.16)$$

dẫn đến $h'(x) = 0$ khi $x = \frac{1}{4}$. Khi đó, ta có $c = \left. \frac{f(x)}{g(x)} \right|_{x=\frac{1}{4}} = \frac{135}{64}$.

```
%  
N = 10^6;  
% Bước 1  
U1 = rand(1,N);  
% Bước 2  
U2 = rand(1,N);  
% Bước 3  
R = U1(U2 < 256/27.*U1.*(1-U1).^3);  
  
% Vẽ đồ thị tần suất  
xMin = 0;  
xMax = 1;  
  
NoB = 50;  
dx = (xMax - xMin)/NoB;  
  
x = xMin:dx:xMax;  
% Bước 4  
PDF_s = zeros(size(x));  
for idx = 2:length(x)  
    PDF_s(idx) = PDF_s(idx) + sum((R >= x(idx-1))&(R < x(idx)))/(dx*length(R));  
end  
  
PDF_t = 20.*x.*(1-x).^3;  
  
plot(x,PDF_s,'.',x,PDF_t,'-', 'linewidth', 2);  
xlabel('x');  
ylabel('PDF');  
legend('Biểu đồ tần suất', 'Hàm PDF lý thuyết');  
set(gcf, 'color', 'white');
```



Hình 4.6: Biểu đồ tần suất của biến ngẫu nhiên tạo ra từ giải thuật loại bỏ.

4.1.2 Kỳ vọng

Kỳ vọng của biến ngẫu nhiên (expectation) là một tham số thống kê quan trọng nhất của một biến ngẫu nhiên. Nếu X là một biến ngẫu nhiên rời rạc sẽ có những giá trị x_1, x_2, \dots , kỳ vọng của X được tính như sau:

$$E[X] = \sum_i x_i \Pr(X = x_i) \quad (4.17)$$

với $E[.]$ là toán tử kỳ vọng.

Nếu X là một biến ngẫu nhiên liên tục có hàm mật độ phân bố xác suất f , kỳ vọng của biến ngẫu nhiên X được tính như sau:

$$E[X] = \int_{-\infty}^{\infty} x f(x) dx. \quad (4.18)$$

Nếu X là một biến ngẫu nhiên rời rạc có hàm xác suất khói $p(x)$, kỳ vọng của $g(X)$ được tính như sau:

$$E[X] = \sum_i x_i p(x). \quad (4.19)$$

Khi X là biến ngẫu nhiên liên tục với hàm mật độ phân bố xác suất $f(x)$, ta có

$$E[g(x)] = \int_{-\infty}^{\infty} g(x) f(x) dx. \quad (4.20)$$

Nếu a và b là hằng số, ta có

$$E[aX + b] = aE[X] + b. \quad (4.21)$$

Nếu X_1 và X_2 là độc lập thống kê, ta có

$$E[a_1 X_1 + a_2 X_2] = a_1 E[X_1] + a_2 E[X_2]. \quad (4.22)$$

Matlab cung cấp cho chúng ta hàm mean() để tính kỳ vọng của biến ngẫu nhiên.

Cú pháp:

mean(X,'all'): tính kỳ vọng của mọi thành phần trong biến ngẫu nhiên X.

mean(X,DIM): tính kỳ vọng theo chiều DIM của biến ngẫu nhiên.

Ví dụ 4.8

Dùng hàm rand() để tạo 1000 biến ngẫu nhiên X phân bố đều trong khoảng [0,1]

- Tính kỳ vọng của X .
- Tính kỳ vọng và phương sai của $aX + b$ với $a = 2$ và $b = 3$.

Giải: Trong Ví dụ 4.8, chúng ta sử dụng hàm rand(), hàm mean() và hàm var().

```
N = 1000;  
%%  
X = rand(1,N);  
meanX = mean(X)  
varX = var(X)  
  
%%  
a = 2;  
b = 3;  
Y = 2*X + b;  
meanY = mean(Y)  
varY = var(Y)
```

4.1.3 Phương sai

Phương sai là thước đo độ lệch của những giá trị của X so với giá trị trung bình. Nghĩa là nếu cùng giá trị trung bình mà phương sai lớn nghĩa là biến ngẫu nhiên có mức dao động lớn và ngược lại. Nếu phương sai bằng không thì ta có thể kết luận rằng biến ngẫu nhiên là hằng số.

Nếu X là biến ngẫu nhiên có kỳ vọng μ , phương sai của X được tính như sau:

$$\text{Var}(X) = E[(X - \mu)^2]. \quad (4.23)$$

Một cách tính phương sai khác là:

$$\text{Var}(X) = E[X^2] - \mu^2. \quad (4.24)$$

Phương sai của biến ngẫu nhiên $aX + b$ là:

$$\text{Var}(aX + b) = a^2 \text{Var}(X). \quad (4.25)$$

Matlab cung cấp hàm var() và std() để tính phương sai và độ lệch chuẩn của biến ngẫu nhiên.

Cú pháp:

$$V = \text{var}(A)$$

$$V = \text{var}(A, w)$$

$$V = \text{var}(A, w, 'all')$$

$$V = \text{var}(A, w, \text{dim})$$

với A là ma trận biến ngẫu nhiên, w là vector trọng số khi tính phương sai và giá trị mặc định của w là 0 và 1, dim là chiều chúng ta muốn tính phương sai.

Ví dụ 4.9

Viết chương trình Matlab kiểm chứng công thức $\text{Var}(aX + b) = a^2 \text{Var}(X)$.

Giải: Để kiểm chứng, chúng ta lần lượt tính giá trị của vế trái và vế phải, sau đó so sánh với nhau.

```
N = 10^3;  
  
X = rand(1,N);  
Y = a*X + b;  
  
a = 2;  
b = 3;  
  
varX = var(X);  
  
LHS = a^2.*varX  
RHS = var(Y)
```

Nhận xét: Kết quả mô phỏng trả về là vế phải và vế trái bằng nhau và đều bằng 0.335.

Ví dụ 4.10

Cho X là biến ngẫu nhiên nhận hai giá trị 1 và -1 với xác suất tương ứng là $1/2$ và $1/2$.

- Tính kỳ vọng và phương sai của X theo lý thuyết.
- Mô phỏng và ước lượng kỳ vọng và phương sai của X .
- So sánh và nhận xét kết quả câu a) và b).

Giải: Kỳ vọng của X là:

$$\mu_X = \frac{1}{2} \times 1 + \frac{1}{2} \times (-1) = 0. \quad (4.26)$$

Phương sai của X là:

$$\begin{aligned}\sigma_x^2 &= E[X^2] - \mu^2 \\ &= \frac{1}{2} \times (1)^2 + \frac{1}{2} \times (-1)^2 . \\ &= 1\end{aligned}\tag{4.27}$$

Mã nguồn Matlab tính kỳ vọng và phương sai của X như sau:

```
N = 10^3;
X = randi([0 1],1,N);
X = 2*X - 1;
```

```
meanX = mean(X)
```

```
varX = var(X)
```

```
meanX =
```

```
-0.0040
```

```
varX =
```

```
1.0010
```

Chúng ta có thể thấy rằng kết quả tính toán và kết quả mô phỏng của kỳ vọng và phương sai là gần nhau.

4.1.4 Hiệp phương sai

Hiệp phương sai là độ đo sự biến thiên cùng nhau của hai biến ngẫu nhiên. Cho hai biến ngẫu nhiên X và Y , hiệp phương sai của X và Y là:

$$\text{cov}(X, Y) = E[(X - \mu_x)(Y - \mu_y)]\tag{4.28}$$

với $\mu_x = E[X]$ và $\mu_y = E[Y]$.

Cách thứ hai để tính hiệp phương sai cho X và Y là:

$$\text{cov}(X, Y) = E[XY] - E[X]E[Y].\tag{4.29}$$

Cách thứ ba để tính hiệp phương sai cho X và Y là:

$$\text{cov}(X, Y) = \frac{\text{Var}(X + Y) - \text{Var}(X) - \text{Var}(Y)}{2}.\tag{4.30}$$

Nếu X và Y độc lập với nhau, ta dễ dàng nhận thấy hai điều

$$\text{cov}(X, Y) = 0\tag{4.31}$$

và

$$\text{Var}(X + Y) = \text{Var}(X) + \text{Var}(Y).\tag{4.32}$$

Hệ số tương quan giữa hai biến ngẫu nhiên X và Y được định nghĩa như sau:

$$\text{Corr}(X, Y) = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}(X)\text{Var}(Y)}}. \quad (4.33)$$

Để tính hiệp phương sai và hệ số tương quan trong Matlab, chúng ta dùng hàm **cov()** và **corrcoef()**.

Cú pháp:

cov(X,Y): tính hiệp phương sai

R = corrcoef(X,Y)

với X và Y là ma trận biến ngẫu nhiên mà chúng ta muốn tính.

Ví dụ 4.11

Tạo hai biến ngẫu nhiên X và Y độc lập có phân bố chuẩn dùng hàm **randn()** có trung bình bằng 0 và phương sai là σ^2 . Tạo biến ngẫu nhiên $Z = \rho X + \sqrt{1-\rho^2}Y$

- Tìm hệ số tương quan của Z và X .
- So sánh hệ số tương quan tìm được với ρ .

Giải: Chúng ta chọn giá trị $\rho = 0.5$.

```
N = 10^6;  
rho = 0.5;  
  
X = randn(1,N);  
N = randn(1,N);  
  
Y = rho*X + sqrt(1-rho^2)*N;  
  
mu_X = mean(X)  
mu_Y = mean(Y)  
  
var_X = mean((X - mu_X).^2)  
var_Y = mean((Y - mu_Y).^2)  
  
% cov_XY = mean((X - mu_X).*(Y - mu_Y))  
cov_XY = mean(X.*Y) - mean(X).*mean(Y)  
  
corr = cov_XY./sqrt(var_X.*var_Y)
```

mu_X =

7.6818e-04

mu_Y =

-0.0012

```

var_X =
1.0008

var_Y =
1.0008

cov_XY =
0.5010

corr =
0.5007

```

4.1.5 Bất đẳng thức Markov

Nếu X là biến ngẫu nhiên chỉ lấy giá trị dương, với a bất kỳ, ta có:

$$\Pr(X \geq a) \leq \frac{E[X]}{a}. \quad (4.34)$$

Ta dễ dàng chứng minh bất đẳng thức (4.34) bằng cách định nghĩa biến ngẫu nhiên Y như sau:

$$Y = \begin{cases} a, & X \geq a \\ 0, & X < a \end{cases}. \quad (4.35)$$

Khi $X \geq 0$, ta có thể viết:

$$X \geq Y. \quad (4.36)$$

Lấy kỳ vọng hai vế của (4.36), ta có:

$$E[X] \geq E[Y] = a \Pr(X \geq a). \quad (4.37)$$

Ví dụ 4.12

Tạo biến ngẫu nhiên X có chiều dài 1000, kiểm chứng hai vế của bất đẳng thức Markov.

Giải: Chúng ta lần lượt tính giá trị của vế trái và vế phải sau đó so sánh.

```

N = 10^3;
X = rand(1,N);
a = 0.3;
LHS = sum(X >= a)/N
RHS = mean(X)/a
I = (LHS <= RHS)

```

4.1.6 Bất đẳng thức Chebyshev

Nếu X là biến ngẫu nhiên có trung bình μ và phương sai σ^2 , cho giá trị k bất kỳ, ta có:

$$\Pr(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}. \quad (4.38)$$

Ta dễ dàng chứng minh bất đẳng thức Chebyshev bằng cách sử dụng bất đẳng thức Markov, cụ thể:

$$\Pr\left[\frac{(X - \mu)^2}{\sigma^2} \geq k^2\right] \leq \frac{E\left[\frac{(X - \mu)^2}{\sigma^2}\right]}{k^2} \quad (4.39)$$

mà $E\left[\frac{(X - \mu)^2}{\sigma^2}\right] = 1$, nên ta dễ dàng có được điều phải chứng minh.

Ví dụ 4.13

Viết trình Matlab kiểm chứng bất đẳng thức Chebyshev.

Giải: Mã nguồn Matlab như sau. Kết quả trả về I sẽ là 1.

```
% BDT Chebyshev
%% VD 4.6
N = 10^3;
mu = 2;
sigma = 1;
X = mu + sigma.*rand(1,N);
k = 0.3;

LHS = sum(abs(X - mu) >= k*sigma)/N
RHS = 1/k^2

I = (LHS <= RHS)
```

4.1.7 Định luật số lớn

Cho X_1, X_2, \dots, X_n là các biến ngẫu nhiên độc lập và phân bố giống nhau với kỳ vọng μ .

Cho giá trị $\epsilon > 0$, ta có:

$$\Pr\left[\left|\frac{X_1 + \dots + X_n}{n} - \mu\right| \geq \epsilon\right] \rightarrow 0 \text{ khi } n \rightarrow \infty \quad (4.40)$$

Ví dụ 4.14

Viết chương trình bằng Matlab kiểm chứng định luật số lớn.

Giải: Chúng ta chỉ cần tính vế trái, khi vế phải là 0.

```
N = 10.^[1 2 3 4 5];
```

```
mu = 2;
sigma = 1;
epsilon = 0.1;
```

```

for idx = 1:length(N)
    X = mu + sigma.*rand(1,N(idx));
    LHS(idx)=sum(abs(mean(X) - mu) >= epsilon)/N(idx);
end
LHS

```

Ta nhận thấy khi tăng giá trị N thì xác suất càng tiến về 0.

4.1.8 Định lý giới hạn trung tâm

Giả sử X_1, X_2, \dots, X_n là tập hợp các biến ngẫu nhiên độc lập có cùng phân bố và cùng kỳ vọng μ và phương sai σ^2 hữu hạn. Gọi

$$Z_n = \frac{\left(\sum_{i=1}^n X_i \right) - n\mu}{\sigma\sqrt{n}} \quad (4.41)$$

là tổng chuẩn hóa của X_1, X_2, \dots, X_n . Khi đó Z_n sẽ tiến tới phân bố chuẩn có trung bình bằng không và phương sai bằng 1, $Z_n \sim \mathcal{N}(0,1)$, khi n tiến tới vô cùng.

Ví dụ 4.15

Kiểm chứng lại định lý giới hạn trung tâm với biến có phân bố mũ.

Giải: Đây là một ví dụ quan trọng cho phép người đọc cảm nhận được định lý giới hạn trung tâm. Việc kiểm chứng bao gồm hai bước chính là vẽ biểu đồ tần suất và vẽ phân bố Gauss có trung bình và phương sai phù hợp với biểu đồ tần suất.

Để vẽ biểu đồ tần suất của M biến ngẫu nhiên phân bố hàm mũ, chúng ta cần tạo ra ma trận 2 chiều với $M \times N$ trong đó N là số lượng mẫu cần thiết để vẽ biểu đồ tần suất. Chúng ta cần có N đủ lớn để biểu đồ tần suất đạt được mức hội tụ cần thiết. Cần lưu ý rằng với mỗi loại biến ngẫu nhiên thì giá trị kỳ vọng và giá trị phương sai có mối liên hệ toán học, chứ không phải tự gán giá trị bất kỳ. Trong Ví dụ 4.15, chúng ta xem xét biến ngẫu nhiên hàm mũ, do đó phương sai sẽ là bình phương của giá trị kỳ vọng.

Chúng ta có thể tăng giá trị M để kiểm chứng định lý giới hạn trung tâm, theo lý thuyết thì M phải tiến tới vô cùng thì biến Z_n sẽ tiến tới phân bố Gaussian.

Nguyên tắc vẽ biểu đồ tần suất chúng ta có thể tham khảo lại mục biểu đồ tần suất ở chương trước. Tuy nhiên cần nhắc lại là biểu đồ tần suất phụ thuộc vào số lượng phân đoạn dữ liệu, trong Matlab gọi là Bin.

```

% Chiều dài chuỗi
N = 10^4;
% Số lượng điểm để vẽ đồ thị tần suất
NoB = 30;
% Số lượng biến để cộng vào

```

```

M = 20;
% Giá trị kỳ vọng
muX = 2;
% Giá trị phương sai
varX = muX^2;

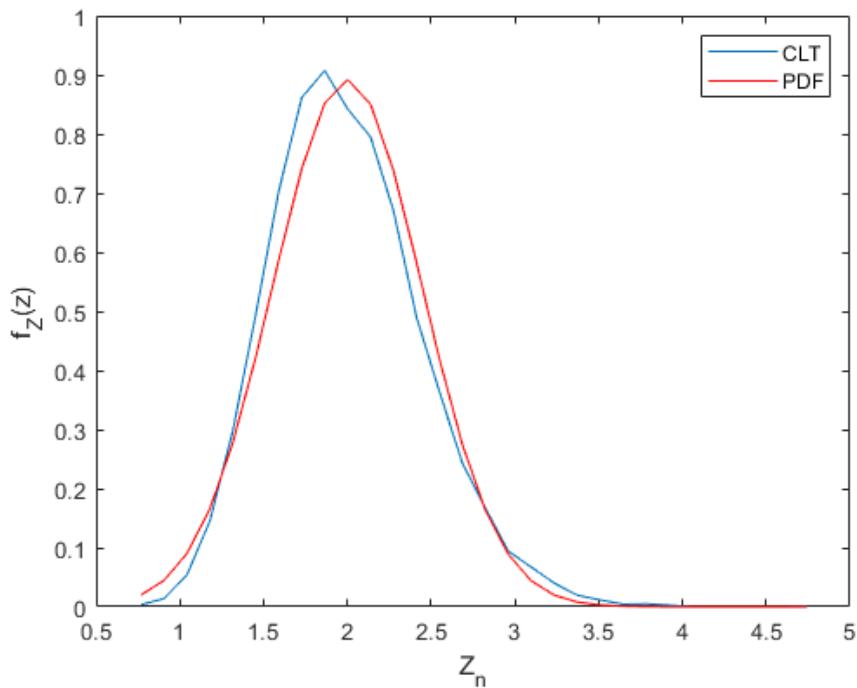
% Tổng của M biến ngẫu nhiên phân bố mũ có trung bình muX
x = sum(exprnd(muX,M,N),1)/M;

% Biểu đồ tần suất
h = histogram(x,NoB);
pdf_e = h.Values./(N*h.BinWidth);
axis_x = (h.BinEdges(1:end-1) + h.BinEdges(2:end))/2;

% Biến ngẫu nhiên phân bố Gauss
mean_g = muX;
var_g = varX/M;
pdf_g = pdf('norm',axis_x,mean_g,sqrt(var_g));

% Vẽ biểu đồ tần suất và hàm PDF của phân bố chuẩn
plot(axis_x,pdf_e,axis_x,pdf_g,'r');
xlabel('Z_n');
ylabel('f_Z(z)');
legend('CLT','PDF')

```



Hình 4.7: Kiểm chứng định lý giới hạn trung tâm với biến phân bố hàm mũ.

Ví dụ 4.16

Tương tự như Ví dụ 4.15, nhưng thực hiện với biến phân bố đều từ -2 đến 2.

Giải: Với phân bố đều chúng ta dùng hàm rand().

```

%
clear all
% Số lượng mẫu
N = 10^4;
% Số lượng phân đoạn trên trục x
NoB = 30;
% Số lượng biến phân bố đều
M = 10;

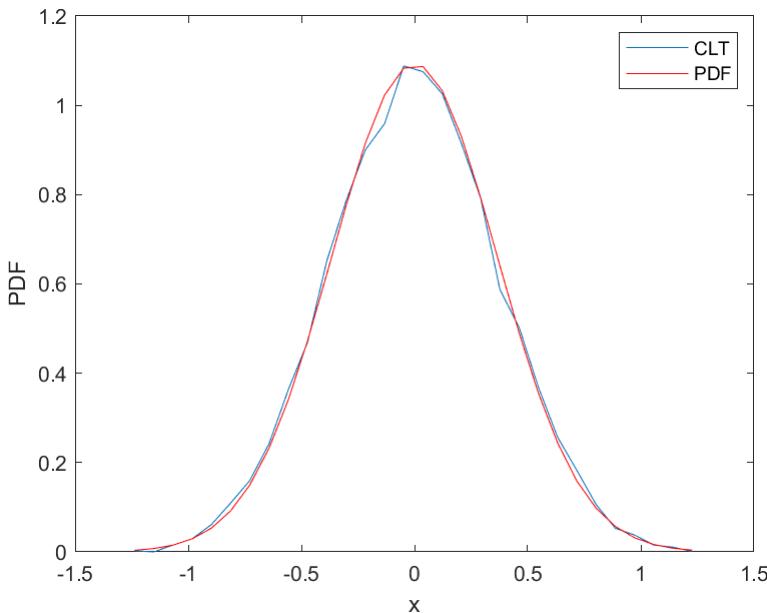
% Tham số của biến phân bố đều
a = -2;
b = 2;
muX = (b + a)/2;
varX = (b - a)^2/12;
% Tổng của M biến ngẫu nhiên
x = sum(a + (b-a).*rand(M,N),1)/M;

% Biểu đồ tần suất
h = histogram(x,NoB);
pdf_e = h.Values./(N*h.BinWidth);
axis_x = (h.BinEdges(1:end-1) + h.BinEdges(2:end))/2;

% Hàm PDF của phân bố Gauss
mean_g = muX;
var_g = varX/M;
pdf_g = pdf('norm',axis_x,mean_g,sqrt(var_g));

% Vẽ biểu đồ tần suất
plot(axis_x,pdf_e, axis_x,pdf_g,'r');
xlabel('x');
ylabel('PDF');
legend('CLT','PDF')

```



Hình 4.8 Kiểm chứng định lý giới hạn trung tâm với biến phân bố đều từ -2 đến 2.

Ví dụ 4.17

Sử dụng định lý giới hạn trung tâm mô phỏng đường truyền Rayleigh fading.

Giải: Chúng ta biết rằng kênh truyền Rayleigh tồn tại giữa máy phát và máy thu khi không có đường truyền thẳng, máy thu nhận nhiều tín hiệu độc lập có phân bố giống nhau từ máy phát do các hiện tượng phản xạ, tán xạ và nhiễu xạ. Theo công thức 3.13 của [4], chúng ta có thành phần đồng pha và vuông pha của tín hiệu nhận tại máy thu là:

$$r_I(t) = \sum_{n=1}^{N(t)} \alpha_n(t) \cos \phi_n(t) \quad (4.42)$$

$$r_Q(t) = \sum_{n=1}^{N(t)} \alpha_n(t) \sin \phi_n(t) \quad (4.43)$$

với $N(t)$ là số lượng tia tín hiệu đến máy thu, $\alpha_n(t)$ là biên độ có phân bố Rayleigh và $\phi(t)$ là pha có phân bố đều trong khoảng $[0, 2\pi]$. Theo định lý giới hạn trung tâm, nếu $N(t)$ đủ lớn, thì r_I và r_Q là biến ngẫu nhiên Gauss có trung bình bằng không và phương sai σ^2 .

Chương trình Matlab mô phỏng như sau.

```
clear all

% Số lượng mẫu
N = 10^5;
M = 20;

% Tham số biến ngẫu nhiên: Scale parameters
sigma = 2;
mu_alpha = sigma*sqrt(pi/2);
var_alpha = (4-pi)/2*sigma^2;

alpha = raylrnd(sigma,M,N);
phi = 2*pi*rand(M,N);

rI = sum(alpha.*cos(phi),1)/M;
rQ = sum(alpha.*sin(phi),1)/M;

mu_rI = mean(rI);
var_rI = var(rI);

%
% Số lượng phân đoạn trên trục x
NoB = 20;
hI = histogram(rI,NoB);
pdf_rI = hI.Values./(N*hI.BinWidth);
hQ = histogram(rQ,NoB);
pdf_rQ = hQ.Values./(N*hQ.BinWidth);

axis_x = (hQ.BinEdges(1:end-1) + hQ.BinEdges(2:end))/2;
% Hàm PDF của phân bố Gauss
```

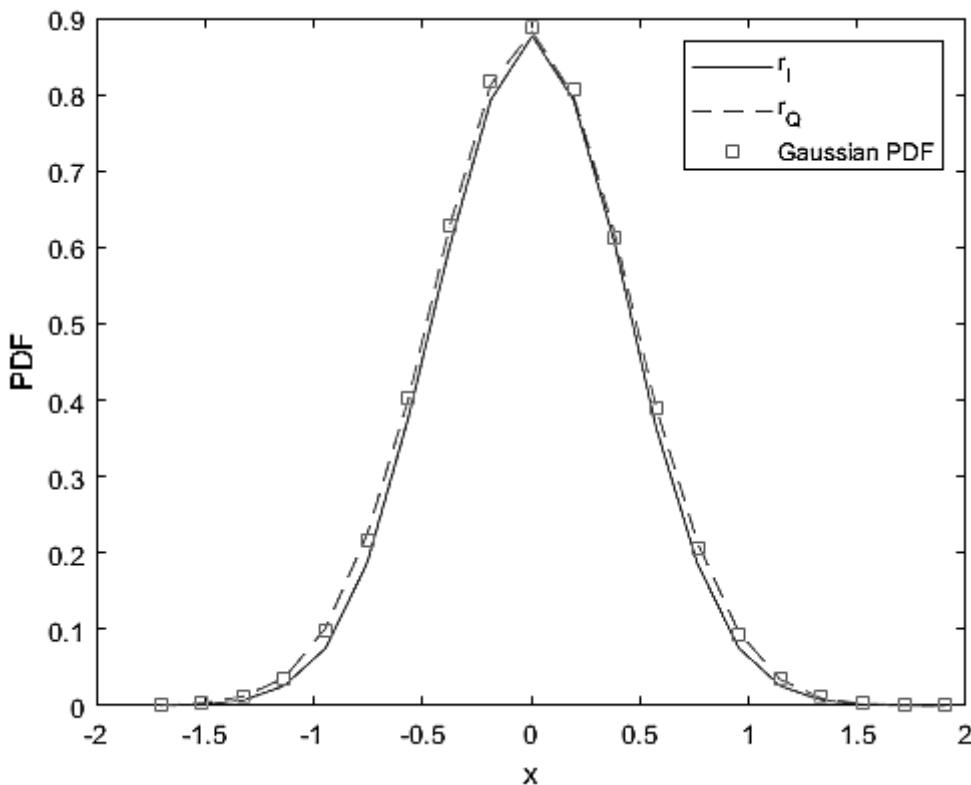
```

mean_g = mu_rI;
var_g = var_rI;
pdf_g = pdf('norm',axis_x,mean_g,sqrt(var_g));

% Vẽ đồ thị
plot(axis_x,pdf_rI,'-',axis_x,pdf_rQ,'--',axis_x,pdf_g,'s');
xlabel('x');
ylabel('PDF');
legend('r_I','r_Q','Gaussian PDF');
set(gcf,'color','white');

```

Kết quả mô phỏng của đoạn mã là như Hình 4.9.



Hình 4.9: Hàm đồ thị tần suất của thành phần đồng pha và vuông pha so sánh với hàm PDF của phân bố Gauss.

4.1.9 Biến ngẫu nhiên rời rạc

4.1.9.1 Phân bố nhị thức (Binomial distribution)

Phân phối nhị thức là một phân phối xác suất rời rạc với hai tham số cho trước là n và p , Phân phối nhị thức thường được dùng cho các biến biến ngẫu nhiên thể hiện số lượng lượt thử thành công trong n lượt thử độc lập. Gọi X là biến ngẫu nhiên có phân phối nhị thức, hàm khôi xác suất có dạng như sau:

$$\Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k} \quad (4.44)$$

với $k = 1, \dots, n$.

Hàm phân phối tích lũy là:

$$F(k, n, p) = \Pr(X \leq k) = \sum_{i=0}^k \binom{n}{i} p^i (1-p)^{i-k}. \quad (4.45)$$

Giá trị kỳ vọng và phương sai của X là:

$$E(X) = p, \quad (4.46)$$

$$\text{Var}(X) = p(1-p). \quad (4.47)$$

Bảng 4.1: Các hàm liên quan đến phân bố nhị phân.

TT	Hàm	Mô tả hàm
1	binornd()	Tạo ra chuỗi ngẫu nhiên theo phân bố nhị phân
2	binoinv()	Hàm ngược của hàm phân bố xác suất tích lũy nhị phân
3	binopdf()	Hàm mật độ phân bố xác suất nhị phân
4	binocdf()	Hàm phân bố xác suất tích lũy nhị phân
5	binostat()	Trả về giá trị kỳ vọng và phương sai của phân bố nhị phân
6	binofit()	Trả về giá trị ước lượng của kỳ vọng và phương sai của một ma trận ngẫu nhiên có phân bố nhị phân

Ví dụ 4.18

Vẽ hàm CDF và PDF của phân bố nhị thức.

Giải: Tương tự như trên, nhưng ở Ví dụ 4.18, chúng ta chỉ sử dụng hàm binornd() và thực hiện vẽ hàm PDF và CDF.

```

clear all
% Số lượng mẫu thử
N = 10^4;
% Tham số p
p = 0.3;
%
nMax = 10;

pdf_t = zeros(1,nMax+1);
% Lý thuyết
n = 0:nMax;
for idx = 1:length(n)
    pdf_t(idx) = nchoosek(nMax,n(idx))*p^n*(1-p)^(nMax-n(idx));
end
cdf_t = cumsum(pdf_t);

% Mô phỏng
x = binornd(nMax,p,1,N);

```

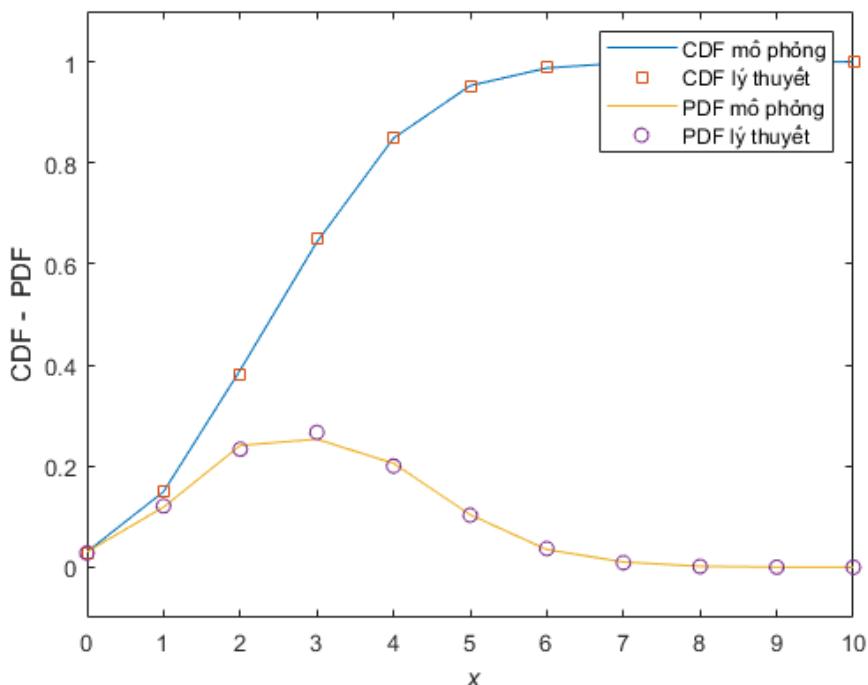
```

cdf_e = zeros(1,nMax+1);
for idx = 1:length(n)
    cdf_e(idx) = cdf_e(idx) + sum(x<=n(idx))/N;
end
pdf_e = [cdf_e(1) diff(cdf_e)];

% Vẽ kết quả
axis_x = n;
plot(axis_x,cdf_e,'-',axis_x,cdf_t,'s',axis_x,pdf_e,'-',axis_x,pdf_t,'o');

xlabel('ít x');
ylabel('CDF - PDF');
legend('CDF mô phỏng','CDF lý thuyết','PDF mô phỏng','PDF lý thuyết');
axis([0 nMax -0.1 1.1]);
set(gcf,'color','white');

```



Hình 4.10 Hàm PDF và CDF của phân bố nhị phân.

Ví dụ 4.19

A và B chơi trận chung kết tennis. Luật chơi là nếu ai thắng trước 3 set trước thì thắng cả trận. Giả sử A chơi yếu hơn và được đánh giá xác suất thắng mỗi set là 0.4. Hãy mô phỏng tính toán khả năng thắng trận chung kết của A. So sánh với lý thuyết.

Giải: Theo luật chơi thì người nào thắng ba trận sẽ thắng, do đó số trận chơi ít nhất là 3 và số trận chơi nhiều nhất là 5. Đối với A, có ba trường hợp xảy ra:

- Chơi 3 trận: A toàn thắng 3 trận;
- Chơi 4 trận: A thắng 2 trận trong 3 trận đầu và 1 trận cuối;

- Choi 5 trận: A thắng 2 trận trong 4 trận đầu và 1 trận cuối.

Theo lý thuyết xác suất, xác suất A thắng sẽ là tổng của ba trường hợp trên:

$$\binom{3}{3}0.4^3(1-0.4)^0 + \binom{4}{3}0.4^3(1-0.4)^1 + \binom{5}{3}0.4^3(1-0.4)^2 = 0.448. \quad (4.48)$$

Chương trình mô phỏng Matlab của Ví dụ 4.19 trong trường hợp trên là

```
% Số lượng mẫu thử
N = 10^6;
% Xác suất A thắng
p = 0.4;
% Số trận cần phải thắng trước
NoW = 3;
% Khởi tạo biến
Awin_t=0;
Awin_s=0;
for n = 3:5
    % Lý thuyết
    Awin_t = Awin_t + nchoosek(n,NoW)*p^NoW*(1-p)^(n-NoW);
    % Mô phỏng
    Awin_s = Awin_s + sum(binornd(n,p,1,N)==NoW)/N;
end

Awin_t
Awin_s
abs(Awin_t-Awin_s)/Awin_t
```

Chương trình Malab trên trả về kết quả A thắng B lần lượt là 0.448 cho kết quả lý thuyết và 0.4479 cho kết quả mô phỏng. Sai số tương đối giữa kết quả lý thuyết và mô phỏng là 0.03%, là chấp nhận được cho 1 triệu mẫu.

4.1.9.2 Phân bố hình học (Geometric Random)

Xem xét n phép thử độc lập với xác suất thành công của mỗi lần thử là p với $0 \leq p \leq 1$. Gọi X là số lần thử thành công, nghĩa là:

$$\Pr(X = n) = p(1-p)^{n-1}, \quad \forall n \in \mathbb{Y} \quad (4.49)$$

nghĩa là $n-1$ lần thử trước là thất bại.

Giá trị trung bình của X là:

$$E(X) = \sum_{n=1}^{\infty} np(1-p)^{n-1} = \frac{1}{p}. \quad (4.50)$$

Phương sai của X là:

$$\text{Var}(X) = \frac{1-p}{p^2}. \quad (4.51)$$

Bảng 4.2: Các hàm liên quan đến phân bố hình học.

TT	Hàm	Mô tả hàm
1	geornd()	Tạo ra chuỗi ngẫu nhiên theo phân bố hình học
2	geoinv()	Hàm ngược của hàm phân bố xác suất tích lũy hình học
3	geopdf()	Hàm mật độ phân bố xác suất hình học
4	geocdf()	Hàm phân bố xác suất tích lũy hình học
5	geostat()	Trả về giá trị kỳ vọng và phương sai của phân bố hình học
6	geofit()	Trả về giá trị ước lượng của kỳ vọng và phương sai của một ma trận ngẫu nhiên có phân bố hình học

Ví dụ 4.20

Sử dụng mô phỏng kiểm chứng xác suất mà sau n lần thay, ta mới có mặt ngửa với xác suất thay mặt ngửa và sấp là $\frac{1}{2}$. Kiểm chứng với phân tích lý thuyết.

Giải: Chúng ta lưu ý sử dụng hàm all() để kiểm chứng n lần thay liên tiếp nhau.

```
%  
n = 3;  
p = 1/2;  
  
% Lý thuyết  
Pr_t = p*(1-p)^(n-1)  
  
% Mô phỏng  
% M = số lần thử  
M = 10^5;  
x = randi([0 1],1,M);  
  
cnt = 0;  
y = [zeros(1,n-1) 1];  
for idx = 1:M-5  
    cnt = cnt + all(x(idx:idx+n-1)==y);  
end  
Pr_s = cnt/M
```

4.1.9.3 Phân bố Poisson

Phân bố Poisson là một phân bố rời rạc, được dùng cho các biến ngẫu nhiên mô hình hóa số lần xảy ra của một sự kiện trong một khoảng thời gian nhất định, đặc trưng bởi giá trị trung bình λ . Phân bố Poisson khác với các phân phối xác suất rời rạc khác là thông tin xác suất cho biết không phải là xác suất để một sự kiện (event) xảy ra (thành công) trong một lần thử. Tùy theo bài toán xem xét, phân phối Poisson có thể dùng cho các đơn vị khác khoảng thời gian, ví dụ như như: khoảng cách, diện tích, hay thể tích.

Một biến ngẫu nhiên X được gọi là có phân bố Poisson với tham số λ nếu X có giá trị nguyên không âm và

$$\Pr(X = k) = \frac{\lambda^k}{k!} e^{-\lambda}. \quad (4.52)$$

Giá trị kỳ vọng và phương sai của X là

$$E[X] = \text{Var}[X] = \lambda. \quad (4.53)$$

Matlab hỗ trợ các hàm liên quan đến phân bố Poisson như Bảng 4.3.

Bảng 4.3: Các hàm liên quan đến phân bố Poisson.

TT	Hàm	Mô tả hàm
1	poissrnd()	Tạo ra chuỗi ngẫu nhiên theo phân bố Poisson
2	poisinv()	Hàm ngược của hàm phân bố xác suất tích lũy Poisson
3	poispdf()	Hàm mật độ phân bố xác suất Poisson
4	poisscdf()	Hàm phân bố xác suất tích lũy Poisson
5	poisstat()	Trả về giá trị kỳ vọng và phương sai của phân bố Poisson
6	poissfit()	Trả về giá trị ước lượng của kỳ vọng và phương sai của một ma trận ngẫu nhiên có phân bố Poisson

Ví dụ 4.21

Tổng đài hỗ trợ khách hàng thông kê có trung bình 5 cuộc gọi trong 1 phút. Gọi X là số cuộc gọi đến tổng đài, hãy mô phỏng quá trình để vẽ ra hàm CDF và PDF và so sánh với lý thuyết.

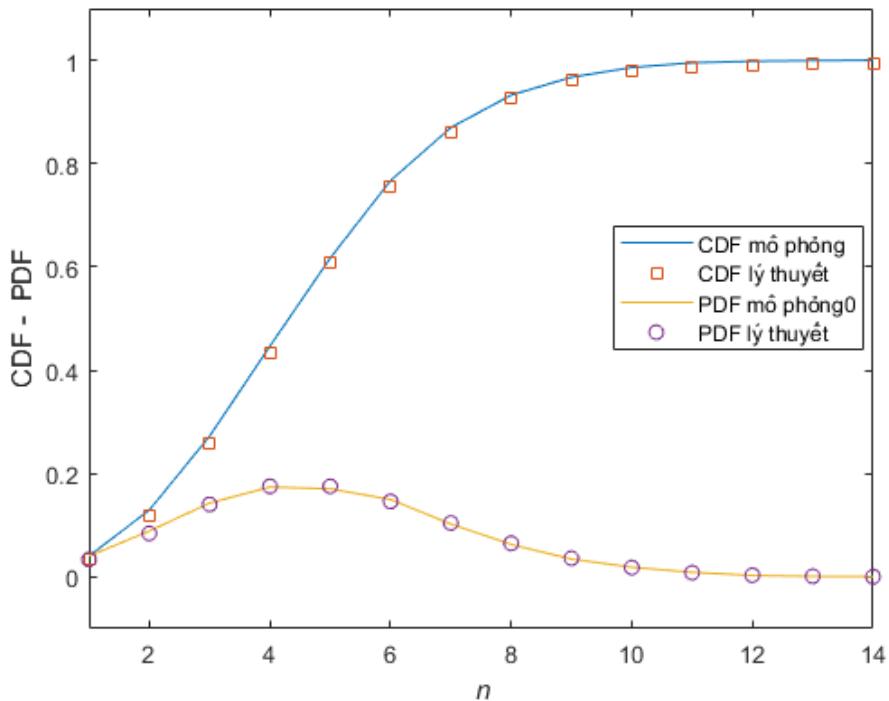
Giai: Số cuộc gọi trong 1 phút là 5, nghĩa là $\lambda = 5$.

```
%  
N = 10^4;  
lambda = 5;  
% Tạo biến ngẫu nhiên với tham số lambda  
x = poissrnd(lambda,1,10^4);  
  
% Khởi tạo biến ngẫu nhiên  
NoB = max(x);  
cdf_e=zeros(1,NoB);  
pdf_t=zeros(1,NoB);  
  
for idx = 1:NoB  
    cdf_e(idx) = cdf_e(idx) + sum(x <= idx);  
    pdf_t(idx) = exp(-lambda).*lambda.^idx./factorial(idx);  
end  
cdf_e = cdf_e./N;  
% Sử dụng hàm diff() để thực hiện đạo hàm  
pdf_e = [cdf_e(1) diff(cdf_e)];  
% Sử dụng hàm cumsum() để thực hiện tích phân  
cdf_t = cumsum(pdf_t);  
  
% Vẽ hai hàm CDF  
axis_x = 1:NoB;  
plot(axis_x,cdf_e,'-',axis_x,cdf_t,'s',axis_x,pdf_e,'-',axis_x,pdf_t,'o');  
xlabel('t n');  
ylabel('CDF - PDF');
```

```

legend('CDF mô phỏng','CDF lý thuyết','PDF mô phỏng0','PDF lý thuyết');
axis([1 NoB -0.1 1.1]);
set(gcf,'color','white');

```



Hình 4.11: Đồ thị kết quả của Ví dụ 4.21.

4.1.10 Biến ngẫu nhiên liên tục

4.1.10.1 Phân bố đều

Một biến ngẫu nhiên X có phân bố đều trong khoảng $[a,b]$ với $a < b$, nếu hàm mật độ phân bố xác suất của X có dạng như sau:

$$f(x) = \begin{cases} \frac{1}{b-a}, & a < b \\ 0, & \text{khác} \end{cases}. \quad (4.54)$$

Hàm phân bố xác suất tích lũy của X trong khoảng $a \leq x \leq b$ là:

$$F(x) = \Pr(X \leq x) = \int_0^x f(x) dx = \frac{x-a}{b-a}. \quad (4.55)$$

Giá trị kỳ vọng và phương sai của biến ngẫu nhiên X là:

$$E[X] = \frac{1}{b-a} \int_a^b x dx = \frac{b^2 - a^2}{2(b-a)} = \frac{b-a}{2}, \quad (4.56)$$

$$E[X^2] = \frac{1}{b-a} \int_a^b x^2 dx = \frac{b^3 - a^3}{2(b-a)} = \frac{a^2 + b^2 + ab}{3}, \quad (4.57)$$

$$Var[X] = \frac{1}{3}(a^2 + b^2 + ab) - \frac{1}{4}(a^2 + b^2 + 2ab) = \frac{(b-a)^2}{12}. \quad (4.58)$$

Để tạo biến ngẫu nhiên phân bố đều trong khoảng từ 0 đến 1 trong Matlab, chúng ta dùng hàm **rand()** có cú pháp là như sau:

Cú pháp:

`rand(M,N)`

với M là số dòng và N là số cột. Muốn tạo ra biến phân bố đều trong khoảng từ a đến b, chúng ta sử dụng biến đổi sau:

Cú pháp:

`r = a + (b-a).*rand(M,N);`

Muốn tạo ra biến ngẫu nhiên phân bố đều rời rạc từ 0 đến IMAX với M dòng và N cột, ta dùng hàm **randi()** với cú pháp như sau:

Cú pháp:

`I = randi(IMAX,M,N);`

Trong một số trường hợp, chúng ta có thể tự tạo biến ngẫu nhiên phân bố đều theo giải thuật đã trình bày ở mục 4.1.1.1.

Bảng 4.4: Các hàm liên quan đến phân bố đều.

TT	Hàm	Mô tả hàm
1	<code>unifrnd()</code>	Tạo ra chuỗi ngẫu nhiên theo phân bố đều
2	<code>unifinv()</code>	Hàm ngược của hàm phân bố xác suất tích lũy đều
3	<code>unifpdf()</code>	Hàm mật độ phân bố xác suất đều
4	<code>unifcdf()</code>	Hàm phân bố xác suất tích lũy hàm mũ
5	<code>unifstat()</code>	Trả về giá trị kỳ vọng và phương sai của phân bố đều
6	<code>uniffit()</code>	Trả về giá trị ước lượng của kỳ vọng và phương sai của một ma trận ngẫu nhiên có phân bố đều

Ví dụ 4.22

Sử dụng hàm `unifrnd()` tạo ra biến ngẫu nhiên có phân bố chuẩn từ 0 đến 1. Vẽ hàm PDF và CDF và so sánh với lý thuyết.

Giải: Chúng ta sử dụng hàm `unifrnd()`, `unifcdf()` và `unipdf()`.

```

N = 10^4;
a = 0;
b = 1;

% Mô phỏng
r = unifrnd(a,b,1,N);

dx = 0.1;
Xm = b;
x = 0:dx:Xm;

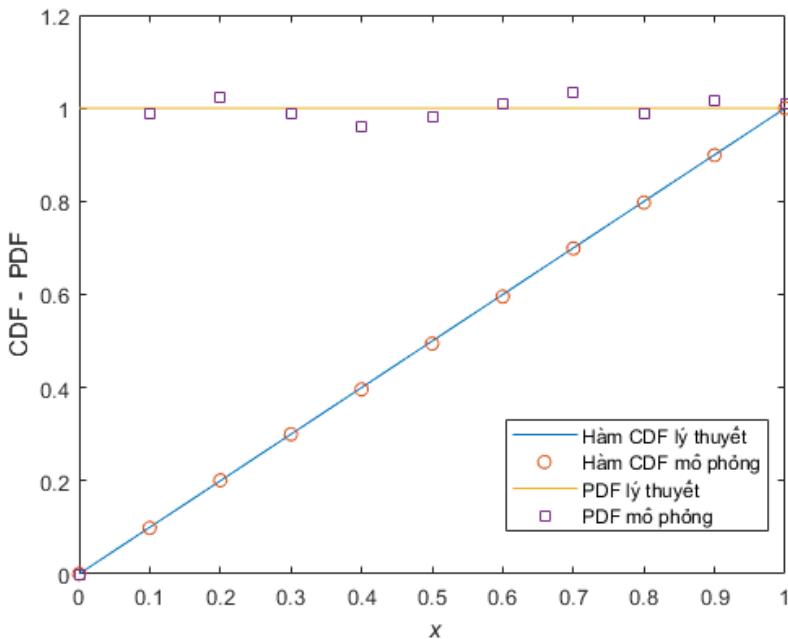
CDF_s = zeros(size(x));
for idx = 1:length(x)
    CDF_s(idx) = CDF_s(idx) + sum(r <= x(idx))/N;
end

PDF_s = zeros(size(x));
for idx = 2:length(x)
    PDF_s(idx) = PDF_s(idx) + sum((r >= x(idx-1))&(r < x(idx)))/(dx*N);
end

% Lý thuyết
PDF_t = unifpdf(x,a,b);
CDF_t = unifcdf(x,a,b);

% Vẽ hình so sánh
plot(x,CDF_t,x,CDF_s,'o',x,PDF_t,x,PDF_s,'s');
xlabel('ít x');
ylabel('CDF - PDF');
legend('Hàm CDF lý thuyết','Hàm CDF mô phỏng','PDF lý thuyết','PDF mô phỏng');
set(gcf,'color','white');

```



Hình 4.12: CDF và PDF của biến ngẫu nhiên phân bố đều của Ví dụ 4.22.

Ví dụ 4.23

Tạo 100 biến ngẫu nhiên phân bố đều trong khoảng từ 2 đến 10. Hãy

- Tính giá trị trung bình và phương sai theo công thức lý thuyết.
- Tính giá trị trung bình và phương sai theo mô phỏng và so sánh.

Giải: Chúng ta sử dụng công thức mean() và var() để tính giá trị kỳ vọng và phương sai khi mô phỏng.

```
%  
clc  
N = 10^2;  
a = 2;  
b = 10;  
X = a + (b - a).*rand(1,N);
```

```
% Lý thuyết  
mX_t = (a + b)/2  
vX_t = (b - a)^2/12
```

```
% Mô phỏng  
mX_s = mean(X)  
vX_s = var(X)
```

mX_t =

6

vX_t =

5.3333

mX_s =

5.9170

vX_s =

4.7715

Chúng ta có thể nhận thấy rằng kết quả mô phỏng hội tụ với kết quả lý thuyết.

Ví dụ 4.24: Sử dụng biến ngẫu nhiên để ước lượng tích phân.

Cho tích phân sau:

$$I = \int_0^1 g(x) dx \quad (4.59)$$

xây dựng phương pháp tính I và áp dụng cho $g(x) = x^2$.

Giải: Một trong những ứng dụng lớn nhất của số ngẫu nhiên là sử dụng để ước lượng tích phân, đặc biệt là khi tích phân không tồn tại dạng đóng – được biểu diễn dưới các tổ hợp hàm cơ sở. Phương pháp này gọi là tích phân theo phương pháp Monte Carlo.

Để tính I , chúng ta thấy rằng tích phân là từ 0 đến 1, nên ta tận dụng khái niệm tính kỳ vọng với biến ngẫu nhiên phân bố đều trong khoảng 0 đến 1. Cụ thể, ta viết lại I như sau:

$$\begin{aligned} I &= \int_0^1 g(x)dx \\ &= \int_0^1 g(U)f(U)dU = E[g(U)] \end{aligned} \tag{4.60}$$

với $f(U)$ là hàm PDF của biến ngẫu nhiên phân bố đều U từ 0 đến 1, hay $f(U)=1$ trong khoảng từ 0 đến 1 và bằng 0 ngoài khoảng.

Áp dụng định luật số lớn khi N tiến đến vô cùng, ta có:

$$I \rightarrow \sum_{n=1}^N \frac{g(U_n)}{N}. \tag{4.61}$$

Khi $g(x) = x^2$, ta có tích phân I như sau:

$$I = \left. \frac{x^3}{3} \right|_0^1 = \frac{1}{3}. \tag{4.62}$$

Giá trị chính xác của I là $1/3$. Ta có thể dùng giá trị chính xác để so sánh với giá trị ước lượng sau đây:

```
% Số lượng mẫu
N = 100;

% Tạo biến ngẫu nhiên phân bố đều
U = rand(1,N);
g = U.^2;

% Ước lượng I
I = sum(g)/N
```

Khi cho $N=100$ và $N=1000$, ta tìm được giá trị ước lượng của N là 0.3378 và 0.3414. Tuy nhiên, hai giá trị này sẽ thay đổi khi chúng ta lặp lại các mô phỏng.

Có một cách để ta cố định các kết quả sau mỗi lần mô phỏng là đặt lại trạng thái của hàm tạo biến ngẫu nhiên. Hay thử lại đoạn mã trên với hàm “**rng default**”.

```
% Số lượng mẫu
N = 1000;

% Tạo biến ngẫu nhiên phân bố đều
rng default
U = rand(1,N);
```

```

g = U.^2;
% Uớc lượng I
I = sum(g)/N

```

Khi có hàm “rng default”, kết quả của I luôn là .3191.

Ví dụ 4.25

Uớc lượng số π bằng kỹ thuật gieo xác suất.

Giải: Đây là một bài toán cổ điển để ước lượng số π . Để thực hiện phương pháp ước lượng này, chúng ta vẽ một hình vuông và một hình tròn lồng vào nhau có cùng tâm. Hình vuông có tọa độ 4 đỉnh lần lượt là $(1,0)$, $(1,1)$, $(-1,1)$ và $(-1,-1)$. Hình tròn có tâm tại tọa độ $(0,0)$ và bán kính là 1. Giả thử ta thấy ngẫu nhiên các hạt đậu có kích thước rất nhỏ vào hình vuông với (x, y) là tọa độ của hạt đậu. Phép thử là phù hợp nếu hạt đậu nằm trong hình vuông. Do ta thấy ngẫu nhiên, nên x và y sẽ có phân bố đều trong khoảng $(-1,1)$. Ta có xác suất hạt đậu nằm trong hình tròn là:

$$\begin{aligned} \Pr[(x, y) \in d] &= \Pr(x^2 + y^2 \leq 1) \\ &= \frac{S_d}{S_w} \end{aligned} \quad (4.63)$$

với S_d và S_w lần lượt là diện tích hình tròn và hình vuông. Ta dễ dàng có được

$$\begin{aligned} \Pr[(x, y) \in d] &= \Pr(x^2 + y^2 \leq 1) \\ &= \frac{\pi}{4}. \end{aligned} \quad (4.64)$$

Khi tăng số lượng lần thử, số lượng hạt đậu rơi trong hình tròn sẽ phủ kín diện tích hình tròn trong khi số lượng hạt đậu rơi trong hình vuông là bằng tổng số lần thử. Do đó, ta có tỷ lệ sau:

$$\Pr[(x, y) \in d] = \frac{\pi}{4} = \frac{N_d}{N}. \quad (4.65)$$

Do đó, ta có thể ước lượng số π như sau:

$$\pi = \frac{4N_d}{N}. \quad (4.66)$$

Mã nguồn Matlab thực hiện giải thuật tính số π như sau:

```

% Tổng số lần thử
N = 10^3;
% Tạo biến phân bố đều trong khoảng [-1,1]
rng default
x = -1 + 2*rand(1,N);
y = -1 + 2*rand(1,N);
% Tính số lượng điểm rơi trong hình tròn

```

```
Nc = sum(x.^2 + y.^2 <=1);
```

```
% Ước lượng số pi  
pi_est = 4*Nc./N
```

```
% Sai số tương đối  
abs(pi_est-pi)/pi
```

Chúng ta có kết quả sau khi tăng số lượng lần thử như Bảng 4.5 sau.

Bảng 4.5: Kết quả của 6 lần thử.

Lần thử	Số lần thay, N	π_{est}	Sai số tương đối
1	10	2.00	0.3634
2	100	3.08	0.0196
3	1.000	3.2200	0.0250
4	10.000	3.1348	0.0022
5	100.000	3.1408	2.3958e-04
6	1.000.000	3.1416	1.7617e-05

4.1.10.2 Phân bố chuẩn

Biến ngẫu nhiên X sẽ có phân bố chuẩn với trung bình μ và phương sai σ^2 khi có hàm mật độ phân bố xác suất như sau:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, -\infty < x < \infty. \quad (4.67)$$

Ta cũng dễ dàng thấy rằng nếu biến ngẫu nhiên X có phân bố chuẩn với trung bình μ và phương sai σ^2 thì biến ngẫu nhiên

$$Z = \frac{X - \mu}{\sigma} \quad (4.68)$$

sẽ có phân bố chuẩn với trung bình bằng 0 và phương sai bằng 1.

Gọi Φ là hàm phân bố xác suất tích lũy của X , ta có:

$$\begin{aligned} F_X(x) &= \Pr(X \leq x) \\ &= \Pr\left(\frac{X - \mu}{\sigma} \leq \frac{x - \mu}{\sigma}\right) \\ &= \Pr\left(Z \leq \frac{x - \mu}{\sigma}\right) \\ &= \Phi\left(\frac{x - \mu}{\sigma}\right). \end{aligned} \quad (4.69)$$

Để tạo biến ngẫu nhiên có phân bố chuẩn có trung bình μ và độ lệch chuẩn σ trên Matlab, ta dùng hàm randn().

Cú pháp:

$$r = \mu + \sigma.*\text{randn}(M,N);$$

với M và N lần lượt là số hàng và số cột.

Bảng 4.6: Các hàm liên quan đến phân bố chuẩn.

TT	Hàm	Mô tả hàm
1	normrnd()	Tạo ra chuỗi ngẫu nhiên theo phân bố chuẩn
2	norminv()	Hàm ngược của hàm phân bố tích lũy của phân phối chuẩn
3	normpdf()	Hàm mật độ phân bố xác suất của phân phối chuẩn
4	normcdf()	Hàm phân bố xác suất tích lũy của phân phối chuẩn
5	normtstat()	Trả về giá trị kỳ vọng và phương sai của phân phối chuẩn
6	normfit()	Trả về giá trị ước lượng của kỳ vọng và phương sai của một ma trận ngẫu nhiên có phân bố chuẩn

Ví dụ 4.26

Tạo biến ngẫu nhiên có phân bố chuẩn phức có trung bình bằng không và phương sai là S cho trước.

Giải: Chúng ta cần tính độ lệch chuẩn σ cho phần thực và phần ảo từ phương sai, cụ thể: $S = 2\sigma^2$, dẫn đến $\sigma = \sqrt{S/2}$.

```
clc
% Số lượng mẫu
N = 100;
% Phương sai
S = 2;
% Trung bình
m = 0;
% Biến phân bố chuẩn phức
x = sqrt(S/2).*(randn(1,N) + 1i*randn(1,N));
% Kiểm tra
xVar = var(x)
xVar =
    1.9838
```

Ví dụ 4.27

Tạo ra hai biến ngẫu nhiên phân bố chuẩn có trung bình bằng không và phương sai bằng 1, chứng minh rằng:

- $\Pr(x > 0, y > 0) = \Pr(x > 0)\Pr(y > 0)$.
- $\Pr(x > 0 | y > 0) = \Pr(x > 0) + \Pr(y > 0) - \Pr(x > 0, y > 0)$.

Giải: Chương trình Matlab kiểm chứng như sau:

```

clc
N = 10^4;
x = randn(1,N);
y = randn(1,N);

% Câu a
PrLa = sum((x > 0) & (y > 0))/N
PrRa = sum((x > 0).*(y > 0))/N

% Câu b
PrLb = sum((x > 0) | (y > 0))/N
PrRb = sum(x > 0)/N + sum(y > 0)/N - sum((x > 0).*(y > 0))/N

```

Kết quả của câu a) cho vé trái và vé phải đều là 0.2431 và câu b) cho vé trái và vé phải là 0.7474.

Ví dụ 4.28

Tạo biến ngẫu nhiên phân bố chuẩn

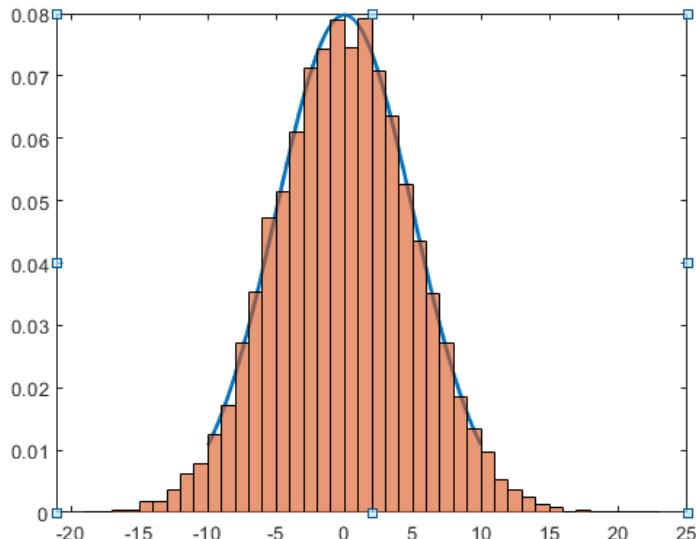
- Vẽ hàm mật độ phân phối xác suất của biến ngẫu nhiên.
- Vẽ biểu đồ tần suất của biến ngẫu nhiên.

Giải: Chúng ta sẽ sử dụng hàm normpdf() và hàm normrnd().

```

N = 10^4;
MU = 0;
SIGMA = 5;
x = -10:0.01:10;
% Câu a
pdf_t = normpdf(x,MU,SIGMA);
plot(x,pdf_t,'LineWidth',2)
hold on
% Câu b
pdf_e = normrnd(MU,SIGMA,1,N);
histogram(pdf_e,'Normalization','probability')

```



Hình 4.13: Hàm PDF và biểu đồ tần suất của phân bố chuẩn.

4.1.10.3 Phân bố mũ

Biến ngẫu nhiên liên tục X có hàm mật độ phân bố xác suất:

$$f(x) = \lambda e^{-\lambda x}, 0 < x < \infty \quad (4.70)$$

với $\lambda > 0$ sẽ là biến ngẫu nhiên có phân bố mũ với tham số λ . X có hàm phân bố xác suất tích lũy như sau:

$$F(x) = \int_0^x f(x)dx = 1 - e^{-\lambda x}. \quad (4.71)$$

Giá trị kỳ vọng của biến ngẫu nhiên X có phân bố mũ với tham số λ là:

$$E[X] = \int_0^\infty xf(x)dx = \frac{1}{\lambda}. \quad (4.72)$$

Phương sai của biến ngẫu nhiên X có phân bố mũ với tham số λ là:

$$\text{Var}[X] = \int_0^\infty \left(x - \frac{1}{\lambda} \right)^2 f(x)dx = \frac{1}{\lambda^2}. \quad (4.73)$$

Một tính chất quan trọng của phân bố mũ là tính không nhớ, phát biểu dưới dạng toán học là như sau:

$$\Pr(X > s + t | X > s) = \Pr(X > t) \quad (4.74)$$

hay

$$\Pr(X > s + t) = \Pr(X > s)\Pr(X > t). \quad (4.75)$$

Để tạo biến ngẫu nhiên có phân bố mũ trong Matlab, chúng ta có thể dùng hàm **exprnd()**.

Cú pháp:

`exprnd(MU,M,N)`

với MU là giá trị kỳ vọng, M là số hàng và N là số cột.

Bên cạnh hàm **exprnd()**, chúng ta cần quan tâm đến các hàm khác liên quan đến phân bố mũ như liệt kê ở Bảng 4.7.

Bảng 4.7: Các hàm liên quan đến phân bố mũ.

TT	Hàm	Mô tả hàm
1	<code>exprnd()</code>	Tạo ra ma trận ngẫu nhiên theo phân bố mũ với tham số cho trước
2	<code>expinv()</code>	Hàm ngược của hàm phân bố xác suất tích lũy của phân bố mũ
3	<code>exppdf()</code>	Hàm mật độ phân bố xác suất chuẩn của phân bố mũ
4	<code>expcdf()</code>	Hàm phân bố xác suất tích lũy của phân bố mũ
5	<code>exptat()</code>	Trả về giá trị trung bình và phương sai của phân bố mũ

6	expfit()	Trả về giá trị ước lượng của kỳ vọng và phương sai của một ma trận ngẫu nhiên có phân bố mũ
7	explike()	Trả về giá trị log-likelihood của phân bố mũ

Ví dụ 4.29

Tạo ra biến ngẫu nhiên phân bố mũ có tham số là 5 với kích thước 1 hàng và 100 cột

- Tìm giá trị kỳ vọng và phương sai theo lý thuyết.
- Tìm giá trị kỳ vọng và phương sai theo mô phỏng Matlab.
- Vẽ đồ thị tần suất và so sánh với hàm mật độ phân bố xác suất.

Giải: Chúng ta dùng hàm exprnd() để tạo biến ngẫu nhiên và dùng hàm stem() để vẽ đồ thị tần suất.

```

N = 10^4;
MU = 5;
NoB = 30;
r = exprnd(MU,1,N);

% Câu a
meanR_t = MU
varR_t = MU^2

% Câu b
meanR_s = mean(r)
varR_s = var(r)

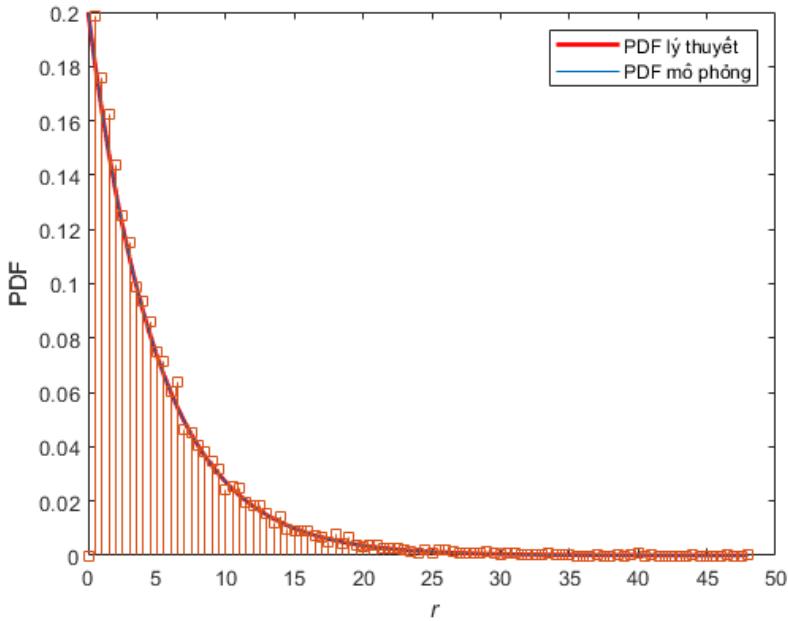
% Câu c
dx = 0.5;
Xm = max(r);
x = 0:dx:Xm;

PDF_t = 1/MU.*exp(-x./MU);
plot(x,PDF_t,'r','LineWidth',2)
hold on

% Vẽ đồ thị tần suất
PDF_s = zeros(size(x));
for idx = 2:length(x)
    PDF_s(idx) = PDF_s(idx) + sum((r >= x(idx-1))&(r < x(idx)))/(dx*N);
end

% Vẽ đồ thị so sánh
plot(x,PDF_t);
hold on
stem(x,PDF_s,'s');
xlabel('it r');
ylabel('PDF');
legend('PDF lý thuyết','PDF mô phỏng');
set(gcf,'color','white');

```



Hình 4.14: Đồ thị tần xuất và hàm PDF của phân bố mũ có tham số bằng 5.

4.1.10.4 Phân bố Rayleigh

Phân bố Rayleigh là một phân bố liên tục, có nhiều ứng dụng trong nhiều lĩnh vực kỹ thuật. Trong lĩnh vực viễn thông, phân bố Rayleigh được biết đến với kênh truyền fading Rayleigh, cụ thể phân bố Rayleigh được sử dụng để mô hình hóa hiện tượng đa đường trong kênh truyền fading. Ngoài ra, phân bố Rayleigh còn được sử dụng trong vật lý để mô hình hóa tốc độ gió, trong kỹ thuật y sinh để mô hình hóa thời gian sống của một đối tượng, trong khoa học y khoa để mô hình hóa phương sai nhiễu của ảnh cộng hưởng từ.

Gọi X là biến ngẫu nhiên phân bố Rayleigh, hàm PDF và CDF của X có dạng như sau:

$$f_X(x) = \frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}, x \geq 0 \quad (4.76)$$

và

$$F_X(x) = 1 - e^{-\frac{x^2}{2\sigma^2}}, x \geq 0 \quad (4.77)$$

Kỳ vọng của X là:

$$\sigma \sqrt{\frac{\pi}{2}} \quad (4.78)$$

và phương sai của X theo σ như sau:

$$\frac{4-\pi}{2} \sigma^2 \quad (4.79)$$

Nếu $X : N(0, \sigma^2)$ và $Y : N(0, \sigma^2)$ là hai biến có phân bố chuẩn độc lập và đồng dạng, thì

$$|Z| = |X + jY| = \sqrt{X^2 + Y^2} \quad (4.80)$$

sẽ có phân bố Rayleigh. Phương trình (4.80) là một cách tiếp cận để tạo ra kênh truyền fading Rayleigh.

Liên quan đến phân bố Rayleigh, Matlab cung cấp cho chúng ta các hàm ở Bảng 4.8.

Bảng 4.8: Các hàm liên quan đến phân bố Rayleigh.

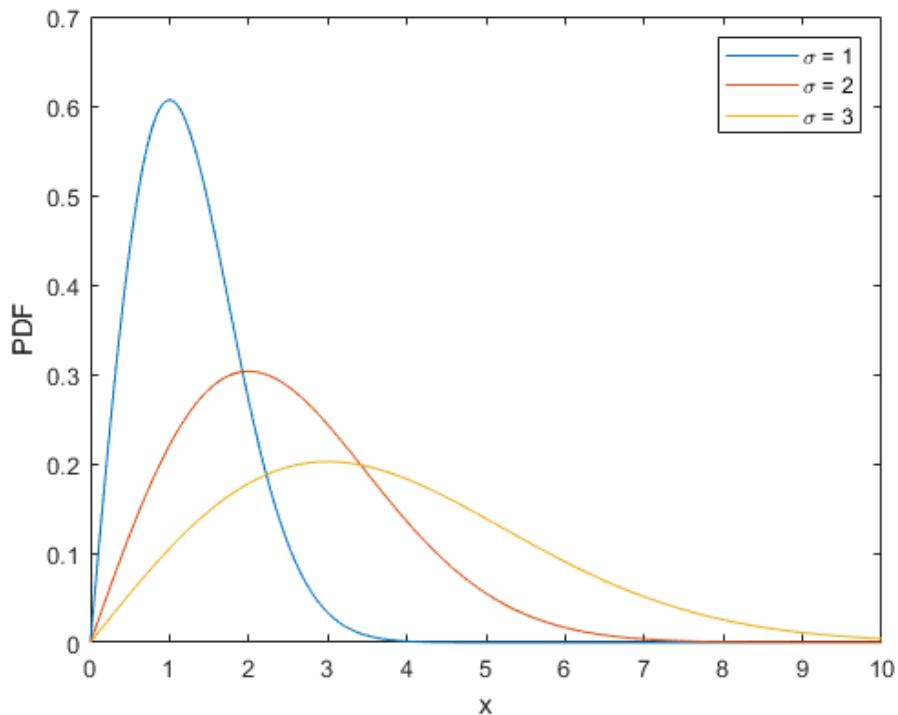
TT	Hàm	Mô tả hàm
1	raylrnd()	Tạo ra chuỗi ngẫu nhiên theo phân bố Rayleigh
2	raylinv()	Hàm ngược của hàm phân bố xác suất tích lũy của phân bố Rayleigh
3	raylpdf()	Hàm mật độ phân bố xác suất của phân bố Rayleigh
4	raylcdf()	Hàm phân bố xác suất tích lũy của phân bố Rayleigh
5	rayltat()	Trả về giá trị kỳ vọng và phương sai của phân bố Rayleigh
6	raylfit()	Trả về giá trị ước lượng của kỳ vọng và phương sai của một ma trận ngẫu nhiên có phân bố Rayleigh
7	rayllike()	Trả về giá trị log-likelihood cho phân bố Rayleigh

Ví dụ 4.30

Vẽ hàm PDF của phân bố Rayleigh sử dụng hàm raylpdf() cho tham số 1, 2 và 3.

Giải: Chúng ta sẽ sử dụng hàm raylpdf(). Chương trình Matlab như sau:

```
x=0:0.01:10;
b = [1 2 3];
r1 = raylpdf(x,1);
r2 = raylpdf(x,2);
r3 = raylpdf(x,3);
plot(x,r1,x,r2,x,r3)
legend('sigma = 1','sigma = 2','sigma = 3');
xlabel('x');
ylabel('PDF');
set(gcf,'color','white');
```



Hình 4.15: Hàm PDF của phân bố Rayleigh.

Ví dụ 4.31

Hãy tạo ra biến ngẫu nhiên phân bố Rayleigh có $\sigma = 2$. Vẽ biểu đồ tần suất và so sánh lý thuyết.

Giải: Chúng ta sử dụng hàm raylrnd(). Chương trình Matlab với 10^4 mẫu như sau:

```
N = 10^4;
sigma = 2;
r = raylrnd(sigma,1,N);
dx = 0.3;
Xm = 8;
x = 0:dx:Xm;

% Mô phỏng
CDF_s = zeros(size(x));
for idx = 1:length(x)
    CDF_s(idx) = CDF_s(idx) + sum(r <= x(idx))/N;
end

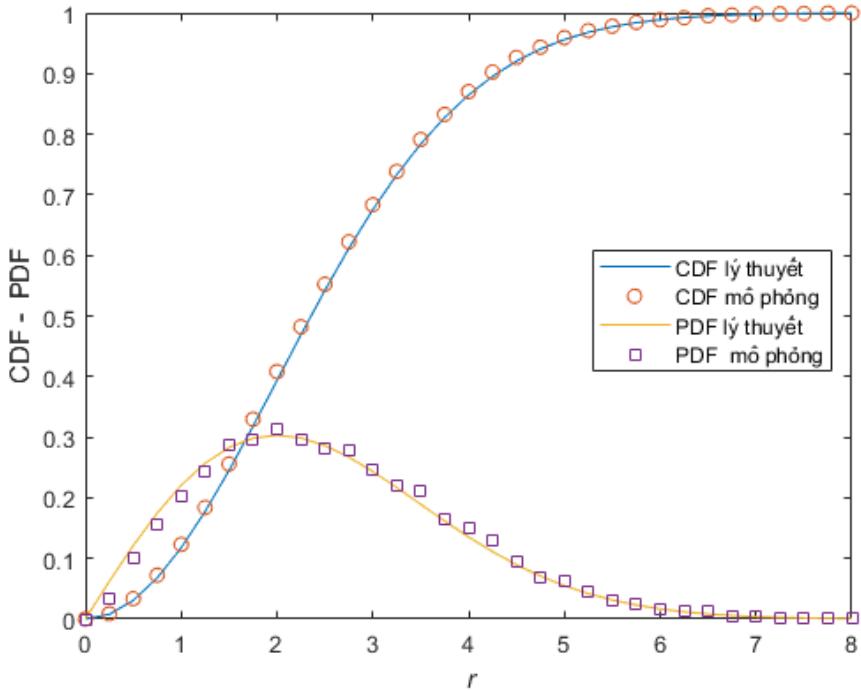
PDF_s = zeros(size(x));
for idx = 2:length(x)
    PDF_s(idx) = PDF_s(idx) + sum((r >= x(idx-1))&(r < x(idx)))/(dx*N);
end

% Lý thuyết
CDF_t = 1 - exp(-x.^2./(2*sigma.^2));
PDF_t = x./sigma.^2.*exp(-x.^2./(2*sigma.^2));
```

```

plot(x,CDF_t,x,CDF_s,'o',x,PDF_t,x,PDF_s,'s');
xlabel('ít x');
ylabel('CDF - PDF');
legend('CDF lý thuyết','CDF mô phỏng','PDF lý thuyết','PDF mô phỏng');
set(gcf,'color','white');

```



Hình 4.16: Hàm PDF và CDF của phân bố Rayleigh sử dụng hàm raylrnd().

Ví dụ 4.32

Tương tự Ví dụ 4.31, hãy tạo ra biến ngẫu nhiên có phân bố Rayleigh từ biến phân bố chuẩn. Vẽ biểu đồ tần suất và so sánh lý thuyết.

Giải: Từ (4.80), chúng ta thay thế một dòng lệnh Matlab ở Ví dụ 4.31 như sau:

```
r = raylrnd(sigma,1,N);
```

bằng

```
r = abs(sigma.*randn(1,N) + 1i*randn(1,N));
```

Kết quả đạt được sẽ tương tự như Hình 4.16.

4.1.10.5 Phân bố Nakagami- m

Phân bố Nakagami- m được đề xuất vào năm 1960 để mô hình hóa mô hình kênh truyền fading mà sau đó được đặt tên là kênh truyền fading Nakagami- m . Gọi X là biến ngẫu nhiên có phân bố Nakagami- m , ta có hàm PDF và CDF của X như sau:

$$f_X(x) = \frac{2m^m}{\Gamma(m)\Omega^m} x^{2m-1} \exp\left(-\frac{m}{\Omega}x^2\right), \forall x \geq 0 \quad (4.81)$$

$$F(x) = \Gamma\left(m, \frac{m}{\Omega}x^2\right) \quad (4.82)$$

với $m \geq 1/2$, $\Omega > 0$ và $\Gamma(.,.)$ là hàm Gamma không hoàn chỉnh [5].

Kỳ vọng của X được tính theo công thức sau:

$$E[X] = \frac{\Gamma(m+1/2)}{\Gamma(m)} \left(\frac{\Omega}{m}\right)^{1/2}. \quad (4.83)$$

Phương sai của X là hàm của m và Ω như sau:

$$\text{Var}[X] = \Omega \left[1 - \frac{1}{m} \left(\frac{\Gamma(m+1/2)}{\Gamma(m)} \right)^2 \right]. \quad (4.84)$$

Phân bố Nakagami- m có mối liên hệ với phân bố Gamma, cụ thể nếu X có phân bố Nakagami theo tham số m và Ω thì X^2 có phân bố Gamma với tham số m và $\frac{\Omega}{m}$. Ngoài ra, chúng ta có thể tạo ra phân bố Nakagami có tham số m và Ω từ phân bố Chi, Y , có tham số $k = 2m$, cụ thể là:

$$X = \sqrt{\frac{\Omega}{2m}} Y. \quad (4.85)$$

Nếu kênh truyền có phân bố Nakagami- m , nghĩa là biên độ của hệ số kênh truyền phức sẽ có phân bố Nakagami- m và tỷ số tín hiệu trên nhiễu, γ , sẽ có phân bố Gamma. Hàm PDF của γ là:

$$f_\gamma(\gamma) = \frac{m^m \gamma^{m-1}}{\bar{\gamma}^m \Gamma(m)} \exp\left(-\frac{m\gamma}{\bar{\gamma}}\right) \quad (4.86)$$

với giá trị trung bình của γ là $\bar{\gamma} = E[\gamma]$.

Hàm CDF của γ sẽ là:

$$F_\gamma(\gamma) = 1 - \frac{m^m \gamma^{m-1}}{\bar{\gamma}^m \Gamma(m)} \exp\left(-\frac{m\gamma}{\bar{\gamma}}\right). \quad (4.87)$$

Matlab không hỗ trợ trực tiếp cho phân bố Nakagami- m ngoại trừ Toolbox Statistics and Machine Learning Toolbox với mục Probability Distributions. Các hàm liên quan đến phân bố Nakagami- m như Bảng 4.9.

Bảng 4.9: Các hàm liên quan đến phân bố Nakagami- m .

TT	Tên hàm	Thuyết minh
1	makedist()	Tạo đối tượng phân bố xác suất.
2	cdf()	Hàm phân bố xác suất tích lũy của phân bố Nakagami- m cho đối tượng tạo ra từ hàm makedist().
3	icdf()	Hàm ngược của hàm phân bố xác suất tích lũy của phân bố Nakagami- m cho đối tượng tạo ra từ hàm makedist().
4	mean()	Tính kỳ vọng của phân bố Nakagami- m cho đối tượng tạo ra từ hàm makedist().
5	median()	Tính trung vị của phân bố Nakagami- m cho đối tượng tạo ra từ hàm makedist().
6	pdf()	Hàm mật độ phân bố xác suất của phân bố Nakagami- m cho đối tượng tạo ra từ hàm makedist().
7	random()	Tạo biến ngẫu nhiên có phân bố Nakagami- m cho đối tượng tạo ra từ hàm makedist().
8	std()	Tính độ lệch chuẩn cho đối tượng tạo ra từ hàm makedist().

Ví dụ 4.33

Sử dụng hàm makedist() để tạo đối tượng phân bố ngẫu nhiên có phân bố Nakagami- m với tham $m = 2$ và $\Omega = 1$. Vẽ biểu đồ tần suất và so sánh với hàm PDF lý thuyết.

Giải: Chúng ta sử dụng hàm random() để tạo ra biến ngẫu nhiên và hàm pdf() để trả về giá trị của PDF. Chương trình Matlab như sau:

```

clc

%% Mô phỏng
N = 10^6;
% Shape parameter
mu = 2;
% Scale parameter
Omega = 2;
% Tạo biến ngẫu nhiên
pd = makedist('Nakagami','mu',mu,'omega',Omega);
r = pd.random(1,N);

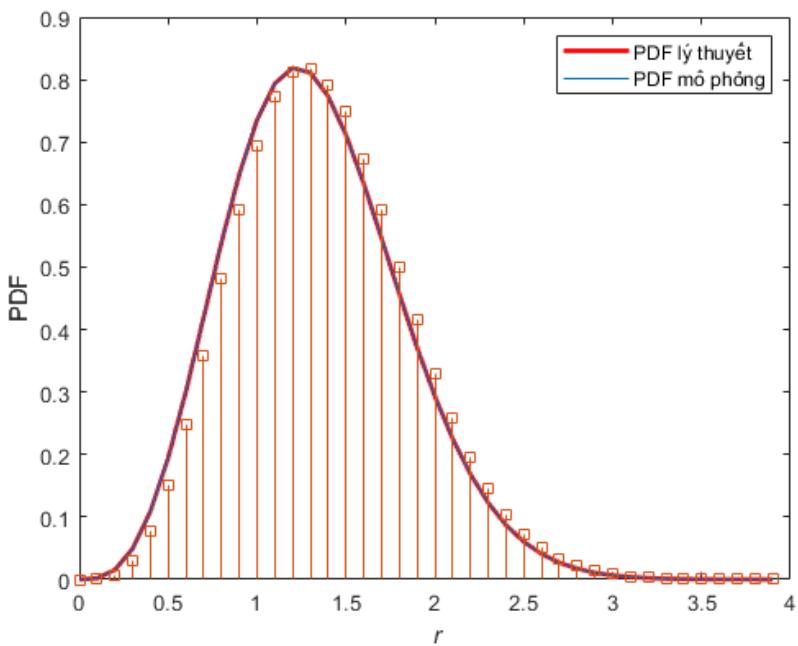
% Vẽ đồ thị tần suất
dx = 0.1;
Xm = max(r);
x = 0:dx:Xm;

PDF_t = pd.pdf(x);
plot(x,PDF_t,'r','LineWidth',2)
hold on

% Vẽ tần suất
PDF_s = zeros(size(x));
for idx = 2:length(x)
    PDF_s(idx) = PDF_s(idx) + sum((r >= x(idx-1))&(r < x(idx)))/(dx*N);
end

```

```
% Vẽ hình so sánh
plot(x,PDF_t);
hold on
stem(x,PDF_s,'s');
xlabel('r');
ylabel('PDF');
legend('PDF lý thuyết','PDF mô phỏng');
set(gcf,'color','white');
```



Hình 4.17: Hàm PDF lý thuyết và mô phỏng của phân bố Nakagami- m .

Ví dụ 4.34

Hãy mô phỏng tạo biến ngẫu nhiên có phân bố Nakagami- m $m = 3$ và $\Omega = 2$ từ phân bố Gamma. Vẽ biểu đồ tần suất và so sánh với hàm PDF lý thuyết bằng cách sử dụng hàm pdf() từ đối tượng phân bố tạo ra từ hàm makedist().

Giải: Chúng ta sử dụng công thức (4.85) và hàm chi2rnd() để tạo biến ngẫu nhiên có phân bố Chi. Chương trình Matlab là như sau.

```
clear all
clc
N = 10^4;
% Shape parameter
mu = 3;
% Scale parameter
Omega = 2;

% Tạo biến ngẫu nhiên Nakagami từ biến ngẫu nhiên phân bố Chi
r = sqrt(Omega/(2*mu)).*sqrt(chi2rnd(2*mu,1,N));
% Thiết lập thông số vẽ đồ thị
```

```

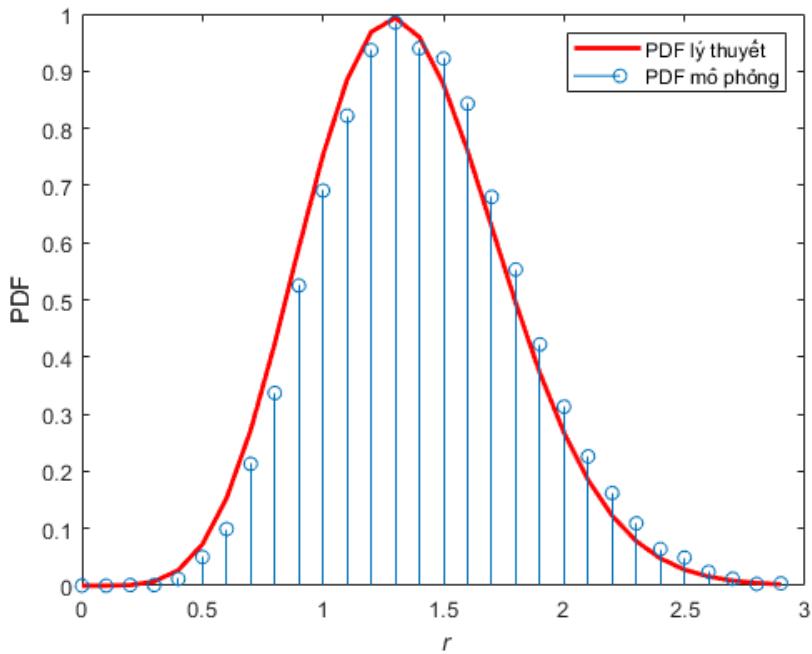
dx = 0.1;
Xm = max(r);
x = 0:dx:Xm;

% Vẽ tần suất
PDF_s = zeros(size(x));
for idx = 2:length(x)
    PDF_s(idx) = PDF_s(idx) + sum((r >= x(idx-1))&(r < x(idx)))/(dx*N);
end

% Vẽ đồ thị PDF phân bố Nakagami-m lý thuyết
pd = makedist('Nakagami','mu',mu,'omega',Omega);
r = pd.random(1,N);
PDF_t = pd.pdf(x);

% Vẽ hình so sánh
plot(x,PDF_t,r,'LineWidth',2);
hold on
stem(x,PDF_s,'o');
xlabel('r');
ylabel('PDF');
legend('PDF lý thuyết','PDF mô phỏng');
set(gcf,'color','white');

```



Hình 4.18: Vẽ hàm PDF của phân bố Nakagami-m từ phân bố Chi.

4.2 Mô phỏng nguồn tín hiệu

4.2.1 Tín hiệu tương tự

Toolbox Signal Processing trong Matlab cung cấp các hàm để tạo và/hoặc vẽ các dạng tín hiệu tương tự liệt kê trong Bảng 4.10.

Bảng 4.10: Bảng liệt kê các hàm tạo và vẽ tín hiệu tương tự.

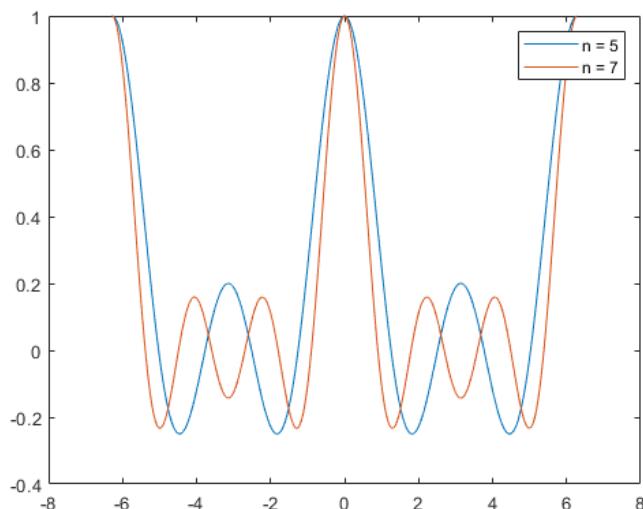
TT	Tên hàm	Thuyết minh	Cú pháp
1	chirp()	Trả về tín hiệu quét tần số cosin	$y = \text{chirp}(t, f_0, t_1, f_1, \text{method})$
2	diric()	Vẽ hàm Diriclet	$y = \text{diric}(x, n)$
3	gauspuls()	Vẽ xung cao tần của tín hiệu điều chế Gaussian	$y = \text{gauspuls}(t, fc, bw)$
4	gmonopuls()	Vẽ các xung đơn Gauss	$y = \text{gmonopuls}(t, fc)$
5	pulstran()	Vẽ các xung theo hình dạng quy định	$y = \text{pulstran}(t, d, \text{func}, fs)$
6	rectpuls()	Vẽ xung chữ nhật không tuần hoàn	$y = \text{rectpuls}(t, w)$
7	sawtooth()	Vẽ tín hiệu dạng hình răng cưa hay hình tam giác	$x = \text{sawtooth}(t, xmax)$
8	sinc()	Hàm trả về giá trị của hàm sinc. Kết hợp với hàm plot nếu muốn vẽ hàm.	$y = \text{sinc}(x)$
9	square()	Vẽ tín hiệu xung vuông	$x = \text{square}(t, duty)$
10	stem()	Vẽ tín hiệu dạng chuỗi rời rạc	$\text{stem}(X, Y)$
11	tripuls()	Vẽ tín hiệu dạng tam giác không tuần hoàn	$y = \text{tripuls}(T, w, s)$
12	vco()	Bộ dao động điều khiển theo điện áp	$y = \text{vco}(x, fc, fs)$

Ví dụ 4.35

Vẽ hàm Dirichlet trong khoảng $(-2\pi, 2\pi)$ cho $n = 5$ và $n = 7$.

Giải: Chúng ta sử dụng hàm diric() với chương trình Matlab như sau:

```
N = 200;
x = -2*pi:1/N:2*pi;
y5 = diric(x,5);
y7 = diric(x,7);
plot(x,y5,x,y7);
legend('n = 5','n = 7');
set(gcf,'color','white');
```



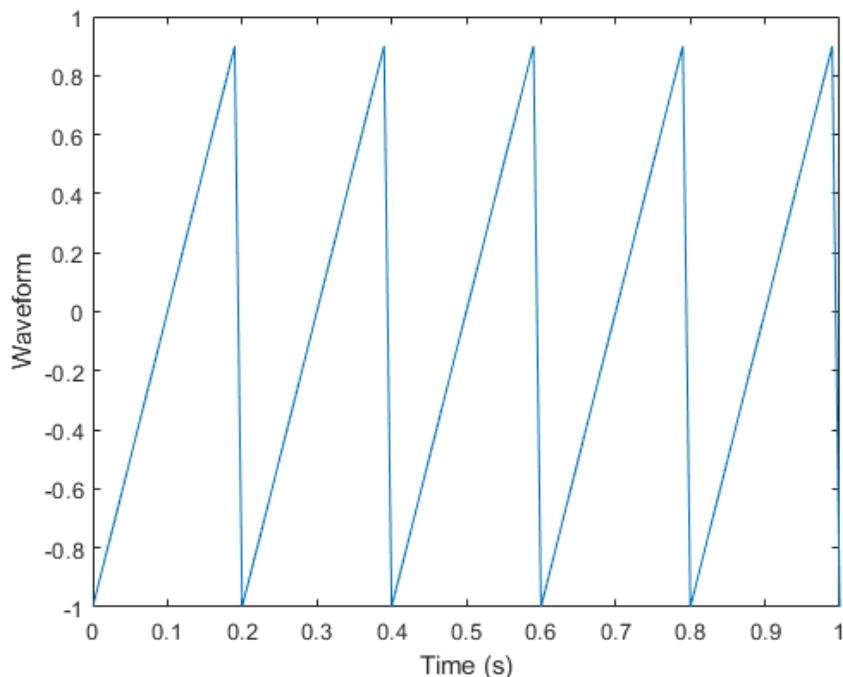
Hình 4.19: Tín hiệu Dirichlet .

Ví dụ 4.36

Hãy tạo tín hiệu răng cưa có tần số là 5 Hz, tần số lấy mẫu là 100 Hz và thời gian là 1 s.

Giải: Chúng ta sử dụng hàm **sawtooth()** với chương trình Matlab như sau:

```
T = 1;  
fc = 5;  
fs = 10^2;  
t = 0:1/fs:T;  
x = sawtooth(2*pi*fc*t);  
plot(t,x);  
set(gcf,'color','white');  
xlabel('Time (s)')  
ylabel('Waveform')
```



Hình 4.20: Tín hiệu răng cưa.

4.2.2 Tín hiệu số

Trong hệ thống thông tin số, dữ liệu vào thông thường là tín hiệu số. Để tạo tín hiệu số cho mô phỏng, Matlab hỗ trợ chúng ta các hàm trong Bảng 4.11.

Bảng 4.11: Danh mục các hàm Matlab tạo tín hiệu số.

TT	Tên hàm Matlab	Chú thích
1	randi()	Hàm tín hiệu số rời rạc phân bố đều.
2	randerr()	Hàm tạo mẫu lõi ngẫu nhiên.
3	randsrc()	Hàm tạo tín hiệu nguồn ngẫu nhiên theo các định nghĩa cho trước.

Cú pháp:

`randi(IMAX,N)`: tạo ra ma trận tín hiệu số rời rạc phân bố đều từ 1 đến IMAX có kích thước $N \times N$.

`randi(IMAX,M,N)`: tạo ra ma trận tín hiệu số rời rạc phân bố đều từ 1 đến IMAX có kích thước $M \times N$.

`randi([IMIN IMAX],M,N)`: tạo ra ma trận tín hiệu số rời rạc phân bố đều từ IMIN đến IMAX có kích thước $M \times N$.

Ví dụ 4.37

Tạo chuỗi bit có chiều dài 1000 bit, tính xác suất xuất hiện của bit 0 và bit 1.

Giải: Chuỗi bit sẽ có hai giá trị 0 và 1. Chúng ta sử dụng hàm `randi()` để tạo chuỗi bit.

```
N = 1000;  
X = randi([0 1],1,N);  
Pr0=sum(x==0)/N  
Pr1=sum(x==1)/N
```

```
ans =
```

```
0.4990
```

```
ans =
```

```
0.5010
```

Cú pháp:

`OUT = randerr(M)`: tạo ra ma trận lỗi OUT kích thước $M \times M$ có 1 bit một ở vị trí ngẫu nhiên trên mỗi hàng.

`OUT = randerr(M,N)`: tạo ra ma trận OUT kích thước $M \times N$ có 1 bit một ở vị trí ngẫu nhiên trên mỗi hàng.

`OUT = randerr(M,N,ERRORS)`: tạo ra ma trận OUT kích thước $M \times N$ có số lượng bit 1 trên mỗi hàng quy định bởi ERROR.

Ví dụ 4.38

- Tạo ma trận lỗi 3×3 có 1 lỗi trên mỗi hàng.
- Tạo ma trận lỗi 3×10 có 1 lỗi trên mỗi hàng.
- Tạo ma trận lỗi 10×5 với 1 lỗi có xác suất 80% và 3 lỗi có xác suất 20%.

Giải: Chúng ta dùng hàm `randerr()` với chương trình Matlab là như sau.

```
randerr(3,3)  
randerr(3,10)  
randerr(10,5,[1 3; 0.8 0.2])
```

Kết quả là như sau.

```

ans =
 1  0  0
 0  1  0
 0  1  0

ans =
 0  0  0  0  0  0  0  0  0  1
 0  0  0  0  0  0  0  1  0  0
 0  0  0  0  1  0  0  0  0  0

ans =
 0  1  0  0  0
 0  0  0  0  1
 0  1  0  0  0
 0  0  0  1  0
 1  0  0  0  0
 1  1  1  0  0
 0  0  1  0  0
 1  0  0  1  1
 0  0  1  0  0
 1  1  1  0  0

```

Cú pháp:

out = randsrc: tạo ra một biến ngẫu nhiên phân bố đều có giá trị 1 hay -1.

out = randsrc(m): tạo ra ma trận có kích thước $m \times m$ với các phần tử trong ma trận có thể lấy giá trị 1 hay -1 một cách độc lập với xác suất bằng nhau.

out = randsrc(m,n): tạo ra ma trận có kích thước $m \times n$ với các phần tử trong ma trận là 1 hay -1 với xác suất xuất hiện bằng nhau.

out = randsrc(m,n, [A;p]): tạo ra ma trận có kích thước $m \times n$ với các phần tử trong ma trận thuộc vector A với xác suất xuất hiện tương ứng thuộc vector p.

Ví dụ 4.39

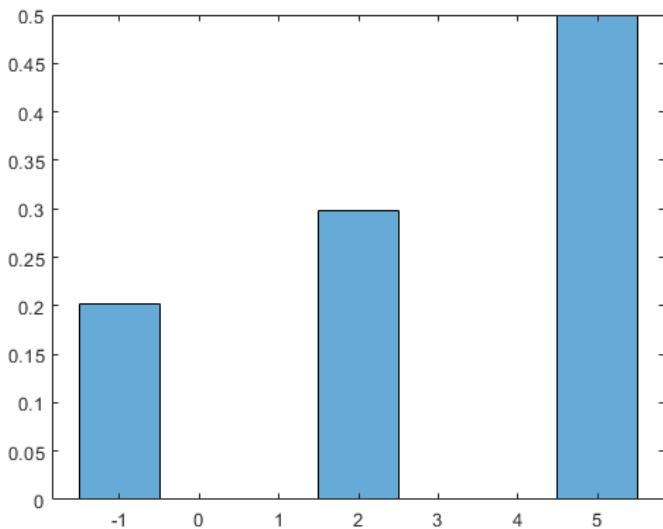
Tạo một nguồn tín hiệu có 1000 phần tử chỉ có ba giá trị -1, 2 và 5, với xác suất tương ứng là 0.2, 0.3 và 0.5.

Giải: Ta dễ dàng nhận thấy $A = [-1 2 5]$ và $p = [0.2 0.3 0.5]$. Chương trình Matlab như sau.

```

% Chiều dài của nguồn
N = 1000;
% Các tín hiệu của nguồn
A = [-1 2 5];
% Xác suất của tín hiệu nguồn
p = [0.2 0.3 0.5];
% Tạo tín hiệu nguồn
src = randsrc(1,N,[A;p]);
% Vẽ đồ thị tần suất chuẩn hóa để kiểm chứng
histogram(src,'Normalization','probability')

```



Hình 4.21: Biểu đồ tần suất chuẩn hóa của các phần tử trong nguồn để kiểm chứng hàm `randsrc()`.

4.2.3 Tín hiệu ngẫu nhiên

Để tạo tín hiệu ngẫu nhiên theo một phân bố với tham số cho trước, chúng ta sử dụng hàm `random()` với các phân bố mà Matlab hỗ trợ trong Bảng 4.12.

Trong trường hợp, chúng ta không biết tên phân bố mà chỉ biết hàm PDF hay CDF của biến ngẫu nhiên cần tạo, chúng ta có thể tạo ra biến ngẫu nhiên bằng cách sử dụng phương pháp đề cập ở mục 4.1.1.2 hoặc 4.1.1.3.

Cú pháp:

`R = random(NAME,A,B,C)`

% NAME là tên phân bố cho trong Bảng 4.12.

% A, B và C là các tham số đầu vào của phân bố.

Bảng 4.12: Các phân bố mà hàm `random()` hỗ trợ.

TT	Tên phân phối chuẩn hóa theo hàm <code>random</code> (NAME)	Tên phân phối	Tham số đầu vào của hàm <code>random()</code> (A, B và C)
1	'Beta'	Phân phối Beta	a first shape parameter, b second shape parameter
2	'Binomial'	Phân phối Binomial	n number of trials, p probability of success for each trial
3	'BirnbaumSaunders'	Phân phối Birnbaum-Saunders	β scale parameter, γ shape parameter
4	'Burr'	Phân phối Burr Type XII	α scale parameter, c first shape parameter, k second shape parameter
5	'Chisquare'	Phân phối Chi-Square	v degrees of freedom
6	'Exponential'	Phân phối Exponential	μ mean

7	'Extreme Value'	Phân phối Extreme Value	μ location parameter, σ scale parameter
8	'F'	Phân phối F	v1 numerator degrees of freedom, v2 denominator degrees of freedom
9	'Gamma'	Phân phối Gamma	a shape parameter, b scale parameter
10	'Generalized Extreme Value'	Phân phối Generalized Extreme Value	k shape parameter, σ scale parameter, μ location parameter
11	'Generalized Pareto'	Phân phối Generalized Pareto	k tail index (shape) parameter, σ scale parameter, μ threshold (location) parameter
12	'Geometric'	Phân phối Geometric	p probability parameter
13	'HalfNormal'	Phân phối Half-Normal	μ location parameter, σ scale parameter
14	'Hypergeometric'	Phân phối Hypergeometric	m size of the population, k number of items with the desired characteristic in the population, n number of samples drawn
15	'InverseGaussian'	Phân phối Inverse Gaussian	μ scale parameter, λ shape parameter
16	'Logistic'	Phân phối Logistic	μ mean, σ scale parameter
17	'LogLogistic'	Phân phối Loglogistic	μ mean of logarithmic values, σ scale parameter of logarithmic values
18	'Lognormal'	Phân phối Lognormal	μ mean of logarithmic values, σ standard deviation of logarithmic values
19	'Nakagami'	Phân phối Nakagami	μ shape parameter, ω scale parameter
20	'Negative Binomial'	Phân phối Negative Binomial	r number of successes, p probability of success in a single trial
21	'Noncentral F'	Phân phối Noncentral F	v1 numerator degrees of freedom, v2 denominator degrees of freedom, δ noncentrality parameter
22	'Noncentral t'	Phân phối Noncentral t	v degrees of freedom, δ noncentrality parameter
23	'Noncentral Chi-square'	Phân phối Noncentral Chi-Square	v degrees of freedom, δ noncentrality parameter
24	'Normal'	Phân phối Normal	μ mean, σ standard deviation
25	'Poisson'	Phân phối Poisson	λ mean
26	'Rayleigh'	Phân phối Rayleigh	b scale parameter
27	'Rician'	Phân phối Rician	s noncentrality parameter, σ scale parameter
28	'Stable'	Phân phối Stable	α first shape parameter, β second shape parameter, γ scale parameter, δ location parameter
29	't'	Phân phối Student's t	v degrees of freedom

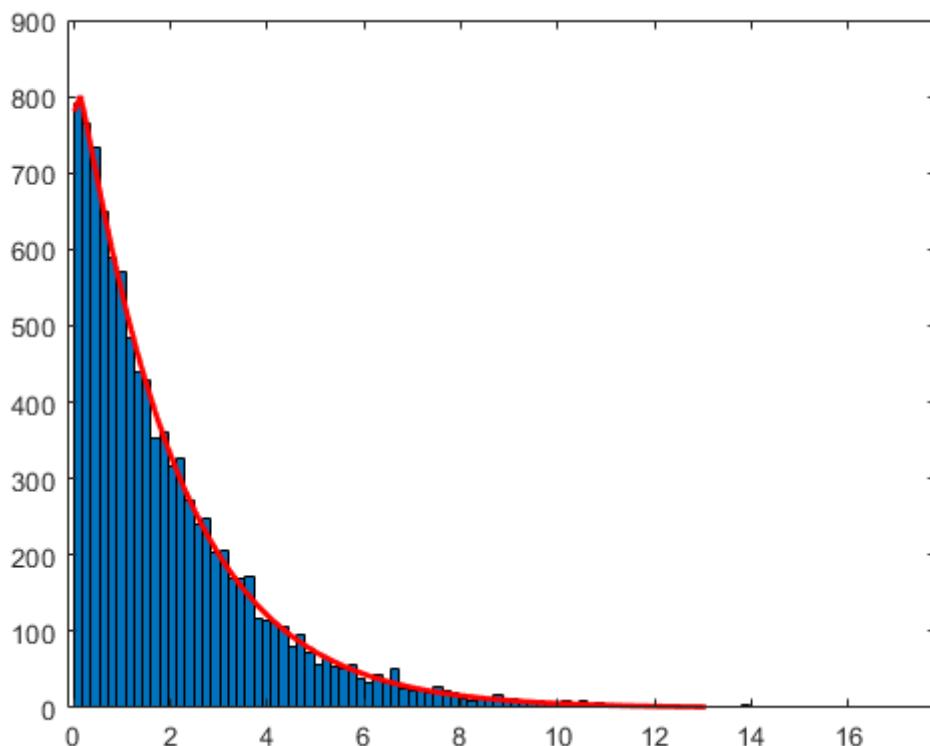
30	'tLocationScale'	Phân phối t Location-Scale	μ location parameter, σ scale parameter, v shape parameter
31	'Uniform'	Phân phối Uniform (Continuous)	a lower endpoint (minimum), b upper endpoint (maximum)
32	'Discrete Uniform'	Phân phối Uniform (Discrete)	n maximum observable value
33	'Weibull'	Phân phối Weibull	a scale parameter, b shape parameter

Ví dụ 4.40

Tạo biến ngẫu nhiên có 10^4 phần tử có phân bố Gamma với hai tham số lần lượt là 1 và 2.

Giải: Trong Ví dụ 4.40, chúng ta sử dụng hàm **histfit()** để kiểm chứng kết quả tạo biến ngẫu nhiên.

```
% Phân bố Gamma
nameDist = 'Gamma';
a = 1;
b = 2;
% Số lượng mẫu cần tạo
N = 10000;
r = random(nameDist,a,b,[N 1]);
% Vẽ đồ thị histogram
histfit(r,100,nameDist)
set(gcf,'color','white');
```



Hình 4.22: Đồ thị tần suất của phân phối Gamma.

4.3 Mã hóa

4.3.1 Mã hóa nguồn

Mã hóa nguồn là quá trình nén thông tin của nguồn (loại bỏ tín hiệu dư thừa) trước khi truyền đi để tăng hiệu quả cho quá trình truyền thông tin. Mong muốn của quá trình nén là tiến tới giới hạn entropy của nguồn với entropy là đơn vị chỉ ra lượng thông tin có trong mỗi ký tự của nguồn. Vì tần suất xuất hiện của mỗi ký tự là khác nhau, nên mỗi ký tự mang một lượng thông tin khác nhau [6].

Thông tin của nguồn có thể là dữ liệu thoại, số liệu hay hình ảnh của quá trình truyền thông giữa con người với con người, con người với máy và máy với máy. Một số phương pháp mã hóa nguồn cho phép tiến tới giới hạn entropy, ví dụ như Huffman, Shannon Fano và Lempel-Ziv-Welch.

Mã hóa nguồn bao gồm ba bước cơ bản sau:

- Nén tín hiệu (thường áp dụng cho tín hiệu âm thanh với giải thuật logarithm).
- Lượng tử tín hiệu.
- Mã hóa tín hiệu.

Nén tín hiệu dùng giải thuật logarithm là để tăng giải động của âm thanh hay tăng tỷ số tín hiệu trên nhiễu của tín hiệu, đó là giải thuật “ μ -law” hay “A-law”. Matlab hỗ trợ hàm **com pand()** để thực hiện nén/giải nén cả hai loại giải thuật “ μ -law” hay “A-law”. Quá trình nén tín hiệu nếu có thường áp dụng trước khi thực hiện lượng tử. Cú pháp của hàm com pand() như sau:

Cú pháp:

```
out = com pand(in,Mu,v)  
out = com pand(in,Mu,v,'mu/compressor')  
out = com pand(in,Mu,v,'mu/expander')  
out = com pand(in,A,v,'A/compressor')  
out = com pand(in,A,v,'A/expander')
```

Ví dụ 4.41

Cho tín hiệu $y = e^x$ với x từ -4 đến 4. Hãy nén theo luật μ .

- a. Vẽ tín hiệu trước và sau khi nén.
- b. So sánh giá trị méo dạng trước và sau khi nén.

Giải: Chương trình mã Matlab cho Ví dụ 4.41 như sau. Lưu ý rằng hệ số méo dạng là sai số bình phương trung bình giữa tín hiệu trước và sau khi nén với tín hiệu gốc.

```

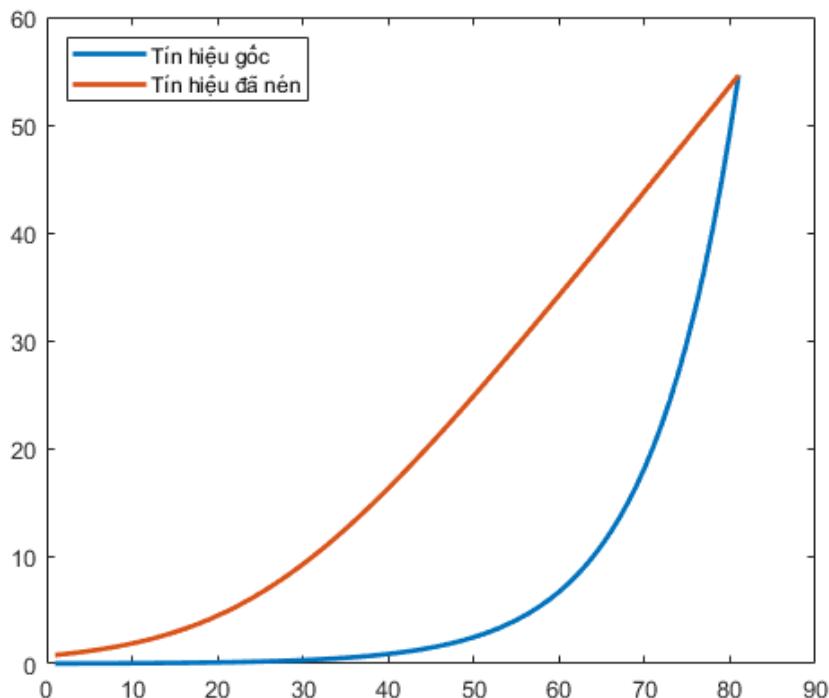
% Số mức lượng tử
Mu = 255;
% Tín hiệu hàm mũ
t = -4:0.1:4;
sig = exp(t);
V = max(sig);
% Lượng tử tín hiệu
partition = 0:2^6-1;
codebook = 0:2^6;
[~,~,distor1] = quantiz(sig,partition,codebook)

% Nén tín hiệu theo luật mu
compsig = compand(sig,Mu,V,'mu/compressor');
[~,quants] = quantiz(compsig,partition,codebook);
% Giải nén tín hiệu theo luật mu
newsig = compand(quants,Mu,max(quants),'mu/expander');
distor2 = sum((newsig-sig).^2)/length(sig)

% Vẽ tín hiệu
plot([sig' compsig'],'linewidth',2);
legend('Tín hiệu gốc','Tín hiệu đã nén','location','nw')
set(gcf,'color','white');

```

Hệ số méo dạng trước và sau khi nén lần lượt là .5348 và .0397. Chúng ta thấy rằng hệ số méo dạng giảm hơn 10 lần sau khi áp dụng luật nén.



Hình 4.23 Tín hiệu trước và sau khi nén

Cho quá trình lượng tử tín hiệu, Matlab hỗ trợ các hàm cho trong Bảng 4.13.

Bảng 4.13: Bảng danh mục các hàm cho quá trình lượng tử tín hiệu.

TT	Hàm	Chú thích	Cú pháp
1	quantiz()	Trả về chỉ số lượng tử và giá trị lượng tử.	index = quantiz(sig,partition,codebook)
2	lloyds()	Tối ưu các tham số lượng tử theo giải thuật Lloyds	index = quantiz(sig,partition)
3	dpcmenco()	Mã hóa vi sai	[indx,quants] = dpcmenco(sig,codebook,partition,predictor)
4	dpcmdeco()	Giải mã vi sai	[sig,quanterror] = dpcmdeco(indx,codebook,predictor)
5	dpcmopt()	Tối ưu mã hóa vi sai	[predictor,codebook,partition] = dpcmopt(training_set,ord,len)

Các tham số của các hàm trong cột cuối cùng ở Bảng 4.13 có thể xem thêm ở phần hướng dẫn của hàm thông qua hàm help() hay hàm doc().

Ví dụ 4.42

Hãy tạo tín hiệu cos trong 2 chu kỳ và thực hiện lượng tử với 12 vùng. Vẽ tín hiệu trước và sau khi lượng tử.

Giải: Chúng ta chọn một bước mẫu là 0.1.

```
t = [0:.1:4*pi];
% Tạo tín hiệu cos
sig = cos(t);

% Chia vùng
partition = [-1:.2:1];

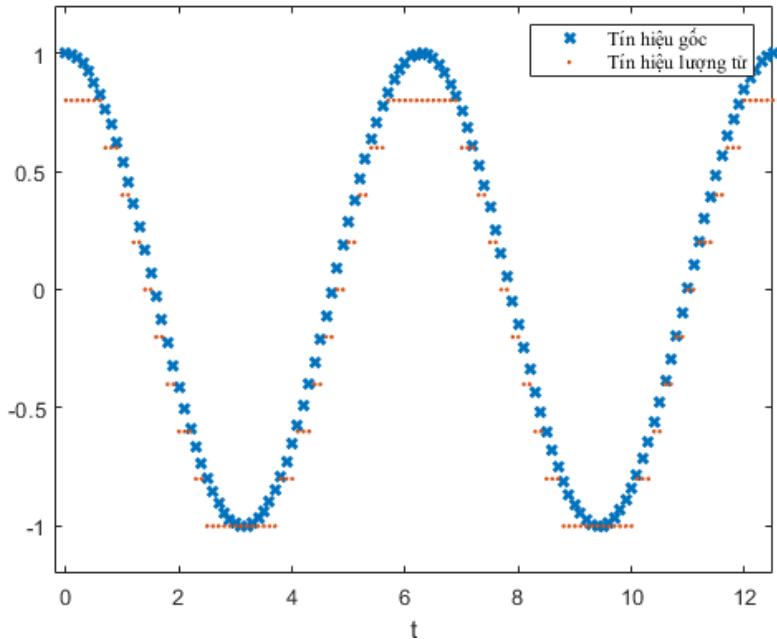
% Từ mã
codebook = [-1.2:.2:1];

% Lượng tử
[index,quants] = quantiz(sig,partition,codebook);

% Vẽ tín hiệu trước và sau khi lượng tử
plot(t,sig,'x',t,quants,'.','linewidth',2)

% Chú thích cho hình
legend('Tín hiệu gốc','Tín hiệu lượng tử','fontname','Time New Roman');

% Điều chỉnh vùng vẽ theo trục x và trục y của hình
axis([-2 max(t) -1.2 1.2])
set(gcf,'color','white');
xlabel('t');
```



Hình 4.24: Tín hiệu gốc và tín hiệu sau khi lấy mẫu.

Một nhận xét quan trọng từ Hình 4.24 là các tham số mã hóa ảnh hưởng nhiều đến méo dạng tín hiệu sau khi lấy mẫu, do đó việc tối ưu các tham số của quá trình lượng tử là rất quan trọng.

Ví dụ 4.43

Sử dụng giải thuật Lloyds để tối ưu các tham số lượng tử ở Ví dụ 4.42.

Giải: Với giải thuật Lloyds, chúng ta sử dụng hàm **lloyds()** với chương trình Matlab như sau:

```
% Thời gian của tín hiệu, chọn 1 chu kỳ với một bước mẫu là 0.1
t = [0:1/2*pi];
% Tạo tín hiệu cos
sig = sin(t);

% Chia vùng
partition = [-1:.2:1];

% Từ mã
codebook = [-1.2:.2:1];

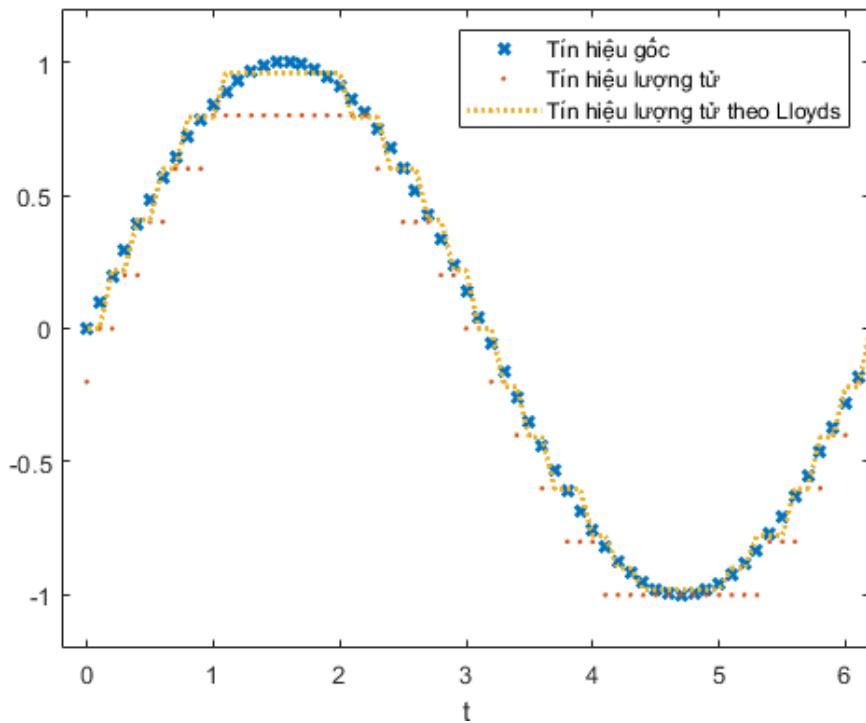
% Lượng tử
[index,quants,distor] = quantiz(sig,partition,codebook);

% Vẽ tín hiệu trước và sau khi lượng tử
plot(t,sig,'x',t,quants,'.','linewidth',2)

% Tối ưu dùng Lloyds
[partition,codebook] = lloyds(sig, codebook)
```

```
[index,quants,distor] = quantiz(sig,partition,codebook);

% Vẽ trên cùng đồ thị
hold on
% Vẽ tín hiệu trước và sau khi lượng tử
plot(t,quants,'-','linewidth',2)
% Chú thích cho hình
legend('Tín hiệu gốc','Tín hiệu lượng tử','Tín hiệu lượng tử dùng Lloyds');
% Điều chỉnh vùng vẽ theo trục x và trục y của hình
axis([-2 max(t) -1.2 1.2])
set(gcf,'color','white');
xlabel('t');
```



Hình 4.25: Đồ thị so sánh hiệu quả của phương pháp lượng tử Lloyds.

Chúng ta thấy rằng giải thuật Lloyds cho tín hiệu sau lượng tử ít méo dạng hơn rất nhiều so với quá trình lượng tử truyền thống.

Tín hiệu trong Hình 4.25 có tổng cộng 63 mẫu trong khoảng từ 0 đến 2π . Với mỗi mẫu, chúng ta lượng tử với 12 vùng, nghĩa là dùng 4 bit cho mỗi mẫu. Do đó, tổng số bit chúng ta cần dùng để truyền tín hiệu cos trong 2 chu kỳ là 252 bit. Để giảm tổng số bit cần truyền, thay vì truyền đầy đủ 4 bit cho từng mẫu, chúng ta chỉ truyền độ lệch giữa mẫu hiện tại và mẫu trước đó, đó là khái niệm của DPCM. Xem thêm Ví dụ 4.44 về mã DPCM.

Ví dụ 4.44

Hãy mã hóa và giải mã DPCM tín hiệu răng cưa dùng hàm sawtooth() trong 3 chu kỳ. Vẽ và so sánh với tín hiệu gốc. Nhận xét về nhiễu trung bình bình phương.

Giải: Chúng ta chọn bộ dự đoán (predictor) trong mã DPCM là $y(k) = x(k - 1)$.

```
% y(k)=x(k-1)
predictor = [0 1];
partition = [-1:.1:.9];
codebook = [-1:.1:1];
t = [0:pi/50:2*pi];

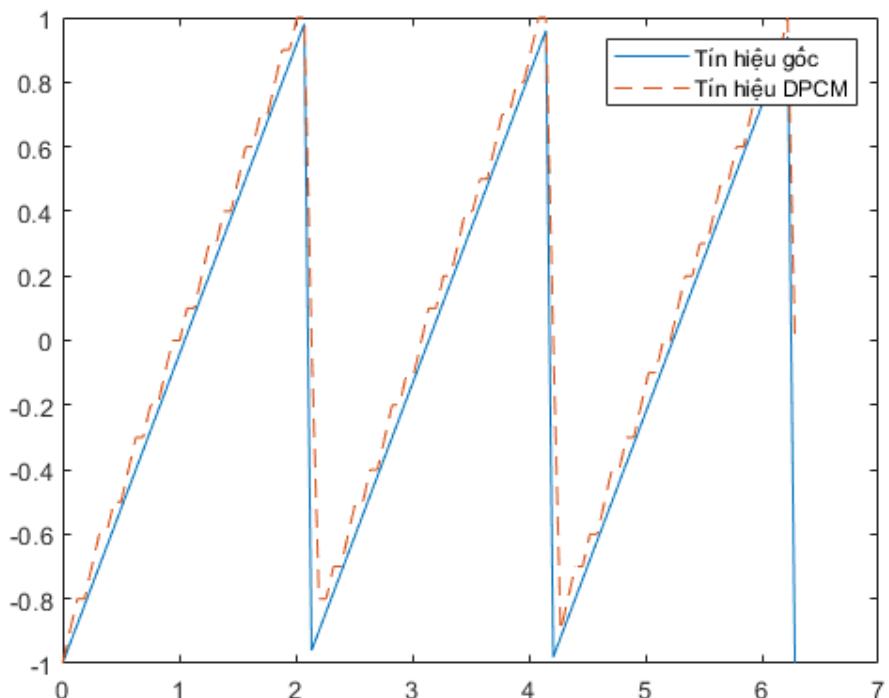
% Tạo tín hiệu gốc
x = sawtooth(3*t);

% Lượng tử DPCM
encodedx = dpcmenco(x,codebook,partition,predictor);

% Phục hồi tín hiệu lượng tử DPCM
decodedx = dpcmdeco(encodedx,codebook,predictor);

% Vẽ hình so sánh
plot(t,x,t,decodedx,'--')
legend('Tín hiệu gốc','Tín hiệu DPCM ','Location','NorthOutside');
set(gcf,'color','white');

% Sai số của lỗi trung bình bình phương
distor = sum((x-decodedx).^2)/length(x)
```



Hình 4.26 : Tín hiệu gốc và tín hiệu mã hóa DPCM.

Mỗi mẫu chúng ta dùng 20 phân vùng, nghĩa là 5 bit cho mỗi mẫu. Tổng cộng số mẫu ta dùng là 101 mẫu, nên tổng số bit dùng để lưu trữ tín hiệu là 505 bit. Với mã hóa DPCM, ta chỉ dùng 202 bit. Độ méo dạng bằng 0.0327 là không lớn.

Sau khi lấy mẫu tín hiệu, chúng ta thực hiện nén tín hiệu. Matlab hỗ trợ chúng ta các giải thuật nén như trong Bảng 4.14.

Bảng 4.14: Các hàm Matlab hỗ trợ mã hóa nguồn.

	Hàm Matlab	Chú thích
Mã hóa số học	arithenco()	Mã hóa số học
	arithdeco()	Giải mã số học
Mã hóa Huffman	huffmanenco	Mã hóa Huffman
	huffmandeco	Giải mã Huffman
	huffmandict	Trả về từ điển Huffman

Mã hóa Huffman là một thuật toán mã hóa dùng để nén dữ liệu, dựa trên bảng tần suất xuất hiện các ký tự cần mã hóa để xây dựng một bộ mã nhị phân cho các ký tự đó sao cho dung lượng (số bit) sau khi mã hóa là nhỏ nhất. Để hiểu hơn về mã hóa Huffman, xem Ví dụ 4.45.

Ví dụ 4.45

Cho nguồn tín hiệu s có 10 ký tự là biến ngẫu nhiên phân bố đều từ 1 đến 6. Xác suất xuất hiện của ký tự 1 đến 6 lần lượt là [.5 .125 .125 .125 .0625 .0625].

- Hãy tính số lượng bit cần thiết để truyền 10 ký tự khi chưa thực hiện mã hóa nguồn.
- Hãy tính entropy của nguồn.
- Hãy mã hóa nguồn theo mã hóa Huffman và cho biết số bit cần thiết để truyền.
- Nhận xét.

Giải: Chúng ta có 6 loại ký tự nên phải dùng 3 bit để lưu trữ ký tự, do đó số bit cần thiết để truyền 10 ký tự sẽ là $3 \times 10 = 30$ bit. Entropy của nguồn theo định nghĩa là:

$$\sum_{k=1}^6 p_k \log_2 \frac{1}{p_k} = 2.125 \quad (4.88)$$

bit/ký tự. Do đó, số bit tối thiểu để truyền 10 ký tự là: 21,25 ký tự.

```
% Mã hóa Huffman
dict = huffmandict(1:6, [.5 .125 .125 .125 .0625 .0625]);
S = randi([1 6], 1, 10)
enS = huffmanenco(S, dict)
deS = huffmandeco(enS, dict)
S =
4   3   1   1   1   1   5   6   5   5
enS =
Columns 1 through 19
1   1   0   1   1   1   0   0   0   0   1   0   1   1   1   0   1   0   1
```

Columns 20 through 26

0 1 1 1 0 1 1

deS =

4 3 1 1 1 5 6 5 5

Nhận xét: Nếu dùng mã hóa Huffman, thì chúng ta sẽ chỉ cần truyền 26 bit trong khi entropy của nguồn tín hiệu là 21,25 bit. Khác với mã hóa Huffman, mã hóa số học sẽ cho hiệu năng tốt hơn khi xử lý với nguồn vào dài và phân bố bị lệch. Hãy xem Ví dụ 4.46 sau đây.

Ví dụ 4.46

Xem xét một nguồn tín hiệu từ cảm biến nhiệt độ, dữ liệu trả về trong 1 ngày khoảng 1000 lần với chỉ hai trạng thái là nhiệt độ bình thường ký hiệu là mức 1 và quá nhiệt là mức 2, 99% dữ liệu đo đạc trong ngày là ở mức 1. Hãy thực hiện mô phỏng quá trình truyền dữ liệu sử dụng mã hóa số học.

Giải: Nếu không thực hiện mã hóa, do có 2 mức và 1000 bảng ghi dữ liệu, nên số bit cần thiết để truyền là 1000 bit và không phụ thuộc vào đặc tính của dữ liệu.

Chương trình Matlab sau đây mô phỏng mã hóa số học. Kết quả trả về là 89 bit so với 1000 bit nghĩa là quá trình nén dữ liệu rất hiệu quả, đặc biệt trong trường hợp dữ liệu có phân bố bị lệch.

```
% Đặc tính dữ liệu  
counts = [99 1];  
  
% Chiều dài dữ liệu  
N = 1000;  
  
% Dữ liệu nhiệt độ theo phân bố cho trước  
sig = randsrc(1,N,[1 2;counts/100]);  
  
% Mã hóa số học  
sig_en = arithenco(sig,counts);  
  
% Chiều dài dữ liệu sau mã hóa  
sig_len = size(sig_en)  
sig_len =
```

1 89

4.3.2 Mã phát hiện và sửa lỗi

Mã hóa phát hiện lỗi và sửa lỗi là chức năng quan trọng trong hệ thống thông tin số. Matlab hỗ trợ 04 hàm cho mã CRC, 28 hàm cho mã khối, 15 hàm cho mã xoắn và 32 hàm cho trường Galois.

4.3.2.1 Mã CRC

Với mã CRC, Matlab không hỗ trợ hàm độc lập mà hỗ trợ qua đối tượng **comm** và chúng ta thực hiện gọi hàm thông qua đối tượng comm. Bảng 4.15 trình bày 4 hàm của mã CRC.

Bảng 4.15: Hàm cho mã CRC.

TT	Hàm	Chú thích hàm
1	comm.CRCDetector	Phát hiện lỗi trong dữ liệu vào dùng CRC
2	comm.CRCGenerator	Tạo ra mã CRC và nhúng vào dữ liệu
3	comm.HDLCRCDetector	Phát hiện lỗi trong dữ liệu vào dùng CRC
4	comm.HDLCRCGenerator	Tạo ra mã CRC và nhúng vào dữ liệu

Ví dụ 4.47

Hãy tạo một chuỗi bit toàn 1 có chiều dài 20

- Chèn mã CRC có đa thức là $x^4 + x^2 + x + 1$ vào chuỗi bit
- Kiểm tra giải thuật chèn bao nhiêu bit và chèn ở đầu hay cuối chuỗi.
- Thay đổi nội dung của hai bit đầu tiên và thực hiện kiểm tra mã CRC.

Giải: Chúng ta dùng hàm ones() để tạo chuỗi bit toàn 1. Số lượng bit chèn vào là 4 và mã CRC được chèn vào cuối chuỗi bit.

```
% Chiều dài chuỗi bit
N = 20;

% Tạo tín hiệu vào toàn 1
x = ones(N,1);% Khởi tạo đối tượng CRC
h = crc.generator('Polynomial', [1 0 1 1 1], 'InitialState', [0 0 0 0], 'FinalXOR', [1 1 1 1]);

% Tạo dữ liệu chèn với mã CRC
x_crc = generate(h, x);

% Chèn lỗi vào tín hiệu nhận
x_crc(1:2) = ~x_crc(1:2);

% Khởi tạo bộ kiểm tra CRC
h = crc.detector('Polynomial', [1 0 1 1 1], 'InitialState', [0 0 0 0], 'FinalXOR', [1 1 1 1]);

% Kiểm tra mã_CRC phía thu
[x_, error] = detect(h, x_crc);

% Kiểm tra có lỗi hay không?
error

% Tính số lỗi
sum(x~=x_)
error =
logical
```

```
ans =
2
```

4.3.2.2 Mã khói

Mã khói là mã sửa lỗi không nhó. Matlab hỗ trợ mã khói ở ba dạng: hàm Matlab, khói Simulink và trong đối tượng comm. Trong phần này, chúng ta chỉ tập trung vào các hàm mã khói trong Bảng 4.16.

Bảng 4.16: Các hàm Matlab cho mã khói tuyến tính.

TT	Hàm Matlab	Thuyết minh
1	cyclgen	Tạo ra ma trận sinh và ma trận kiểm tra chẵn lẻ cho mã vòng
2	cyclpoly	Tạo ra đa thức sinh cho mã vòng
3	decode	Bộ giải mã khói
4	encode	Bộ mã hóa khói
5	gfweight	Tính khoảng cách tối thiểu của mã khói tuyến tính
6	gen2par	Chuyển đổi giữa ma trận sinh và ma trận kiểm tra chẵn lẻ
7	hammgen	Tạo ra ma trận sinh và ma trận kiểm tra chẵn lẻ cho mã Hamming
8	syndtable	Tạo ra bảng giải mã syndrome

Ví dụ 4.48

Thực hiện mã hóa và giải mã với mã Hamming $(n, k) = (15, 11)$. Hãy xem xét hai trường hợp sau:

- Trường hợp 1: bit nhận thứ 3 bị sai.
- Trường hợp 2: bit nhận thứ 1 và 2 bị sai.

Giải: Để làm bit sai, chúng ta thực hiện phép đảo bit tại vị trí bị sai.

```
% Thiết lập mã Hamming
k = 11;
n = 15;
data = randi([0 1],k,1);
TxData = encode(data,n,k,'hamming/binary');

%% Trường hợp 1
% Sai 1 bit tại bit thứ 3
RxData = TxData;
RxData(3) = ~RxData(3);
decData = decode(RxData,n,k,'hamming/binary');
numerr = sum(data~=decData)

%% Trường hợp 2
% Sai hai bit 1 và 2
RxData = TxData;
RxData(1:2) = ~RxData(1:2);
decData = decode(RxData,n,k,'hamming/binary');
```

```

numerr = sum(data~=decData)

numerr =
    0

numerr =
    1

```

Ta nhận thấy rằng số lỗi trong Trường hợp 1 bằng 0 và số lỗi trong Trường hợp 2 bằng 1, nghĩa là mã Hamming $(n,k) = (15,11)$ chỉ sửa được 1 lỗi.

Ví dụ 4.49

Thực hiện mã hóa và giải mã với mã vòng $(n,k) = (15,5)$. Hãy xem xét hai trường hợp:

- Trường hợp 1: sai 4 bit.
- Trường hợp 2: sai 6 bit.

Giải: Trường hợp 1, chúng ta đảo 4 bit ở vị trí 4 đến 7. Trường hợp 2, chúng ta đảo 6 bit từ vị trí 1 đến 6.

```

% Mã vòng (n,k) = (15,5)
n = 15;
k = 5;

% Khởi tạo dữ liệu
data = randi([0 1],k,1);
% Khi tạo đa thức vòng
gpol = cyclpoly(n,k);
% Tạo ra ma trận sinh
parmat = cyclgen(n,gpol);
% Tạo ra bảng giải mã syndrome
trt = syndtable(parmat);
% Mã hóa vòng
TxData = encode(data,n,k,'cyclic/binary',gpol);

% Trường hợp 1: sai 4 bit từ 4 đến 7
RxData = TxData;
RxData(4:7) = ~RxData(4:7);
decData = decode(RxData,n,k,'cyclic/binary',gpol,trt);
numerr = biterr(data,decData)

% Trường hợp 2: sai 6 bit từ 1 đến 6
RxData = TxData;
RxData(1:6) = ~RxData(1:6);
decData = decode(RxData,n,k,'cyclic/binary',gpol,trt);
numerr = biterr(data,decData)

```

Trường hợp 1 không có lỗi và Trường hợp 2 có 1 lỗi, nghĩa là mã vòng $(n, k) = (15, 5)$ có thể sửa đến 5 lỗi.

Với mã BCH, Matlab hỗ trợ 4 hàm để thực hiện mã hóa và giải mã như trong Bảng 4.17.

Bảng 4.17: Các hàm Matlab cho mã BCH.

TT	Hàm Matlab	Thuyết minh
1	bchenc	Mã hóa BCH
2	bchdec	Giải mã BCH
3	bchgenpoly	Hàm sinh đa thức của mã BCH
4	bchnumerr	Trả về số lượng lỗi có thể sửa được của mã BCH

Ví dụ 4.50

Thực hiện mã hóa và giải mã với mã BCH $(n, k) = (15, 5)$

- a. Kiểm tra khả năng sửa lỗi của mã BCH.
- b. Trường hợp 1: số lượng bit sai nhỏ hơn hoặc bằng khả năng sửa lỗi của mã.
- c. Trường hợp 2: số lượng bit sai lớn hơn khả năng sửa lỗi của mã.

Giải: Dùng hàm **bchnumerr()** để kiểm tra số lượng lỗi mà mã có thể sửa được.

```

M = 4;

% Chiều dài từ mã
n = 2^M-1;

% Chiều dài bản tin
k = 5;
% Số lượng từ mã để mã hóa
nwords = 10;

% Chuyển sang trường GF
msgTx = gf(randi([0 1],nwords,k));

% Mã hóa
enc = bchenc(msgTx,n,k);

% Kiểm tra khả năng sửa lỗi của mã, giá trị ne là giá trị số lỗi mà mã BCH có khả năng sửa được
ne = bchnumerr(n,k);

%% Trường hợp 1: Số lỗi sai bằng khả năng sửa lỗi
noisycode1 = enc + randerr(nwords,n,1:ne);
[msgRx1,numerr1] = bchdec(noisycode1,n,k);
% Các vị trí trên vector numerr1 bằng 1 là bit sai
sum(numerr1==1)

%% Trường hợp 2: Số lỗi sai hơn khả năng sửa lỗi
noisycode2 = enc + randerr(nwords,n,1:ne+1);
[msgRx2,numerr2] = bchdec(noisycode2,n,k);
sum(numerr2==1)

```

Bảng 4.18: Các hàm Matlab cho mã Reed-Solomon.

TT	Hàm Matlab	Thuyết minh
1	rsenc()	Bộ mã hóa Reed-Solomon
2	rsdec()	Bộ giải mã Reed-Solomon
3	rsgenpoly()	Bộ tạo đa thức sinh của mã Reed-Solomon
4	rsgenpolycoeffs()	Hệ số ma trận sinh cho mã Reed-Solomon

Ví dụ 4.51

Mã hóa và giải mã RS $(n,k) = (7,3)$ với chuỗi 10 ký tự. Xem xét ba trường hợp với 1, 2 và 3 lỗi ký tự.

Giải: Mã hóa và giải mã RS làm việc trên trường Galois, nên chúng ta phải chuyển dữ liệu mã hóa vào trường Galois bằng hàm **gf()**.

```
% Số lượng bit trên 1 symbol
m = 3;
% Chiều dài codeword
n = 2^m-1;
% Mã RS (k,n) = (7,3)
k = 3;

% Mã RS xử lý trên trường Galois
N = 10;
msg = gf(randi([1 n],N,k),m);
Txrscode = rsenc(msg,n,k);

% Tạo ma trận lỗi có cùng kích thước với ma trận Txrscod
errors = zeros(size(Txrscod));
% Dòng 1: 1 lỗi
errors(1,1) = 1;
% Dòng 2: 2 lỗi
errors(2,1:2) = 1;
% Dòng 3: 3 lỗi
errors(3,1:3) = 1;
errors = gf(errors,m);
% Cộng nhiễu tại máy thu
Rxrscode = Txrscode + errors;
% Giải mã RS
[rxcode,cnumerr] = rsdec(Rxrscode,n,k);
cnumerr'
sum(cnumerr== -1)

ans =
1   2   -1   0   0   0   0   0   0
```

ans =

1

Từ giá trị của vector cnumerr, ta thấy dòng thứ 3 có lỗi khi có giá trị -1.

Matlab hỗ trợ mã LDPC và mã Turbo thông qua ba lệnh trong Bảng 4.19.

Bảng 4.19: Các hàm Matlab cho mã LDPC và mã Turbo.

TT	Hàm Matlab	Thuyết minh
1	dvbs2ldpc()	Mã LDPC cho tiêu chuẩn DVB-S.2.
2	tpcenc()	Bộ mã hóa mã Turbo.
3	tpcdec()	Bộ giải mã mã Turbo.

4.3.2.3 Mã xoắn

Mã xoắn có nhiều ứng dụng trong các hệ thống thông tin vô tuyến như GSM, EDGE, 3G và mạng thông tin vệ tinh. Matlab hỗ trợ mã xoắn với 7 hàm cho trong Bảng 4.20.

Bảng 4.20: Danh mục các hàm mã xoắn của Matlab.

TT	Hàm Matlab	Thuyết minh
1	convenc()	Mã hóa mã xoắn
2	distspec()	Tính khoảng cách phô của mã xoắn
3	iscatastrophic()	Kiểm tra cấu trúc trellis có phù hợp với mã xoắn catastrophic
4	istrellis()	Kiểm tra cấu trúc trellis
5	oct2dec()	Chuyển số hệ 10 sang hệ 8
6	poly2trellis()	Chuyển đa thức mã xoắn sang cấu trúc trellis
7	vitdec()	Giải mã mã xoắn dùng giải thuật Viterbi

Ví dụ 4.52

Mô phỏng mã hóa và giải mã mã xoắn có các tham số lần lượt: constraintLength = [4 3], codeGenerator = [4 5 17;7 4 2] và tracebackLen = 2.

Giải: Chúng ta lần lượt sử dụng các hàm poly2trellis(), randi(), convenc() và vitdec().

```
% Khởi tạo mã xoắn
constraintLength = [4 3];
codeGenerator = [4 5 17;7 4 2];
trellis = poly2trellis(constraintLength,codeGenerator);
tracebackLen = 2;

% Tạo tín hiệu ngẫu nhiên đầu vào
N = 10;
x = randi([0 1],N,1);

% Mã hóa
Txcode = convenc(x,trellis);

% Giải mã
x_ = vitdec(Txcode,trellis,tracebackLen,'trunc','hard');
% Tổng số lỗi
sum(x~=x_)
```

Bảng 4.21: Các hàm hỗ trợ mã xoắn và mã Turbo theo đối tượng comm.

TT	Hàm Matlab	Mô tả hàm
1	comm.APPDecoder	Giải mã mã xoắn dùng phương pháp xác suất hậu nghiệm
2	comm.ConvolutionalEncoder	Mã hóa mã xoắn
3	comm.gpu.ConvolutionalEncoder	Mã hóa mã xoắn với GPU
4	comm.TurboDecoder	Giải mã tín hiệu dùng sơ đồ giải mã ghép nối song song
5	comm.gpu.TurboDecoder	Giải mã tín hiệu dùng sơ đồ giải mã ghép nối với GPU
6	comm.TurboEncoder	Mã hóa tín hiệu dùng sơ đồ mã hóa ghép nối song song
7	comm.ViterbiDecoder	Giải mã mã chập dùng giải thuật Viterbi
8	comm.gpu.ViterbiDecoder	Giải mã mã chập dùng giải thuật Viterbi với GPU

4.4 Điều chế và giải điều chế

Có hai cách trong Matlab để thực hiện mô phỏng quá trình điều chế và giải điều chế: mô phỏng bằng gốc và mô phỏng bằng dải.

Mô phỏng bằng gốc còn được gọi là phương pháp mô phỏng tương đương thông thấp, có ưu điểm là độ phức tạp thấp. Toolbox Communications của Matlab hỗ trợ mô phỏng bằng gốc cho điều chế số và mô phỏng bằng dải cho điều chế tương tự.

Mô phỏng bằng gốc có tương đương với mô phỏng bằng dải hay không? Để trả lời câu hỏi này chúng ta cần tìm hiểu mô hình toán của tín hiệu mô phỏng bằng gốc. Gọi x là tín hiệu số cần mô phỏng, $y(t)$ là tín hiệu ở ngõ ra của bộ điều chế bằng dải, ta có

$$y(t) = I(t)\cos(2\pi f_c t + \theta) - Q(t)\sin(2\pi f_c t + \theta) \quad (4.89)$$

với f_c là tần số sóng mang và θ là góc pha ban đầu của sóng mang. Tùy theo loại điều chế mà các tham số trong (4.89) như $I(t)$, $Q(t)$, f_c , θ , sẽ thay đổi theo x . Trong $y(t)$ ở (4.89), $I(t)$ là tín hiệu đồng pha và $Q(t)$ là tín hiệu vuông pha.

Sử dụng công thức Euler, ta có thể biểu diễn $y(t)$ ở dạng phần thực của một số phức như sau:

$$y(t) = \operatorname{Re} \left\{ B(t) e^{j2\pi f_c t} \right\} \quad (4.90)$$

với

$$B(t) = [I(t) + jQ(t)] e^{j\theta} \quad (4.91)$$

là tín hiệu tương đương bằng gốc và $\operatorname{Re}\{\cdot\}$ là toán tử lấy phần thực.

So sánh $y(t)$ ở (4.90) và $B(t)$ ở (4.91), chúng ta sẽ thấy khác biệt ở thành phần, $e^{j2\pi f_c t}$, phụ thuộc vào tần số sóng mang f_c . Thành phần này sẽ không chịu ảnh hưởng của suy hao kênh truyền và nhiễu, nên nếu chúng ta loại bỏ ra khỏi tín hiệu mô phỏng sẽ giảm rất nhiều khói lượng tính toán và bộ nhớ khi mà tần số sóng mang thường cao hơn rất nhiều so với tần số tín hiệu và dẫn đến tần số lấy mẫu phải lớn hơn ít nhất gấp đôi lần tần số sóng mang theo định lý Nyquist. Hình 4.27 trình bày mô hình điều chế/giải điều chế bằng gốc với kênh truyền.



Hình 4.27: Mô hình điều chế bằng gốc.

4.4.1 Điều chế tương tự

Matlab hỗ trợ cả điều chế tương tự bằng gốc và bằng dải.

4.4.1.1 Điều chế tương tự bằng gốc

Matlab chỉ hỗ trợ điều chế FM tương tự bằng gốc với các hàm liệt kê ở Bảng 4.22.

Bảng 4.22: Hàm điều chế tương tự FM bằng gốc.

Loại điều chế	Hàm điều chế	Hàm giải điều chế	Chú thích
FM	comm.FMBroadcastModulator	comm.FMBroadcastDemodulator	Điều chế FM dạng quảng bá
FM	comm.FMModulator	comm.FMDemodulator	Điều chế FM

Các hàm trên sẽ tạo ra một đối tượng điều chế FM, chứ không trả về một giá trị. Xem cách sử dụng hàm thông qua Ví dụ 4.53.

Ví dụ 4.53

Thực hiện điều chế FM dạng bằng gốc với tần số lấy mẫu là 1000 Hz, thời gian lấy mẫu là 0.5 s, độ lệch tần số trong điều chế FM là 50 Hz.

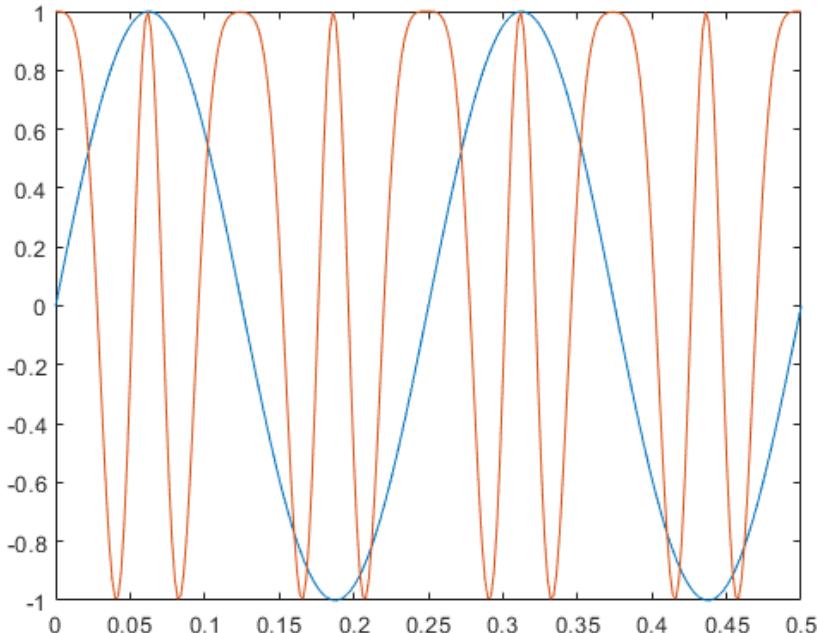
Giải: Một điểm lưu ý khi thực hiện mô phỏng điều chế FM bằng gốc là chúng ta lấy phần thực của tín hiệu thu được sau điều chế khi muốn quan sát tín hiệu. Từ đối tượng tạo ra từ hàm comm.FMModulator(), chúng ta dùng hàm **step()** để thực hiện điều chế.

fs = 500;	% Sample rate (Hz)
ts = 1/fs;	% Sample period (s)
fd = 25;	% Frequency deviation (Hz)

```

t = (0:ts:1/2)';
x = sin(2*pi*4*t);
MOD = comm.FMModulator('SampleRate',fs,'FrequencyDeviation',fd);
y = step(MOD,x);
plot(t,[x real(y)])
set(gcf,'color','white');

```



Hình 4.28: Tín hiệu thực của điều chế FM.

4.4.1.2 Điều chế tương tự băng dài

Với điều chế băng dài, Matlab hỗ trợ điều chế AM, FM, PM và SSB với các hàm cho trong Bảng 4.23.

Bảng 4.23: Các hàm điều chế và giải điều chế tương tự băng dài.

Loại điều chế	Hàm điều chế	Hàm giải điều chế	Chú thích
AM	ammod()	amdemod()	Điều chế/Giải điều chế AM
FM	fmmmod()	fmdemod()	Điều chế/Giải điều chế FM
PM	pmmod()	pmdemod()	Điều chế/Giải điều chế PM
SSB	ssbmod()	ssbdemod()	Điều chế/Giải điều chế SSB

Ví dụ 4.54

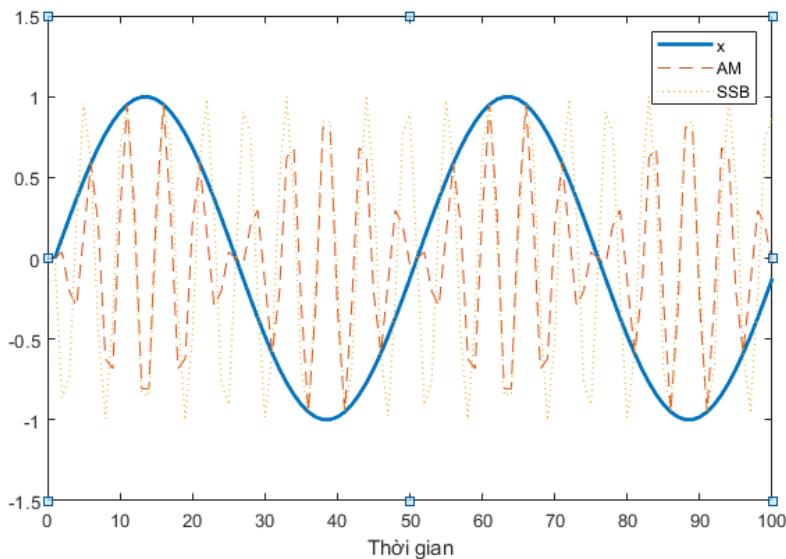
Hãy thực hiện điều chế AM và SSB cho tín hiệu sin có tần số bằng 10 Hz. Vẽ tín hiệu điều chế.

Giải: Với tín hiệu vào có tần số 10 Hz, ta chọn tần số lây mẫu lớn hơn ít nhất 2 lần, nên chúng ta chọn $f_s = 50$ Hz. Để dễ quan sát, chúng ta khoảng thời gian quan sát là 2 giây. Chương trình Matlab là như sau:

```

fs = 50;
t = (0:1/fs:2)';
fc = 10;
x = sin(2*pi*t);
ydouble = ammod(x,fc,fs);
ysingle = ssbmod(x,fc,fs);
plot(x,'linewidth',2);
hold on
plot(ydouble,'--');
hold on
plot(ysingle,:');
legend('x','AM','SSB');
set(gcf,'color','white');

```



Hình 4.29: Tín hiệu điều chế AM và SSB của tín hiệu sin.

Chúng ta dễ dàng nhận thấy rằng, x là đường bao của tín hiệu điều chế AM, nhưng không phải là của tín hiệu SSB. Ví dụ 4.55 tiếp theo sẽ vẽ phổ của hai tín hiệu điều chế AM và SSB để giúp chúng ta thấy được sự khác biệt của hai loại điều chế.

Ví dụ 4.55

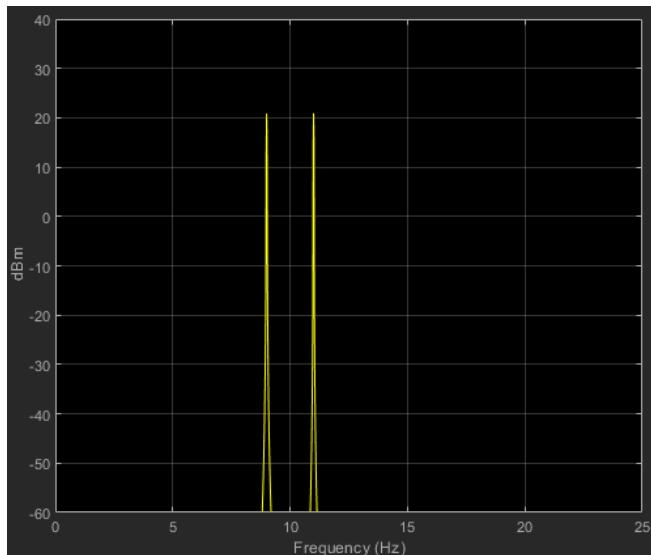
Hãy thực hiện điều chế AM và SSB cho tín hiệu sin có tần số 10 Hz. Vẽ phổ tín hiệu điều chế.

Giải: Để vẽ phổ tín hiệu, chúng ta cần số lượng mẫu lớn. Do đó chúng ta chọn thời gian là 100 giây. Cách tạo tín hiệu và tín hiệu điều chế là tương tự như Ví dụ 4.54, chúng ta chỉ cần bổ sung thêm 3 dòng lệnh để vẽ phổ như sau.

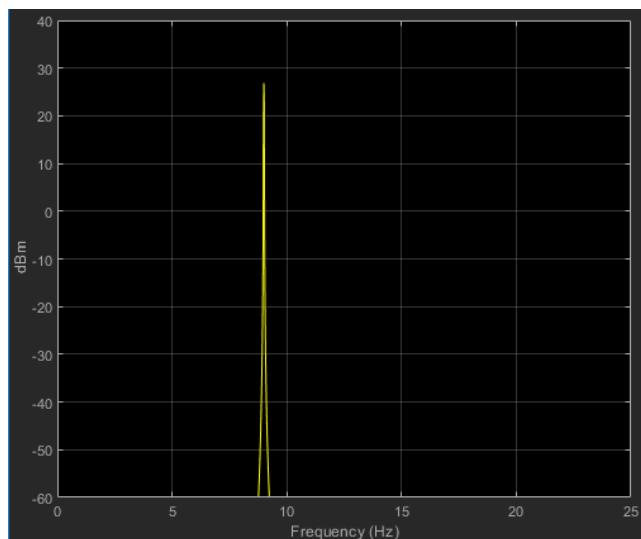
```

sa = dsp.SpectrumAnalyzer('SampleRate',fs, ...
    'PlotAsTwoSidedSpectrum',false, ...
    'YLimits',[-60 40]);
step(sa,ydouble)
step(sa,ysingle)

```



Hình 4.30: Phổ của tín hiệu điều chế AM.



Hình 4.31: Phổ của tín hiệu điều chế SSB.

Có một cách khác để thực hiện điều chế tương tự bằng dải là sử dụng các hàm modulate() và demod() của Toolbox Signal Processing ở mục Waveform Generation: Time Vectors and Sinusoids. Cú pháp của hàm modulate() như sau:

Cú pháp:

$y = \text{modulate}(x, fc, fs)$

$[y, t] = \text{modulate}(x, fc, fs)$

$[\] = \text{modulate}(x, fc, fs, \text{method})$

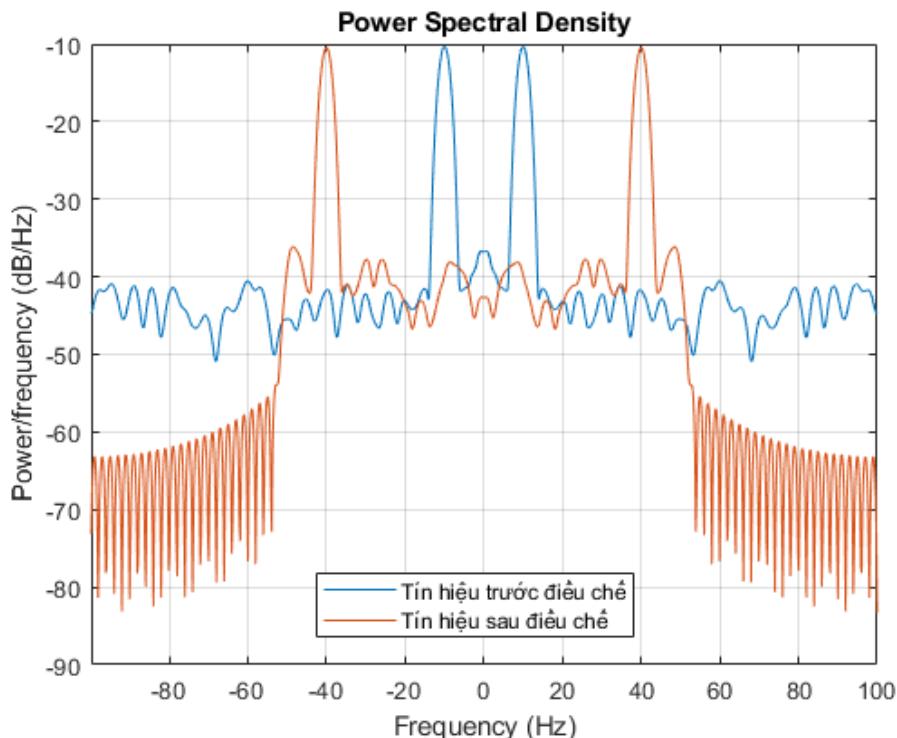
x là tín hiệu điều chế, fc là tần số sóng mang, fs là tần số lấy mẫu, method là phương pháp điều chế.

Ví dụ 4.56

Thực hiện điều chế SSB cho tín hiệu sin có tần số 10 Hz với tần số sóng mang là 50 Hz, lấy mẫu với tần số 200 Hz. Biết rằng tín hiệu này bị nhiễu trắng có công suất (phương sai) là 0.01. Vẽ phô của tín hiệu trước và sau khi điều chế.

Giải: Nhiễu có phương sai là 0.01, nên biên độ nhiễu sẽ là 0.1. Chương trình Matlab như sau. Chúng ta dùng hàm `pwelch()` để vẽ phô tín hiệu.

```
fs = 200;
fc = 50;
t = 0:1/fs:1;
% Tín hiệu sin có tần số 10 Hz và nhiễu có phương sai là 0.01
x = sin(2*pi*10*t) + randn(size(t))/10;
% Điều chế tín hiệu
y = modulate(x,fc,fs,'amssb');
% Vẽ phô tín hiệu
pwelch([x;y]',hamming(100),80,1024,fs,'centered');
legend('Tín hiệu trước điều chế','Tín hiệu sau điều chế');
set(gcf,'color','w');
```



Hình 4.32: Phô của tín hiệu AMSSB trước và sau điều chế.

Ví dụ 4.57

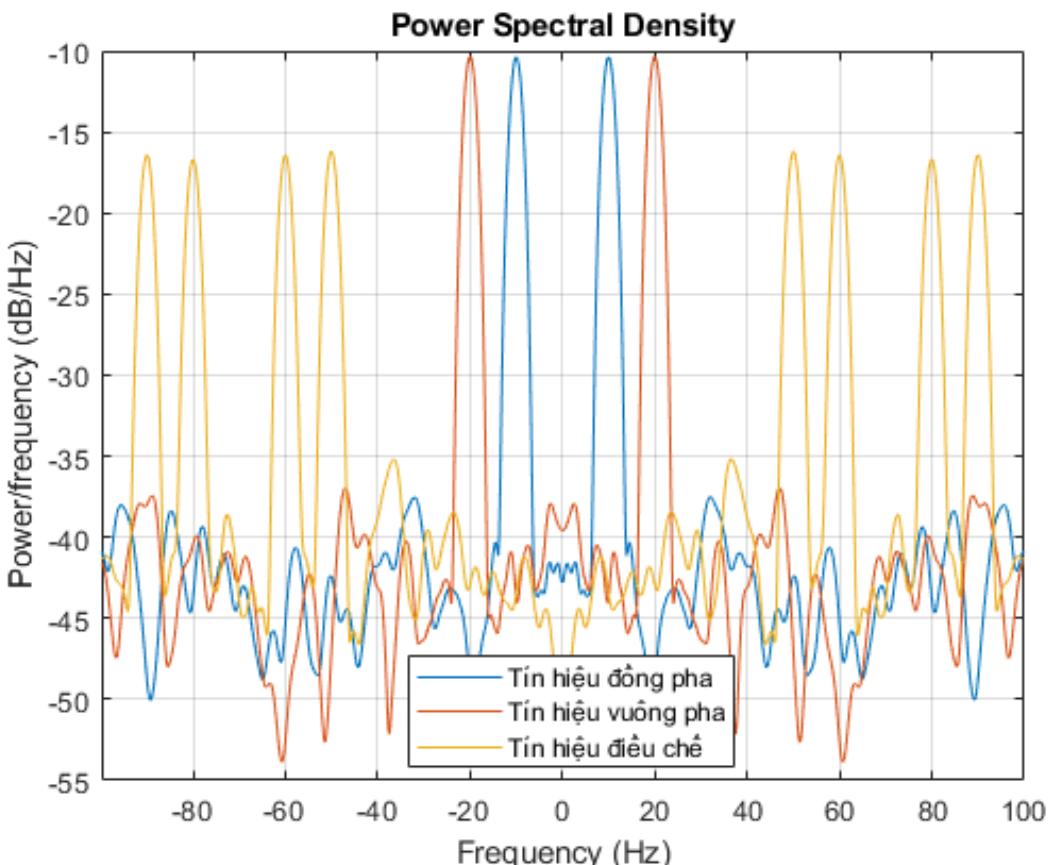
Thực hiện điều chế QAM cho 2 tín hiệu sóng sin có tần số lần lượt là 10 Hz và 20 Hz với tần số sóng mang là 70 Hz và tần số lấy mẫu là 200 Hz. Biết rằng phương sai nhiễu của tín hiệu sóng sin là 0.01. Hãy vẽ phô tín hiệu trước và sau khi điều chế.

Giải: Chúng ta sẽ vẽ tín hiệu trước điều chế bao gồm cả hai thành phần cùng pha và vuông pha.

```

fs = 200;
t = 0:1/fs:1;
% Tao thành phần đồng pha
I = sin(2*pi*10*t) + randn(size(t))/10;
% Tạo thành phần vuông pha
Q = sin(2*pi*20*t) + randn(size(t))/10;
% Điều chế tín hiệu: q được vào ở dạng tham số opt
y = modulate(I,70,fs,'qam',Q);
% Vẽ phổ tín hiệu
pwelch([I;Q;y]',hamming(100),80,1024,fs,'centered');
legend('Tín hiệu đồng pha','Tín hiệu vuông pha','Tín hiệu điều chế','location','best')
set(gcf,'color','w');

```

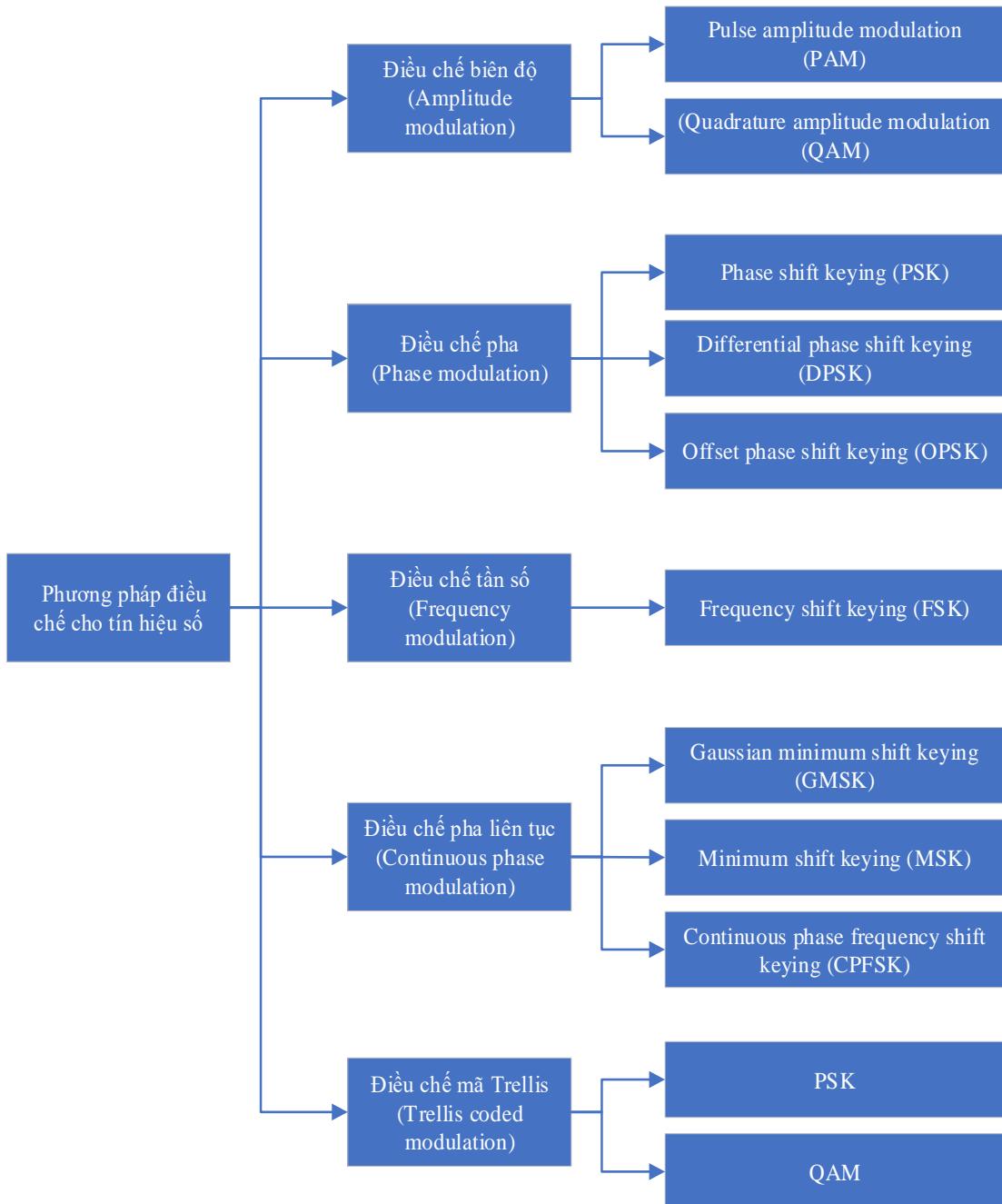


Hình 4.33: Phổ của tín hiệu QAM trước và sau khi điều chế.

4.4.2 Điều chế số

Giống như điều chế tương tự, điều chế số thực hiện thay đổi một hay nhiều tham số của sóng mang theo sự thay đổi của tín hiệu cần điều chế. Điểm khác biệt giữa điều chế tương tự và điều chế số là tín hiệu cần điều chế giới hạn ở một tập hữu hạn, do đó tăng khả năng chống nhiễu của điều chế số khi thực hiện giải điều chế.

Matlab hỗ trợ gần như đầy đủ các loại điều chế số như ở Hình 4.34 và các hàm điều chế xuất ra đường bao phức của tín hiệu điều chế. Một điểm lưu ý là muốn sử dụng các hàm điều chế và giải điều chế, chúng ta cần cài đặt Toolbox Communications.



Hình 4.34: Các phương pháp điều chế cho tín hiệu số.

Toolbox Communications của Matlab hỗ trợ các loại điều chế sau: điều chế biên độ, điều chế pha và biên độ, điều chế pha liên tục, điều chế tần số, điều chế OFDM và điều chế pha. Các hàm điều chế số mà Matlab hỗ trợ liệt kê ở Bảng 4.24.

Bảng 4.24: Các hàm điều chế Matlab hỗ trợ.

Loại điều chế	Hàm điều chế	Hàm giải điều chế	Thuyết minh
Điều chế biên độ (Amplitude Modulation)	genqammod()	genqamdemod()	Điều chế/giải điều chế QAM tổng quát
	pammod()	pamdemod()	Điều chế/giải điều chế PAM
	qammod()	qamdemod()	Điều chế/giải điều chế QAM
	modnorm()		Hàm chuẩn hóa công suất tín hiệu ngõ ra
Điều chế biên độ và pha (Amplitude and Phase Modulation)	apskmod()	apskdemod()	Điều chế/giải điều chế APSK
	dvbapskmod()	dvbapskdemod	Điều chế/giải điều chế APSK tiêu chuẩn DVB-S2/S2X/SH
	mil188qamod()	mil188demod()	Điều chế/giải điều chế MIL-STD-188-110 B/C
Điều chế pha liên tục (Continuous Phase modulation)	mskmod()	mskdemod()	Điều chế/giải điều chế MSK
Điều chế tần số	fskmod()	fskdemod()	Điều chế/giải điều chế FSK
Điều chế OFDM	ofdmmod()	ofdm demod()	Điều chế/giải điều chế OFDM
Điều chế pha (Phase modulation)	dpskmod()	dpskdemod()	Điều chế/giải điều chế DPSK
	pskmod()	pskdemod()	Điều chế/giải điều chế PSK

Ví dụ 4.58

Xem xét điều chế QPSK cho chuỗi dữ liệu [0 1 2 3] và so sánh với hàm pskmod() trong hai trường hợp:

- Không sử dụng mã Gray.
- Có sử dụng mã Gray.
- So sánh kết quả câu a) và b) với kết quả từ hàm pskmod().

Giải: Chúng ta lưu ý sử dụng công thức Euler khi thực hiện điều chế MPSK: $e^{j\theta} = \cos \theta + j \sin \theta$. Chương trình Matlab là như sau:

```

x =[0 1 2 3];
M = 4;

% Không dùng mã Gray
y = exp(1i*(2*pi/M).*x);
% Dùng mã Gray
xG = bin2gray(x,'psk',M);
yG = exp(1i*(2*pi/M).*xG);

% Điều chế với hàm pskmod()
y1 = pskmod(x,M,0);
y2 = pskmod(x,M,0,'gray');

% So sánh

```

```
all(y1 == y)  
all(y2 == yG)
```

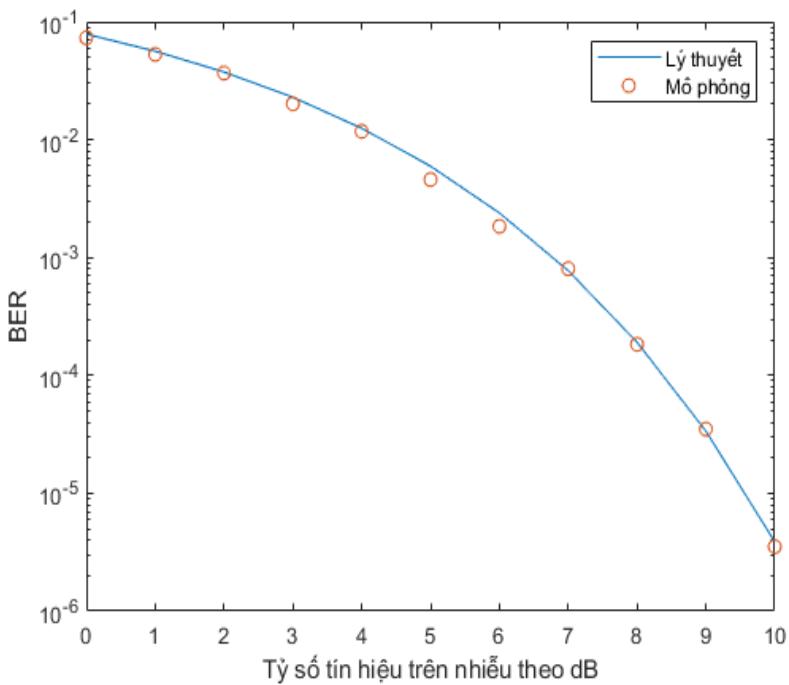
Ví dụ 4.59

Thực hiện điều chế và giải điều chế BPSK trên kênh truyền nhiễu trắng với tỷ số tín hiệu trên nhiễu chạy từ 0 đến 10 dB với bước nhảy là 1 dB. So sánh với công thức lý thuyết.

Giải:

```
EbNo = 0:1:10;  
SNR = 10.^EbNo/10);  
M = 2;  
  
% Lý thuyết  
BER_t = qfunc(sqrt(2*SNR));  
  
% Mô phỏng  
N = ceil(100./BER_t);  
BER_s = zeros(size(EbNo));  
for idx = 1:length(EbNo)  
    Ps = SNR(idx)  
    Pn = 1;  
    m = randi([0 1],1,N(idx));  
    Tx = pskmod(m,M);  
    n = sqrt(Pn/2).*(randn(1,N(idx)) + 1i*randn(1,N(idx)));  
    Rx = sqrt(Ps)*Tx + n;  
    m_ = pskdemod(Rx/Ps,M);  
    BER_s(idx) = BER_s(idx) + sum(m~=m_)/N(idx);  
end  
  
% Vẽ đồ thị  
semilogy(EbNo,BER_t,'.',EbNo,BER_s,'o');  
xlabel('Tỷ số tín hiệu trên nhiễu theo dB');  
ylabel('BER');  
legend('Lý thuyết','Mô phỏng');
```

Kết quả của chương trình Matlab như Hình 4.35.



Hình 4.35: Tỷ lệ lỗi bit ở kênh truyền nhiễu trắng: Mô phỏng và lý thuyết.

Trong các dòng lệnh trên, ta biết rằng tỷ lệ lỗi bit của BPSK trên kênh truyền nhiễu trắng là $Q(\sqrt{2\gamma})$. Tuy nhiên, Matlab có hỗ trợ hàm berawgn() trả về tỷ lệ lỗi bit theo công thức lý thuyết của các kiểu điều chế trên kênh truyền nhiễu trắng.

Cú pháp:

$$\text{BER} = \text{berawgn}(\text{EbNo}, \text{MODTYPE}, M)$$

EbNo là tỷ số của năng lượng bit trên mật độ công suất nhiễu theo dB, MODTYPE là kiểu điều chế muốn đánh giá bao gồm: ‘pam’, ‘qam’, ‘psk’, ‘fsk’, ‘msk’, ‘cpfsk’ và M là số mức điều chế, có giá trị 2, 4, 8, 16,...

Ví dụ 4.60

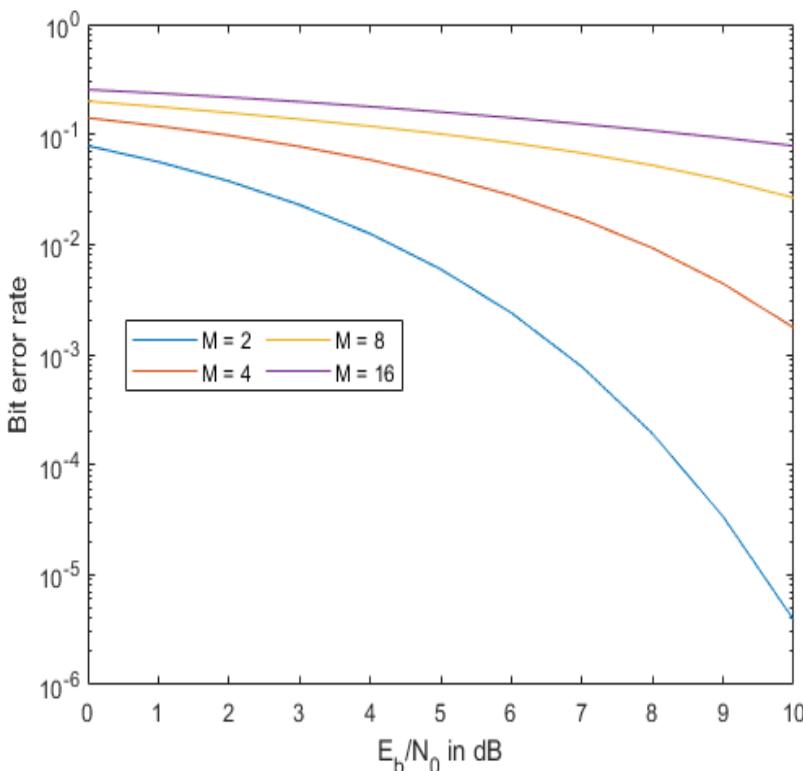
Sử dụng hàm berawgn(), so sánh tỷ lệ lỗi bit của điều chế PAM khi tăng M từ 2 đến 16.

Giải: Chúng ta sử dụng hàm berawgn() như sau.

```
M = [2 4 8 16];
EbNo = 0:10;

for idx = 1:length(M)
    BER(idx,:) = berawgn(EbNo,'pam',M(idx));
end
semilogy(EbNo,BER);
legend('M = 2','M = 4','M = 8','M = 16','Location','best','NumColumns',2);
xlabel('E_b/N_0 theo dB');
ylabel('Tỷ lệ lỗi bit');
```

Kết quả của đoạn mã Matlab trên là đồ thị như ở Hình 4.36.



Hình 4.36: Đồ thị của hàm berawgn() cho điều chế PAM.

Ví dụ 4.61

Hãy thực hiện mô phỏng điều chế MPSK với M cho trước và so sánh kết quả mô phỏng với kết quả lý thuyết từ hàm berawgn(). Vẽ đồ thị tỷ lệ lỗi bit của kết quả mô phỏng và lý thuyết trên cùng đồ thị và so sánh.

Giải: Chúng ta có một số lưu ý sau:

- Hàm beragwn() cung cấp tỷ lệ lỗi bit lý thuyết điều chế MPSK với mã hóa Gray. Do đó, để kết quả mô phỏng trùng khớp với kết quả lý thuyết, chúng ta phải sử dụng điều chế và giải điều chế có mã hóa Gray, cụ thể là `pskmod(m,M,0,'gray')` và `pskdemod(Rx/sqrt(Ps),M,0,'gray')`.
- Khi $M > 2$, chúng ta giả sử công suất nhiễu là 1 thì công suất phát theo symbol sẽ là $\log_2(M)*\text{SNR}(\text{idx})$ với M là mức điều chế và $\text{SNR}(\text{idx})$ là tỷ lệ tín hiệu trên nhiễu tại điểm mô phỏng thứ idx .
- Khi nhiễu trắng bao gồm cả phần thực và phần ảo, biên độ phần thực và phần ảo của nhiễu cần nhân với hệ số $\sqrt{1/2}$.

- Chúng ta sử dụng hàm bitterr() để tính số lượng bit lỗi khi chuỗi dữ liệu nhận được có giá trị từ 0 đến M-1.
- Số lượng bit truyền và nhận sẽ là $\log_2(M) * N(idx)$ với N là số lượng bit truyền tại điểm mô phỏng idx.

```

EbNo = 0:1:10;
SNR = 10.^{(EbNo/10)};
M = 8;

% Lý thuyết
BER_t = berawgn(EbNo,'psk',M,'nondiff');

% Tính toán số lượng bit cần mô phỏng
N = ceil(100./BER_t);

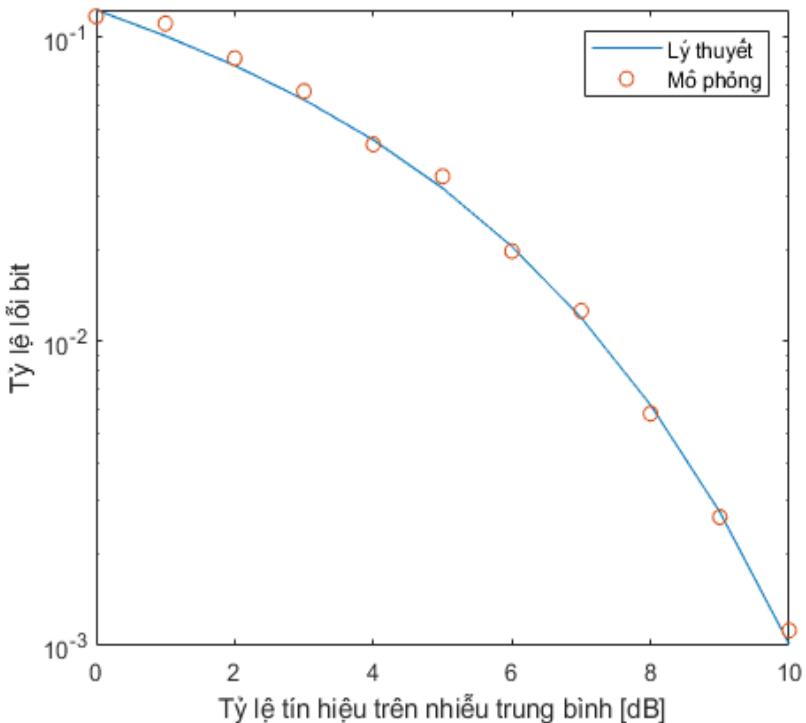
BER_s = zeros(size(EbNo));
for idx = 1:length(EbNo)
    % Công suất tín hiệu
    Ps = log2(M)*SNR(idx);
    % Công suất nhiễu
    Pn = 1;

    % Tx: Máy phát
    m = randi([0 M-1],1,N(idx));
    Tx = pskmod(m,M,0,'gray');

    % Rx: máy thu
    n = sqrt(Pn/2).*(randn(1,N(idx)) + 1i*randn(1,N(idx)));
    Rx = sqrt(Ps)*Tx + n;
    m_ = pskdemod(Rx/sqrt(Ps),M,0,'gray');
    BER_s(idx) = BER_s(idx) + biterr(m,m_)/(log2(M)*N(idx));
end

semilogy(EbNo,BER_t,'-',EbNo,BER_s,'o');
xlabel('Tỷ lệ tín hiệu trên nhiễu trung bình [dB]');
ylabel('Tỷ lệ lỗi bit');
legend('Lý thuyết','Mô phỏng');

```



Hình 4.37: Tỷ lệ lỗi bit lý thuyết và mô phỏng cho điều chế 8-PSK trên kênh truyền nhiễu trắng.

Tiếp theo, chúng ta sẽ thực hiện điều chế và giải điều chế MPAM ở kênh truyền nhiễu trắng ở Ví dụ 4.62.

Ví dụ 4.62

Hãy mô phỏng điều chế và giải điều chế MPAM ở kênh truyền nhiễu trắng trong hai trường hợp có chuẩn hóa và không có chuẩn hóa tín hiệu trước khi phát lên kênh truyền.

Giải: Với điều chế MPAM, chúng ta dùng hàm điều chế **qammod()** và hàm giải điều chế **qamdemod()**. Để tính toán hệ số chuẩn hóa, chúng ta dùng hàm **modnorm()**. Hàm modnorm() thực chất là hàm tính công suất trung bình của M điểm trong biểu đồ chòm sao.

```
% Trường hợp 1: Không chuẩn hóa công suất ra sau bộ điều chế
M = 4;
s = randi([0 M-1],1,100);
```

```
% Điều chế PAM
Tx = pammod(s,M);
% Công suất trung bình của tín hiệu ra
Average_Pow = mean(abs(Tx).^2)
% Giải mã tín hiệu thu bằng tín hiệu nhận
Rx = Tx;

% Giải điều chế
s_ = qamdemod(Rx,M);
% So sánh chuỗi phát và nhận
isequal(s,s_)
```

```

% Trường hợp 2: Có chuẩn hóa
input = 0:M-1;
const = pammod(input,M);
% Hệ số chuẩn hóa Scale
Scale = modnorm(const,'avpow',1);

s = randi([0 M-1],1,100);
% Điều chế và chuẩn hóa công suất phát với hệ số Scale
Tx = Scale * pammod(s,M);
% Tính lại công suất phát xem đã chuẩn hóa chưa?
Average_Pow = mean(abs(Tx).^2)

% Tín hiệu nhận: lưu ý chia lại với hệ số Scale
Rx = Tx/Scale;

% Giải điều chế
Rx = pamdemod(Rx,M);

% So sánh chuỗi phát và nhận
isequal(s,Rx)

```

Chúng ta có các nhận xét sau:

- Công suất trung bình của tín hiệu khi không có chuẩn hóa công suất là 5.32 và khi có chuẩn hóa là 1.
- Khi không có chuẩn hóa công suất, chúng ta sẽ giải điều chế sai mặc dù kênh truyền là không có nhiễu.

Ngoại trừ điều chế MPSK, các điều chế về biên độ đều cần chuẩn hóa công suất trung bình của tín hiệu sau điều chế về một. Quá trình chuẩn hóa cần thực hiện đồng thời ở máy phát và máy thu để đảm bảo cho quá trình điều chế và giải điều chế là đúng.

Ví dụ 4.63

Vẽ biểu đồ chòm sao của điều chế 16-QAM tại máy thu của kênh truyền nhiễu trắng với tỷ lệ tín hiệu trên nhiễu là 20 dB trong hai trường hợp:

- Không có chuẩn hóa công suất.
- Có chuẩn hóa công suất.

Giải: Trong ví dụ này, chúng ta không dùng hàm modnorm(), mà sử dụng tham số “UnitAveragePower” trong hàm qammod() để thực hiện chức năng chuẩn hóa công suất. Chế độ mặc định của hàm qammod() là không chuẩn hóa công suất.

Để vẽ biểu đồ chòm sao, chúng ta dùng hàm scatterplot().

```

% Tạo dữ liệu vào
M = 16;
data = randi([0 M-1],1000,1);
SNR_dB = 20;

```

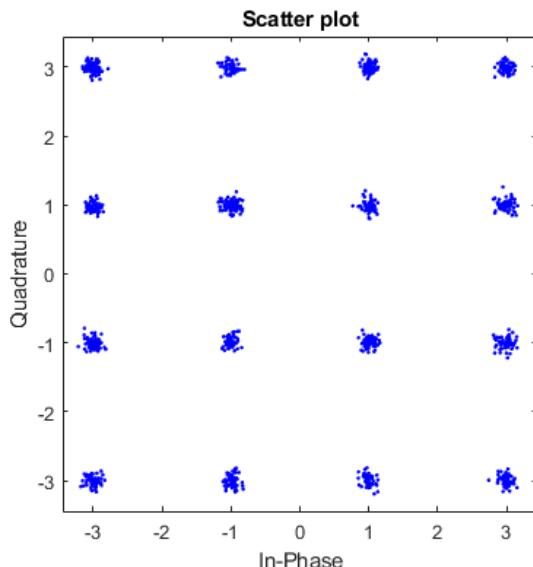
```

%% Trường hợp 1: Không chuẩn hóa công suất
% Điều chỉnh tín hiệu 16-QAM
txSig = qammod(data,M);

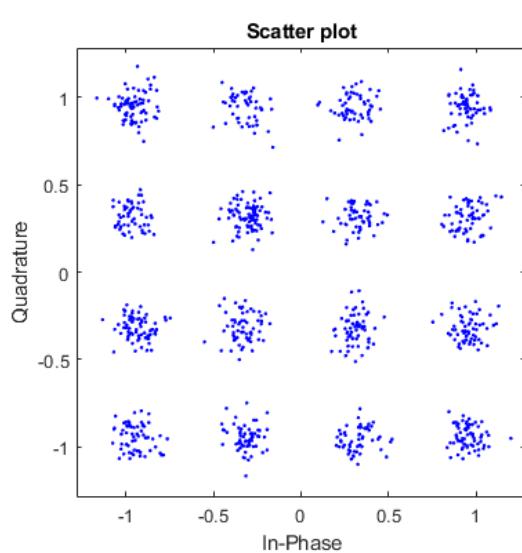
% Cộng nhiễu vào tín hiệu -3dB
rxSig = awgn(txSig,SNR_dB);
scatterplot(rxSig)
set(gcf,'color','w');

%% Trường hợp 2: Chuẩn hóa công suất
txSig_N = qammod(data,16,'UnitAveragePower', true);
rxSig_N = awgn(txSig_N,SNR_dB);
scatterplot(rxSig_N)
set(gcf,'color','w');

```



Hình 4.38: Biểu đồ chòm sao của 16-QAM khi không chuẩn hóa công suất.



Hình 4.39: Biểu đồ chòm sao của 16-QAM khi chuẩn hóa công suất.

Từ Hình 4.38 và Hình 4.39, ta thấy rằng biên độ tín hiệu của hai hình là hoàn toàn khác nhau và mức nhiễu cộng vào tín hiệu cũng khác nhau.

4.5 Quá trình lọc

Matlab cung cấp hàm filter() để thực hiện việc lọc tín hiệu số. Cú pháp của hàm filter() như sau:

Cú pháp: $y = \text{filter}(b,a,x)$

$y = \text{filter}(b,a,x,zi)$

$y = \text{filter}(b,a,x,zi,dim)$

$[y,zf] = \text{filter}(__)$

với x là tín hiệu cần lọc, b và a lần lượt là hệ số của mẫu và tử của hàm truyền; zi là điều kiện bắt đầu cho giá trị trễ của bộ lọc; và dim là số chiều.

Ví dụ 4.64

Tạo ra tín hiệu sin trong hai chu kỳ có biên độ bằng 1, cộng vào tín hiệu nhiễu trắng có biên độ bằng 25% biên độ tín hiệu. Cho tín hiệu qua bộ lọc trung bình động có kích thước cửa sổ bằng 10. Vẽ tín hiệu trước và sau khi cộng nhiễu và lọc.

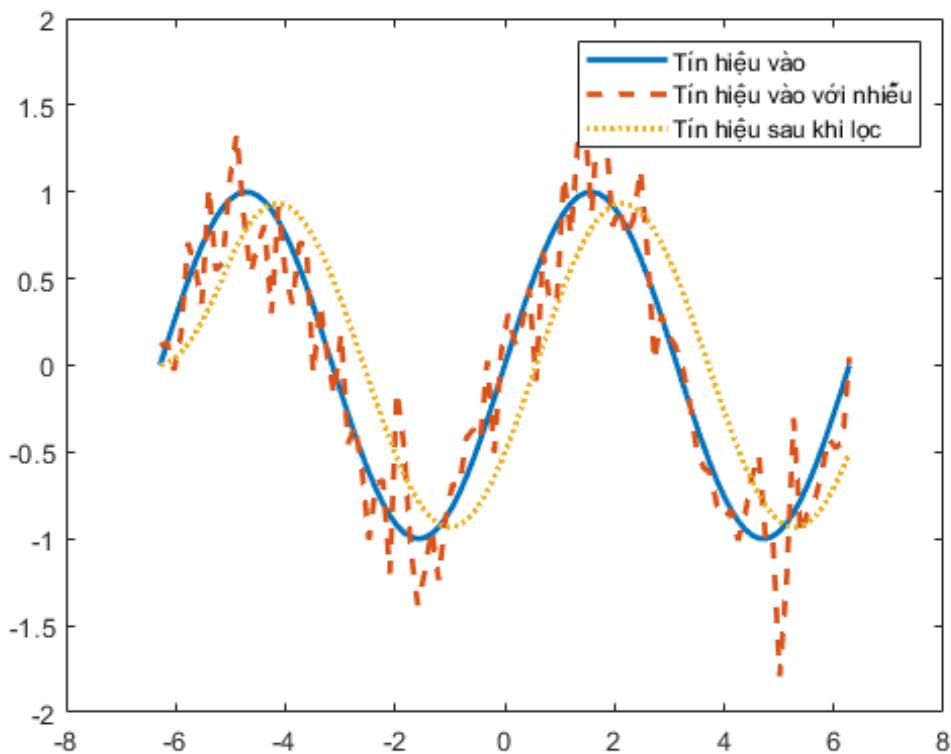
Giải: Kích thước cửa sổ lọc là 10, ta có chương trình Matlab như sau.

```
t = linspace(-2*pi,2*pi,100);
x = sin(t);
n = 0.25*randn(size(t));
s = x + n;

% Filter
windowSize = 10;
b = (1/windowSize)*ones(1,windowSize);
a = 1;

y = filter(b,a,x);

plot(t,x,'-',t,s,'--',t,y,':','LineWidth',2)
legend('Tín hiệu vào','Tín hiệu vào với nhiễu','Tín hiệu sau khi lọc');
set(gcf,'color','w');
```



Hình 4.40: Tín hiệu trước và sau khi lọc nhiễu bằng bộ lọc trung bình động.

4.6 Bài tập

1. Giả sử X_1, X_2, \dots, X_n là các biến ngẫu nhiên độc lập có phân bố đều từ 0 đến 1. Dùng mô phỏng chứng minh $Y_n = n[1 - \max(X_1, X_2, \dots, X_n)]$ sẽ hội tụ theo phân bố đều có tham số 1.
2. Thời gian đến trạm của xe buýt có phân bố đều từ 8:00 đến 8:30. Mô phỏng để tìm xác suất chúng ta đợi từ 5 đến 15 phút, kiểm chứng với lý thuyết.
3. Xem xét một cửa hàng cà phê mang đi, mỗi ngày phục vụ 1000 ly từ 8h sáng đến 9h đêm, thời gian nhận đơn đặt hàng tại quầy của mỗi một khách trung bình là 2 phút. Hãy mô phỏng ước lượng số lượng quầy cần thiết để phục vụ khách.
4. Thực hiện chương trình mô phỏng để chứng minh rằng nếu X và Y là hai biến phân bố mũ độc lập và đồng nhất thì X trong điều kiện $X+Y=t$ có phân bố đều trong khoảng từ $(0, t)$.
5. Cho X và Y là hai biến phân bố mũ độc lập và đồng nhất. Sử dụng mô phỏng để kiểm chứng $\min(X, Y)$ cũng là một biến có phân bố mũ. So sánh với lý thuyết.
6. Sử dụng mô phỏng chứng minh rằng giá trị nhỏ nhất của N biến phân bố mũ độc lập đồng nhất cũng có phân bố mũ.
7. Cho X và Y là hai biến phân bố mũ độc lập và đồng nhất. Sử dụng mô phỏng để kiểm chứng $\max(X, Y)$ có phân bố mũ hay không? Kiểm chứng với lý thuyết.
8. Cho X và Y là hai biến phân bố hàm mũ độc lập và đồng nhất. Sử dụng mô phỏng để kiểm chứng $X+Y$ có phân bố gamma hay không?
9. Kiểm chứng định lý giới hạn trung tâm với biến phân bố Nakagami- m .
10. Kiểm chứng định lý giới hạn trung tâm với phân bố Student.
11. Kiểm chứng định lý giới hạn trung tâm với phân bố Parento.
12. Thời gian giữa hai chuyến xe buýt 150 tại Thành phố Hồ Chí Minh là 30 phút. Hãy mô phỏng để tìm xác suất một người đợi hơn 20 phút mới có xe.
13. Tuổi thọ thiết kế của tủ lạnh là 10 năm. Hãy mô phỏng để tính xác suất tủ lạnh bị hư trong 1, 3 và 5 năm.
14. Thực hiện mô phỏng việc thay xúc sắc 1 triệu lần và lưu kết quả xuống file dưới dạng mã ASCII. Hãy
 - a. Tính dung lượng file lưu nếu không nén.
 - b. Tính entropy của nguồn tín hiệu này.
 - c. Thực hiện nén file dùng winzip và dùng winrar và so sánh tỷ lệ nén với khi không nén. So sánh với entropy của nguồn.

15. Hãy lập trình so sánh mã hóa nguồn A law và μ law cho tín hiệu sin trong 2 chu kỳ.
16. Hãy lập trình hàm có chức năng tương tự như hàm pammod() và hàm pamdemod(). So sánh tốc độ thực thi giữa hàm tự lập trình và hàm của Matlab.
17. Hãy lập trình hàm có chức năng tương tự như hàm qamod() và hàm qamdemod(). So sánh tốc độ thực thi giữa hàm tự lập trình và hàm của Matlab.
18. Hãy lập trình hàm có chức năng tương tự như hàm pskmod() và hàm pskdemod(). So sánh tốc độ thực thi giữa hàm tự lập trình và hàm của Matlab.
19. Hãy lập trình hàm có chức năng tương tự như hàm modnorm(). So sánh tốc độ thực thi giữa hàm tự lập trình và hàm của Matlab.
20. Hãy dùng hàm tạo và kiểm tra CRC lập trình giao thức tự động truyền lại (ARQ).
21. Vẽ đồ thị tỷ lệ lỗi bit của điều chế BPSK với mã BCH trên kênh truyền nhiễu trắng. So sánh độ lợi mã trước và sau khi mã hóa.
22. Vẽ đồ thị tỷ lệ lỗi bit của điều chế MQAM với mã xoắn trên kênh truyền nhiễu trắng. So sánh độ lợi mã trước và sau khi mã hóa.
23. Hãy thực hiện mô phỏng một hệ thống truyền thông bao gồm các khôi: khôi tín hiệu vào là dạng âm thanh, thực hiện mã hóa nguồn (A/mu law), nén tín hiệu, điều chế, kênh truyền nhiễu trắng, giải điều chế, phục hồi tín hiệu và phát ra loa.

CHƯƠNG 5: MÔ PHỎNG KÊNH THÔNG TIN

5.1 Giới thiệu chung

Một hệ thống truyền thông cơ bản bao gồm ít nhất: khối nguồn, khối điều chế, khối kênh truyền và khối giải điều chế. Trong Chương 5, chúng ta tập trung vào mô phỏng và kiểm chứng các kênh truyền, tập trung vào hai loại kênh truyền: kênh truyền thông tin số và kênh truyền vô tuyến cơ bản. Khi kết hợp kênh truyền với khối nguồn, khối điều chế và khối giải điều chế ở Chương 4, chúng ta sẽ có hệ thống thông tin cơ bản. Chương 5 sẽ trình bày sơ lược lý thuyết của các kênh truyền để phục vụ cho phần kiểm chứng kết quả mô phỏng.

Để mô phỏng hệ thống truyền thông có dây và không dây, chúng ta cần phải có Communications Toolbox được cài đặt trên máy tính. Toolbox Communications có đầy đủ các hàm chức năng từ: mã hóa nguồn, mã hóa kênh, các loại kênh truyền, các loại điều chế và kể cả các công cụ để chúng ta kiểm chứng hiệu năng của hệ thống truyền thông mà chúng ta thiết kế.

Để tìm hiểu thêm về Communications Toolbox, chúng ta có thể tham khảo trên website của mathworks tại địa chỉ: <https://www.mathworks.com/help/comm/index.html>.

5.2 Mô hình kênh

Matlab giới thiệu hai cách thức để xây dựng mô hình kênh truyền thông tin. Cách đầu tiên là sử dụng các hàm cho từng chức năng riêng biệt như liệt kê ở Bảng 5.1 và Cách thứ hai là sử dụng các phương thức (method) thuộc đối tượng comm. như liệt kê ở Bảng 5.2.

Bảng 5.1: Các hàm hỗ trợ xây dựng mô hình kênh truyền.

TT	Tên hàm	Chú thích	Cú pháp
1	awgn()	Cộng nhiễu trắng Gauss vào tín hiệu	out = awgn(in,snr,signalpower)
2	bsc()	Kênh truyền nhị phân đối xứng	ndata = bsc(data,probability)
3	stdchan()	Tạo một đối tượng kênh truyền	chan = stdchan(chanType,rs,fd)
4	fogpl()	Tạo suy hao tín hiệu RF gây ra bởi sương mù và mây	L = fogpl(R,freq,T,den)
5	fspl()	Tạo suy hao đường truyền tự do	L = fspl(R,lambda)
6	gaspl()	Tạo suy hao tín hiệu RF do không khí	L = gaspl(range,freq,T,P,den)
7	rainpl()	Tạo suy hao tín hiệu RF do mưa	L = rainpl(range,freq,rainrate)
8	doppler()	Xây dựng một cấu trúc phổ Doppler	s = doppler(specType)

Bảng 5.2: Các hàm thuộc đối tượng comm hỗ trợ xây dựng mô hình và các đặc tính kênh truyền.

TT	Tên hàm	Chú thích
1	comm.AWGNChannel()	Cộng nhiễu trắng vào tín hiệu
2	comm.RayleighChannel()	Lọc tín hiệu vào thông qua một kênh truyền fading Rayleigh đa đường
3	comm.RicianChannel()	Lọc tín hiệu vào thông qua một kênh truyền fading Rician đa đường
4	comm.MIMOChannel()	Lọc tín hiệu vào thông qua một kênh truyền fading MIMO đa đường
5	comm.WINNER2Channel()	Lọc tín hiệu vào thông qua một kênh truyền fading WINNER II đa đường
6	comm.gpu.AWGNChannel()	Cộng nhiễu trắng vào tín hiệu với GPU
7	comm.ThermalNoise()	Cộng nhiễu nhiệt vào tín hiệu

Để tìm hiểu các hàm thuộc đối tượng comm. như ở

Bảng 5.2, chúng ta có thể dùng hàm doc hay help. Chương 5 không tập trung vào các hàm của đối tượng comm., do các hàm chỉ cung cấp chức năng để sử dụng mà không đi vào bản chất toán của hệ thống hay kênh truyền. Bên cạnh đó, đối tượng comm. và các hàm chức năng của comm. thường thay đổi và cập nhật qua các phiên bản của Matlab.

5.2.1 Kênh truyền nhị phân

Kênh truyền nhị phân là kênh truyền đơn giản nhất trong hệ thống truyền tin và không phải là kênh truyền thực tế, thường được sử dụng trong lý thuyết thông tin và mã hóa. Trên kênh truyền nhị phân, máy phát chỉ phát bit 0 và bit 1 với xác suất truyền sai và truyền đúng lần lượt là p và $1 - p$.

Matlab hỗ trợ hàm **bsc()** để mô hình hóa kênh truyền nhị phân bằng cách chèn bit lỗi vào trong chuỗi bit vào theo xác suất sai cho trước. Cú pháp của hàm bsc() là như sau:

Cú pháp:

```
ndata = bsc(data,probability)
```

% data là tín hiệu vào

% probability là xác suất sai

Ví dụ 5.1

Tạo chuỗi bit và truyền qua kênh truyền nhị phân với xác suất sai là 10%.

Giải: Chúng ta có xác suất sai là 10% nghĩa là $p = 0.1$. Sử dụng hàm bsc(), chúng ta có mã nguồn Matlab như sau:

```
% Chiều dài chuỗi bit  
N = 10^2;
```

```
% Tạo tín hiệu vào là một vector 1 dòng và N cột  

Tx = randi([0 1],1,N);  
  

% Xác suất sai  

p = 0.1;  
  

% Tín hiệu sau kênh truyền nhị phân  

Rx = bsc(Tx,p);  
  

% Tính lại xác suất lỗi và so với p.  

sum(Tx~=Rx)/N
```

Ví dụ 5.2

Hãy tạo kênh truyền bsc() bằng hàm randsrc().

Giải: Chúng ta sử dụng hàm randsrc() để tạo ra nguồn có số lượng bit ‘1’ theo xác suất p và thực hiện phép xor tín hiệu nhận được với tín hiệu vào kênh truyền. Có một cách làm khác là dùng phép đảo các bit thay cho phép xor và sẽ đạt được kết quả tương tự như phép xor.

```
% Chiều dài chuỗi bit  

N = 10^6;  
  

% Tạo tín hiệu vào  

Tx = randi([0 1],1,N);  
  

% Xác suất sai  

p = 0.1;  
  

% Sử dụng randsrc()  

I = randsrc(1,N,[1 0; p 1-p]);  

Rx = xor(Tx,I);  
  

% Tính lại xác suất lỗi và so với p.  

sum(Tx~=Rx)/N  
  

% Sử dụng hàm bsc()  

Rx = bsc(Tx,p);  
  

% Tính lại xác suất lỗi và so với p.  

sum(Tx~=Rx)/N
```

5.2.2 Kênh nhiễu trắng

Kênh truyền nhiễu cộng trắng là kênh truyền cơ bản trong lý thuyết thông tin. Nhiễu cộng trắng có ba tính chất: tính cộng là do nhiễu cộng vào tín hiệu thông tin, tính trắng là do có phân phối đều trong băng tần xem xét của hệ thống thông tin và tính Gauss là do nhiễu có phân phối chuẩn trong miền thời gian.

Chú ý rằng kênh nhiễu trắng không xem xét các ảnh hưởng do fading, fading có lựa chọn, can nhiễu, tính không tuyến tính hay méo dạng của kênh truyền. Kênh truyền nhiễu trắng thường được sử dụng trong mô hình kênh vệ tinh và kênh truyền có dây.

Matlab hỗ trợ hàm **awgn()** để mô hình hóa kênh truyền nhiễu trắng, có cú pháp như sau:

Cú pháp:

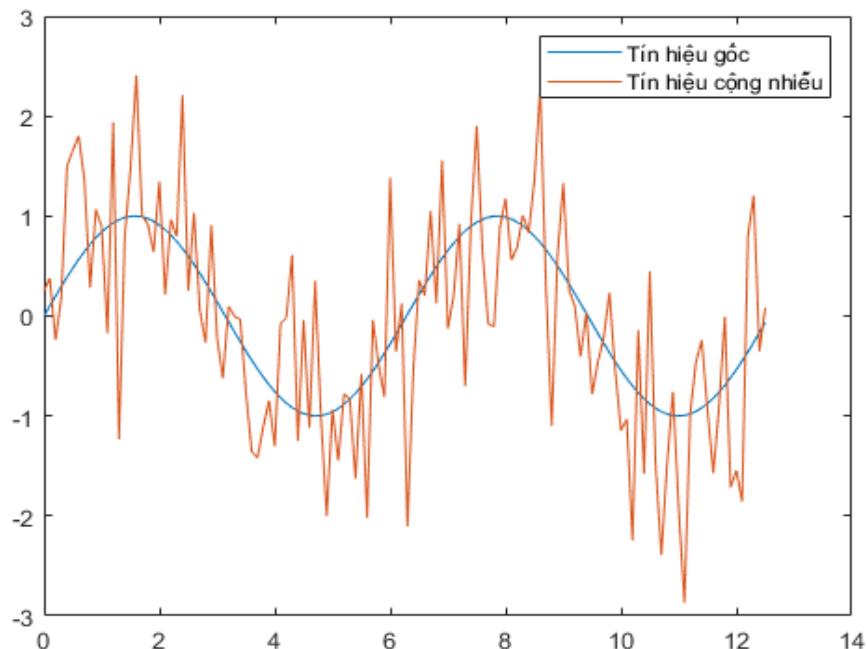
```
out = awgn(in,snr,signalpower)  
% in: tín hiệu vào, giả sử có công suất là 0 dBW  
% snr: tỷ số tín hiệu trên nhiễu đơn vị là dB  
% signalpower: công suất tín hiệu
```

Ví dụ 5.3

Tạo tín hiệu sin trong 2 chu kỳ và cộng nhiễu trắng có cùng công suất dùng hàm **awgn()**.

Giải: Một điểm quan trọng cần chú ý là do tín hiệu sin của chúng ta tạo ra có công suất khác 0 dBW, nên chúng ta phải khai báo tham số “measured” khi sử dụng hàm **awgn()** để công suất nhiễu cộng vào là phù hợp.

```
t = (0:0.1:4*pi)';  
x = sin(t);  
y = awgn(x,0,'measured');  
plot(t,[x y])  
legend('Tín hiệu gốc','Tín hiệu cộng nhiễu')  
set(gcf,'color','white');
```



Hình 5.1: Hình vẽ tín hiệu trước và sau khi cộng nhiễu của Ví dụ 5.3.

Ví dụ 5.4

Tạo tín hiệu điều chế MQAM với $M = 16$, cộng nhiễu trắng với tỷ số tín hiệu trên nhiễu là 10 dB. So sánh công suất tín hiệu trước và sau khi cộng nhiễu. Xem xét hai trường hợp:

- Không chuẩn hóa công suất tín hiệu điều chế.
- Chuẩn hóa công suất tín hiệu điều chế.

Giải: Trong Ví dụ 5.4, chúng ta dùng hàm qammod() và qamdemod() để thực hiện điều chế và giải điều chế MQAM. Để thực hiện chuẩn hóa và không chuẩn hóa tín hiệu, chúng ta lưu ý tham số 'UnitAveragePower' là 'true' hay 'false' của hàm qammod().

```
% Chiều dài mẫu thử  
N = 10^6;  
% Điều chế 16-QAM  
M = 16;  
x = randi([0 M-1],1,N);  
% Tỷ số tín hiệu trên nhiễu 10 dB  
SNRdB = 10;  
  
%% Trường hợp 1: Không chuẩn hóa công suất tín hiệu điều chế  
Tx = qammod(x,M,'UnitAveragePower', false);  
% Cộng nhiễu vào tín hiệu  
Rx = awgn(Tx,10,'measured');  
% Công suất tín hiệu trước khi cộng nhiễu  
var(Tx)  
% Công suất tín hiệu sau khi cộng nhiễu  
var(Rx)  
  
%% Trường hợp 2: Chuẩn hóa công suất tín hiệu điều chế  
Tx = qammod(x,M,'UnitAveragePower', true);  
% Cộng nhiễu vào tín hiệu  
Rx = awgn(Tx,10);  
% Công suất tín hiệu trước khi cộng nhiễu  
var(Tx)  
% Công suất tín hiệu sau khi cộng nhiễu  
var(Rx)  
  
ans =  
10.0085  
  
ans =  
10.1075  
  
ans =  
1.0009  
  
ans =  
1.1008
```

Trong trường hợp 1, công suất tín hiệu trước khi cộng nhiễu là 10.0085 và sau khi cộng nhiễu là 10.1075. Kết quả này là không đúng vì chúng ta đã không chuẩn hóa công suất tín hiệu điều chế và hàm awgn() theo quy ước mặc định đã giả sử tín hiệu vào có công suất là 0 dBW nên công suất nhiễu được cộng vào là 0.1 dBW.

Có một cách thực hiện khác là chúng ta trực tiếp tạo nhiễu trắng và cộng vào tín hiệu. Từ tỷ số tín hiệu trên nhiễu theo dB (SNRdB), chúng ta chuyển đổi thành tỷ số tín hiệu trên nhiễu (SNR) và giả sử rằng công suất nhiễu là bằng một và khi đó công suất tín hiệu là SNR.

Cú pháp:

$\text{SNRdB} = 10;$ % Tỷ số tín hiệu trên nhiễu có đơn vị là dB

$\text{SNR} = 10^{(\text{SNRdB}/10)};$

$\text{N} = \text{sqrt}(2/\text{SNR}).*(\text{randn}(1,\text{N}) + 1j*\text{randn}(1,\text{N}))$ % Nhiễu AWGN giả sử công suất tín hiệu là 1 W.

Ví dụ 5.5

Tạo tín hiệu điều chế MQAM với $M = 16$, cộng nhiễu với tỷ số tín hiệu trên nhiễu là 10 dB không dùng hàm awgn(). So sánh công suất tín hiệu trước và sau khi cộng nhiễu.

Giải: Một lưu ý quan trọng khi thực hiện mô phỏng kênh truyền nhiễu trắng là do nhiễu trắng có phần thực và phần ảo nên công suất nhiễu được chia ra làm hai phần, dẫn đến biên độ của phần thực và phần ảo phải nhân với hệ số $\sqrt{1/2}$.

```
% VD 5.4
N = 10^6;
M = 16;
x = randi([0 M-1],1,N);

% Tỷ số tín hiệu trên nhiễu
SNRdB = 10;

% Điều chế tín hiệu có chuẩn hóa công suất
Tx = qammod(x,M,'UnitAveragePower', true);

% Chuyển đổi SNR theo dB thành SNR
SNR = 10^(SNRdB/10);

% Công suất nhiễu
nP = 1./SNR;

% Tạo nhiễu
aN = sqrt(nP/2).*(randn(size(Tx)) + 1j*randn(size(Tx)));

% Cộng nhiễu vào tín hiệu
Rx = Tx + aN;

% Công suất tín hiệu trước khi cộng nhiễu
var(Tx)
```

```
% Công suất tín hiệu sau khi cộng nhiễu  
var(Rx)
```

```
ans =
```

```
0.9991
```

```
ans =
```

```
1.0993
```

Ví dụ 5.6

Hãy mô phỏng kênh truyền nhiễu trắng cho điều chế BPSK từ 0 đến 10 dB.

- Vẽ đồ thị tỷ lệ lỗi bit cho điều chế BPSK.
- So sánh với xác suất lỗi bit cho điều chế BPSK.

Giải: Chúng ta biết rằng tỷ lệ lỗi bit của điều chế BPSK ở kênh truyền nhiễu trắng là $P_b = Q(\sqrt{2\gamma})$ [4] với γ là tỷ số tín hiệu trên nhiễu. Điểm đặc biệt của chương trình Matlab của Ví dụ 5.6 bên dưới là:

- Không dùng vòng lặp for và chỉ xử lý trên ma trận.
- Số lượng bit chạy là 1 triệu bit trên một điểm chạy.
- Không sử dụng hàm điều chế/giải điều chế của Matlab.

Để tính toán tỷ lệ lỗi bit lý thuyết để so sánh, chúng ta sử dụng hàm **qfunc()**.

```
clear all  
clc  
%  
SNRdB = (0:1:10)';  
N = 10^6;  
M = 2;  
% Tạo ra dữ liệu phân bố đều có số hàng là bằng với số điểm mô phỏng và số cột là số mẫu thử  
x = randi([0 M-1],length(SNRdB),N);  
% Điều chế: thực hiện ánh xạ bit '1' là -1 và bit '0' là 1  
xM = [1 -1];  
Tx = xM(x + 1);  
  
% Tỷ số tín hiệu trên nhiễu: chuyển đổi từ dB sang số thực  
SNR = 10.^^(SNRdB/10);  
  
% Công suất nhiễu với công suất tín hiệu bằng 1W  
nP = 1./SNR;  
  
% Tạo nhiễu trắng cộng  
aN = sqrt(nP/2).*(randn(size(Tx)) + 1i*randn(size(Tx)));  
% Cộng nhiễu vào tín hiệu  
Rx = Tx + aN;
```

```

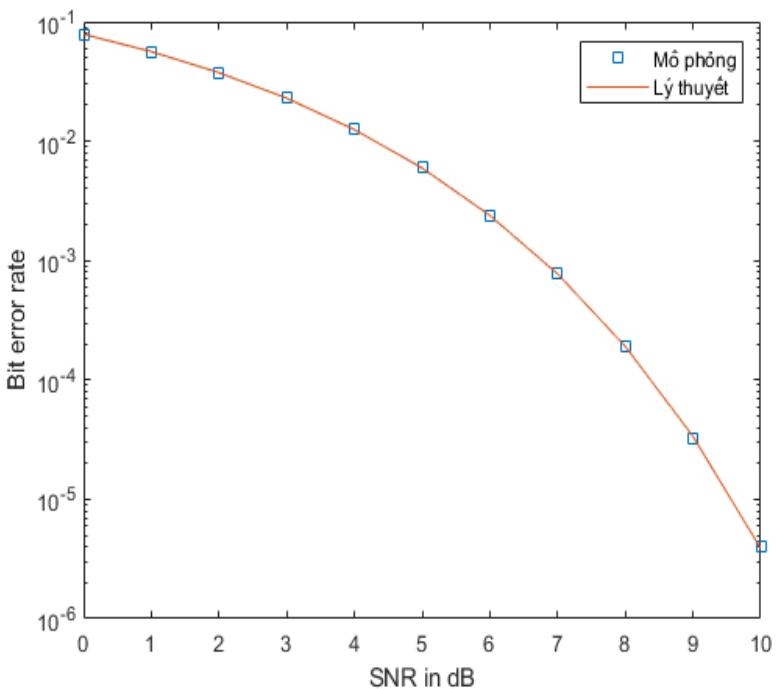
% Giải điều chế: Giả sử vùng quyết định của bit 1 là nửa trái không gian và bit 0 là nửa phải không gian.
x_ = 0.* (real(Rx) > 0) + 1.* (real(Rx) <= 0);

% Tính tỷ lệ lỗi bit
BER_s = sum(x_~=x,2)/N;

% BER lý thuyết
BER_t = qfunc(sqrt(2*SNR));

% Vẽ đồ thị BER
semilogy(SNRdB,BER_s,'s',SNRdB,BER_t);
legend('Mô phỏng','Lý thuyết');
xlabel('SNR theo dB');
ylabel('Tỷ lệ lỗi bit');
set(gcf,'color','white');

```



Hình 5.2: Tỷ lệ lỗi bit của điều chế BPSK ở kênh truyền nhiễu trắng: mô phỏng và lý thuyết.

5.2.3 Kênh truyền log-normal

Khi đường truyền không dây giữa máy phát và máy thu có các vật cản thì công suất của tín hiệu thu tại máy thu sẽ dao động, khi đó kênh truyền giữa máy phát và máy thu sẽ là kênh truyền log normal [7]. Kênh truyền log-normal rất phù hợp cho các kênh truyền trong nhà với tốc độ di chuyển thấp. Hàm PDF của tỷ số tín hiệu trên nhiễu, γ , sẽ có dạng như sau:

$$f(\gamma) = \frac{\xi}{\sigma_\gamma \gamma \sqrt{2\pi}} \exp\left[-\frac{(10 \log_{10} \gamma - \mu_\gamma)^2}{2\sigma_\gamma^2}\right] \quad (5.1)$$

với $\xi = 10 / \ln 10$, μ_γ (theo dB) và σ_γ (theo dB) là giá trị trung bình và độ lệch chuẩn của $10 \log_{10} \gamma$. Hàm CDF của γ có dạng như sau [8]:

$$F_\gamma(\gamma) = Q\left(\frac{\mu_\gamma - 10 \log_{10} \gamma}{\sigma_\gamma}\right) \quad (5.2)$$

Ví dụ 5.7

Hãy mô phỏng và so sánh với lý thuyết xác suất dùng của hệ thống trên kênh truyền log-normal.

Giải: Để tạo ra công suất thu thay đổi theo phân bố log-normal, chúng ta dùng hàm lognrnd() của Matlab. Tuy nhiên, hàm lognrnd() của Matlab có hàm PDF được định nghĩa khác so với hàm PDF của tỷ số tín hiệu trên nhiễu ở công thức (5.1). Cụ thể, nếu biến ngẫu nhiên x có phân bố log-normal được tạo từ hàm lognrnd(), thì hàm PDF của x có dạng như sau:

$$f(x) = \frac{1}{\sqrt{2\pi}x\sigma_x} \exp\left[-\frac{(\ln x - \mu_x)^2}{2\sigma_x^2}\right]. \quad (5.3)$$

Hay nói cách khác, ta có mối liên hệ sau:

$$\mu_x = \frac{\mu_\gamma}{10 / \ln 10}, \quad (5.4)$$

$$\sigma_x = \frac{\sigma_\gamma}{10 / \ln 10}. \quad (5.5)$$

```
clear all
clc

% Công suất của máy phát
EbNo = (0:2:20)';
P = 10.^EbNo/10;

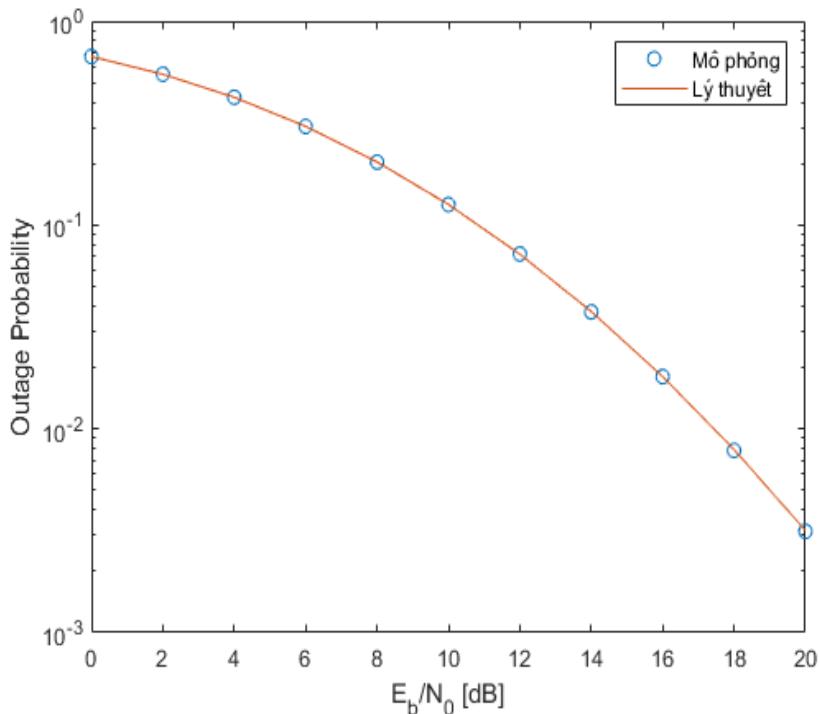
% Tốc độ truyền mong muốn
R = 2;
% Nguồn SNR
GT = 2.^R - 1;
% Tính toán thông số
xi = 10/log(10);
mu = 10^(3/10);
sigma = 10^(8/10);

% Mô phỏng
N = 10^6;
% Tỷ số tín hiệu trên nhiễu
G = P.*lognrnd(mu./xi,sigma./xi,1,N);

% Tính xác suất dùng
OP_s = sum(G < GT,2)/N;
```

```
% Lý thuyết
OP_t = qfunc((mu -10*log10(GT./P))./sigma);

% Vẽ đồ thị
semilogy(EbNo,OP_s,'o',EbNo,OP_t,'-')
xlabel('E_b/N_0 [dB]');
ylabel('Xác suất dừng');
legend('Mô phỏng','Lý thuyết');
set(gcf,'color','white');
```



Hình 5.3: Xác suất dừng của kênh truyền log normal: mô phỏng và lý thuyết.

5.2.4 Kênh fading Rayleigh

Kênh truyền fading Rayleigh là mô hình kênh truyền rất phổ biến trong thông tin vô tuyến, mô hình hóa đường truyền vô tuyến có nhiều vật cản, không có đường truyền trực tiếp mà có nhiều tia tán xạ và phản xạ đến máy thu [4].

Gọi $h = \alpha e^{j\varphi}$ là hệ số kênh truyền phức của kênh truyền fading Rayleigh, h sẽ là biến Gauss phức có biên độ $\alpha = |h|$ tuân theo phân bố Rayleigh với tham số là σ và có pha φ tuân theo phân bố đều trong khoảng $(0, 2\pi)$. Gọi γ là tỷ số tín hiệu trên nhiễu, ta có:

$$\gamma = \frac{P|h|^2}{N_0} \quad (5.6)$$

với P là công suất của máy phát và N_0 là phuơng sai của nhiễu trắng tại máy thu và $\lambda_h = E\{|h|^2\} = 2\sigma^2$ là độ lợi kênh truyền trung bình. Trong khi thực hiện mô phỏng các hệ thống thông tin vô tuyến, chúng ta thường đưa hiệu ứng suy hao đường truyền vào hệ số λ_h , ví dụ với mô hình suy hao đường truyền đơn giản, chúng ta có $\lambda_h = K_0 d^{-\eta}$ với K_0 là hệ số điều chỉnh mô hình, d là khoảng cách giữa máy phát và máy thu và η là hệ số suy hao đường truyền. Do h là biến Gauss phức, γ sẽ có phân bố mũ. Hàm PDF và hàm CDF của γ có dạng như sau:

$$f(\gamma) = \frac{1}{\bar{\gamma}} e^{-\frac{\gamma}{\bar{\gamma}}}, \quad (5.7)$$

$$F(\gamma) = 1 - e^{-\frac{\gamma}{\bar{\gamma}}} \quad (5.8)$$

với $\bar{\gamma} = \frac{P\lambda_h}{N_0}$.

Để tạo kênh truyền fading Rayleigh, chúng ta có ba cách sau:

- Cách 1: Tạo hệ số kênh truyền h từ định nghĩa h là phân bố Gauss phức.
- Cách 2: Tạo hệ số kênh truyền h từ định nghĩa $|h|$ có phân bố Rayleigh.
- Cách 3: Tạo hệ số kênh truyền h từ định nghĩa $|h|^2$ có phân bố hàm mũ.

Ví dụ 5.8

Hãy tạo kênh truyền fading Rayleigh theo 3 cách như trên và kiểm chứng với lý thuyết.

Giải: Tùy thuộc vào tham số hiệu năng hay đặc tính hệ thống mà chúng ta cần nghiên cứu cho hệ thống truyền thông, chúng ta sẽ tạo ra hệ số kênh truyền h hay chỉ cần tạo ra độ lợi kênh truyền tức thời $|h|^2$.

Cụ thể, ngoại trừ khi cần mô phỏng tỷ lệ lỗi bit hay tỷ lệ lỗi symbol, chúng ta cần phải tạo hệ số kênh truyền h , các tham số hiệu năng khác, ví dụ như xác suất dừng hay dung lượng kênh truyền, chúng ta chỉ cần $|h|^2$.

Trong Ví dụ 5.8, chúng ta sẽ tạo ra h trong cả ba trường hợp. Điểm cần chú ý khi tạo ra biến ngẫu nhiên là cần chú ý các tham số đầu vào cho các hàm tạo biến ngẫu nhiên cho phù hợp.

Nhắc lại λ_h là độ lợi kênh trung bình, ta sẽ có các lưu ý sau:

- Cách 1: thành phần thực và thành phần ảo của h trong cách này sẽ có biên độ là $\sqrt{\lambda_h/2}$,
- Cách 2: $|h|$ có tham số điều chỉnh (scale parameter) là $\sqrt{\lambda_h/2}$,
- Cách 3: $|h|^2$ có tham số trung bình là $\sqrt{2\lambda_h}$ vì nếu X là biến ngẫu nhiên hàm mũ có tham số (rate parameter) λ thì $Y = \sqrt{X}$ có phân bố Rayleigh với tham số (scale parameter) $1/\sqrt{2\lambda}$.

```

clear all
clc
SNRdB = (0:3:30)';
SNR = 10.^ (SNRdB/10);
lambda_h = 2;
gth = 1;

%% Mô phỏng
N = 10^6;

%% Tạo hệ số kênh truyền
% Cách 1
h = sqrt(lambda_h/2).*(randn(1,N) + 1i*randn(1,N));
% Tỷ số tín hiệu trên nhiễu
g = SNR.*abs(h).^2;
% Xác suất dùng
OP_1 = sum(g < gth,2)/N;

% Cách 2
h_a = raylrnd(sqrt(lambda_h/2),1,N);
phi = 2*pi*rand(1,N);
h = h_a.^(cos(phi) + 1i*sin(phi));
% Tỷ số tín hiệu trên nhiễu
g = SNR.*abs(h).^2;
% Xác suất dùng
OP_2 = sum(g < gth,2)/N;

% Cách 3
h_a = sqrt(exprnd(sqrt(2*lambda_h),1,N));
phi = 2*pi*rand(1,N);
h = h_a.^(cos(phi) + 1i*sin(phi));
% Tỷ số tín hiệu trên nhiễu
g = SNR.*abs(h).^2;
% Xác suất dùng
OP_3 = sum(g < gth,2)/N;

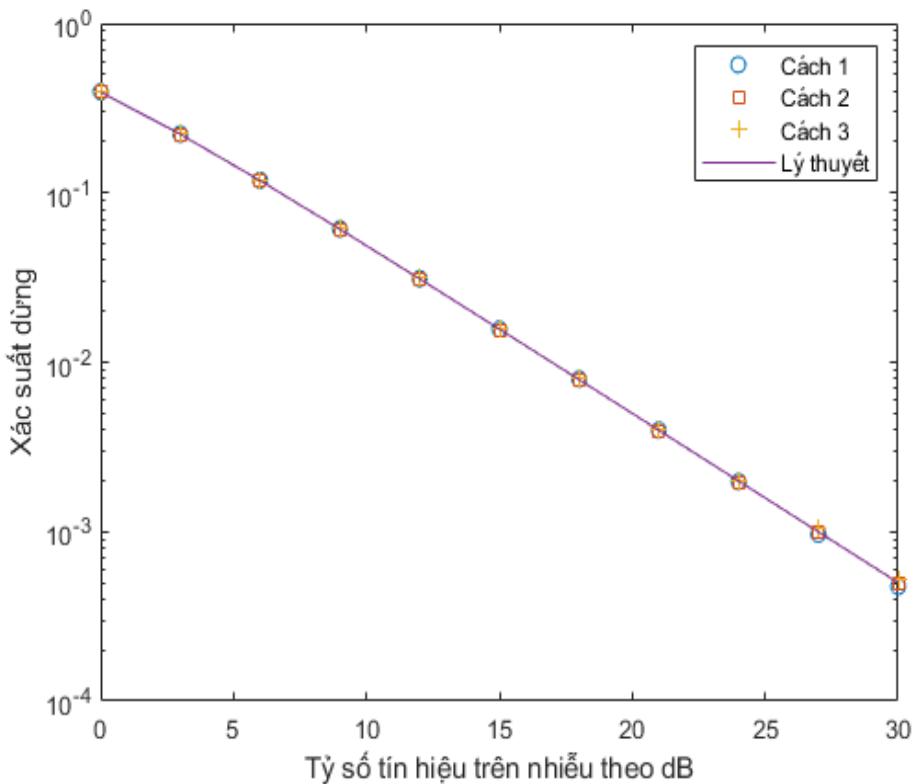
%% OP lý thuyết
% Tỷ số tín hiệu trên nhiễu trung bình
g_ = SNR.*lambda_h;
OP_t = 1 - exp(-gth./g_);
% Vẽ đồ thị BER
semilogy(SNRdB,OP_1,'o',SNRdB,OP_2,'s',SNRdB,OP_3,'+',SNRdB,OP_t);
legend('Cách 1','Cách 2','Cách 3','Lý thuyết');

```

```

xlabel('Tỷ số tín hiệu trên nhiễu theo dB');
ylabel('Xác suất dừng');
set(gcf,'color','white');

```



Hình 5.4: Tạo kênh truyền fading Rayleigh theo ba cách.

5.2.5 Kênh fading Nakagami- m

Kênh truyền fading Nakagami- m có tính tổng quát hơn kênh truyền fading Rayleigh, cụ thể kênh truyền fading Rayleigh là một trường hợp đặc biệt của kênh truyền fading Nakagami- m khi $m=1$ [9]. Nếu $\alpha = |h|$ là biên độ của hệ số kênh truyền phức h của kênh truyền fading Nakagami- m , α sẽ có phân bố Nakagami- m với hàm PDF như sau:

$$f(\alpha) = \frac{2m^m \alpha^{2m-1}}{\Omega^m \Gamma(m)} \exp\left(-\frac{m\alpha^2}{\Omega}\right), \quad \alpha > 0 \quad (5.9)$$

với m là tham số hình dạng (shape parameter) có giá trị từ $\frac{1}{2}$ đến vô cùng và Ω là tham số trãi (spread parameter) của phân bố Nakagami- m .

Gọi γ là tỷ số tín hiệu trên nhiễu của hệ thống tại máy thu, γ sẽ có phân bố gamma với hàm PDF như sau:

$$f_\gamma(\gamma) = \frac{m^m \gamma^{m-1}}{\gamma^m \Gamma(m)} \exp\left(-\frac{m\gamma}{\Omega}\right). \quad (5.10)$$

Hàm CDF của γ là:

$$F(\gamma) = 1 - \frac{\Gamma\left(m, \frac{m\gamma}{\gamma}\right)}{\Gamma(m)} \quad (5.11)$$

với $\Gamma()$ và $\Gamma(.,.)$ lần lượt là hàm Gamma (Gamma function) và hàm Gamma chưa hoàn chỉnh chẵn trên (the upper incomplete Gamma function).

Chúng ta lưu ý là nếu α có phân bố Nakagami- m với tham số m và Ω , thì α^2 có phân bố gamma với tham số hình dạng m và tham số điều chỉnh $\frac{\Omega}{m}$. Ngoài ra ta có thể tạo α có

phân bố Nakagami- m có tham số m và Ω thông qua phân phối Chi, cụ thể $\alpha = \sqrt{\frac{\Omega}{2m}}\beta$, với

β là biến ngẫu nhiên phân bố Chi-Square có tham số $2m$. Để mô phỏng kênh truyền Nakgami- m , chúng ta cần hai tham số m và Ω và có ba cách để tạo kênh truyền:

- Cách 1: Tạo từ phân bố Nakagami- m tham số m và Ω .
- Cách 2: Tạo từ phân bố Gamma có tham số m và $\frac{\Omega}{m}$.
- Cách 3: Tạo từ phân bố Chi có tham số $2m$.

Ví dụ 5.9

Hãy tạo kênh truyền Nakgami- m từ ba cách và so sánh kết quả với phân tích lý thuyết.

Giải: Chúng ta sẽ sử dụng hàm **random()** cho cách 1, hàm **gamrnd()** cho cách 2 và hàm **chi2rnd()** cho cách 3.

```
clear all
clc
SNRdB = (0:2:20)';
SNR = 10.^ (SNRdB/10);
% Shape parameter
m = 2;
% Scale parameter
Omega = 2;
% SNR ngưỡng
gth = 2;

%% Mô phỏng
N = 10^6;
%% Tạo hệ số kênh truyền
% Cách 1
alpha = random('Nakagami',m,Omega,1,N);
phi = 2*pi*rand(1,N);
h = alpha.* (cos(phi) + 1i*sin(phi));
% Tỷ số tín hiệu trên nhiễu
g = SNR.*abs(h).^2;
```

```

% Xác suất dùng
OP_1 = sum(g < gth,2)/N;

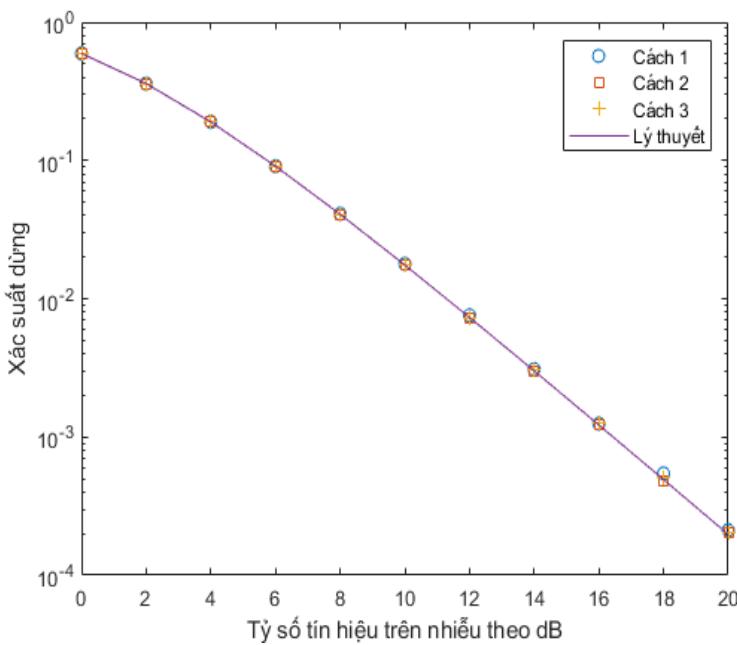
% Cách 2
alpha = sqrt(gamrnd(m,Omega/m,1,N));
phi = 2*pi*rand(1,N);
h = alpha.*(cos(phi) + 1i*sin(phi));
% Tỷ số tín hiệu trên nhiễu
g = SNR.*abs(h).^2;
% Xác suất dùng
OP_2 = sum(g < gth,2)/N;

% Cách 3
alpha = sqrt(Omega/(2*m)).*sqrt(chi2rnd(2*m,1,N));
phi = 2*pi*rand(1,N);
h = alpha.*((cos(phi) + 1i*sin(phi)));
% Tỷ số tín hiệu trên nhiễu
g = SNR.*abs(h).^2;
% Xác suất dùng
OP_3 = sum(g < gth,2)/N;

%% OP lý thuyết
% Tỷ số tín hiệu trên nhiễu trung bình
g_ = SNR.*Omega;
OP_t = 1 - gamma(m).*gammainc(m*gth./g_,m,'upper');

% Vẽ đồ thị BER
semilogy(SNRdB,OP_1,'o',SNRdB,OP_2,'s',SNRdB,OP_3,'+',SNRdB,OP_t);
legend('Cách 1','Cách 2','Cách 3','Lý thuyết');
xlabel('Tỷ số tín hiệu trên nhiễu theo dB');
ylabel('Xác suất dùng');
set(gcf,'color','white');

```



Hình 5.5: Mô phỏng kênh truyền fading Nakagami-m theo ba cách.

5.2.6 Kênh truyền fading Rician

Kênh truyền fading Rician dùng để mô phỏng đường truyền vô tuyến có tồn tại đường truyền trực tiếp từ máy phát đến máy thu (đường Light of Sight - LOS) và các đường ngẫu nhiên khác yếu hơn.

Gọi h là hệ số kênh truyền phức của kênh truyền fading Rician, thì $\alpha = |h|$ sẽ có phân bố Rician (hay phân bố Rice). Một kênh truyền Rician sẽ đặc trưng bởi hai tham số chính, đó là K và Ω . K là tỷ số công suất giữa đường truyền chính (v^2) và các đường truyền còn lại ($2\sigma^2$), hay

$$K = \frac{v^2}{2\sigma^2}, \quad (5.12)$$

$$\Omega = v^2 + 2\sigma^2 \quad (5.13)$$

với Ω là tổng công suất. Hay ta có:

$$\sigma^2 = \frac{\Omega}{2(K+1)}, \quad (5.14)$$

$$v^2 = \frac{K\Omega}{K+1}. \quad (5.15)$$

Hàm PDF của α có dạng như sau:

$$f(\alpha) = \frac{2(K+1)\alpha}{\Omega} \exp\left(-K - \frac{(K+1)\alpha^2}{\Omega}\right) I_0\left(2\sqrt{\frac{K(K+1)}{\Omega}}\alpha\right) \quad (5.16)$$

với $I_0()$ là hàm Bessel điều chỉnh bậc một của loại một.

Gọi γ là tỷ số tín hiệu trên nhiễu của kênh truyền, hàm PDF của γ có dạng như sau:

$$f_\gamma(\gamma) = \frac{(1+K)e^{-K}}{\bar{\gamma}} \exp\left(-\frac{(1+K)\gamma}{\bar{\gamma}}\right) I_0\left(2\sqrt{\frac{K(1+K)}{\bar{\gamma}}}\gamma\right). \quad (5.17)$$

Hàm CDF của γ có dạng như sau [10]:

$$F(\gamma) = 1 - Q_1\left(\sqrt{2K}, \sqrt{\frac{2(K+1)\gamma}{\bar{\gamma}}}\right) \quad (5.18)$$

với $Q_m(.,.)$ là hàm Marcum Q [11].

Chú ý rằng nếu $X \sim N(v\cos\theta, \sigma^2)$ và $Y \sim N(v\sin\theta, \sigma^2)$, thì $\alpha = \sqrt{x^2 + y^2}$ là biến phân bố Rician với tham số (v, σ) .

Chúng ta có hai cách để tạo kênh truyền Rician trong Matlab:

- Cách 1: Sử dụng hàm random() để tạo biến ngẫu nhiên phân phối Rician.
- Cách 2: Sử dụng định nghĩa từ biến ngẫu nhiên phân bô chuẩn có trung bình khác không.

Ví dụ 5.10

Mô phỏng kênh truyền fading Rician với hệ số K cho trước bằng hai cách.

Giải: Hàm Marcum Q trong Matlab là hàm **marcumq()**. Mã nguồn Matlab cho kênh Rician là như sau:

```

clear all
clc
SNRdB = (0:2:20)';
SNR = 10.^ (SNRdB/10);
% Shape parameter
K = 5;
% Scale parameter
Omega = 2;

% LoS: A - sigma
v2 = Omega*K/(1 + K);
v = sqrt(v2);

% NLOS - sigma - B
s2 = 1/2*Omega/(1+K);
s = sqrt(s2);

% SNR ngưỡng
gth = 2;

%% Mô phỏng
N = 10^6;

%% Tạo hệ số kênh truyền
% Cách 1
alpha = random('Rician',v,s,1,N);
phi = 2*pi*rand(1,N);
h = alpha.* (cos(phi) + 1i*sin(phi));
% Tỷ số tín hiệu trên nhiễu
g = SNR.*abs(h).^2;
% Xác suất dừng
OP_1 = sum(g < gth,2)/N;

% Cách 2
X = v*cos(pi/4) + s.*randn(1,N);
Y = v*sin(pi/4) + s.*randn(1,N);
h = X + 1i*Y;
% Tỷ số tín hiệu trên nhiễu
g = SNR.*abs(h).^2;
% Xác suất dừng
OP_2 = sum(g < gth,2)/N;

%% OP lý thuyết

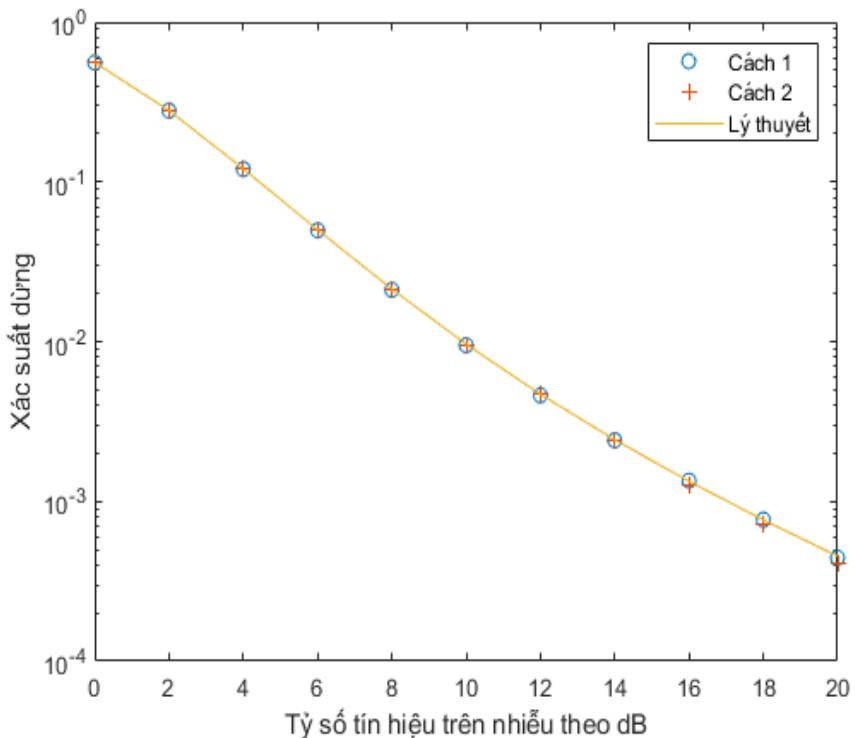
```

```

% Tỷ số tín hiệu trên nhiễu trung bình
g_ = SNR.*Omega;
OP_t = 1 - marcumq(sqrt(2*K),sqrt(2*(1+K)*gth./g_),1);

% Vẽ đồ thị BER
semilogy(SNRdB,OP_1,'o',SNRdB,OP_2,'+',SNRdB,OP_t);
legend('Cách 1','Cách 2','Lý thuyết');
xlabel('Tỷ số tín hiệu trên nhiễu theo dB');
ylabel('Xác suất đúng');
set(gcf,'color','white');

```



Hình 5.6: Mô phỏng kênh truyền fading Rician bằng hai cách.

5.2.7 Kênh truyền tương quan

Trong quá trình thực hiện mô phỏng các hệ thống thông tin vô tuyến, đặc biệt là các hệ thống đa anten, chúng ta cần mô phỏng các kênh truyền tương quan khi mà khoảng cách giữa các anten là nhỏ, thường là trên cùng một thiết bị, dẫn đến các kênh truyền đến các anten này là tương quan.

Trước hết chúng ta xem xét mô hình tương quan đơn giản giữa 1 máy phát và 1 máy thu, trong đó máy phát có 1 anten và máy thu có 2 anten. Gọi h_1 và h_2 lần lượt là hệ số kênh truyền từ máy phát đến anten thứ nhất và hai của máy thu. Gọi ρ là hệ số tương quan giữa hai kênh truyền, ta có mô hình tương quan sau:

$$h_2 = \rho h_1 + \sqrt{1-\rho^2} n \quad (5.19)$$

với h_1 và n có chung phân bố và giá trị trung bình, cụ thể là $E\{|h_1|^2\} = E\{|n|^2\} = \lambda$.

Giả sử máy thu sử dụng hệ thống lựa chọn anten phát với hai anten, ta có tỷ số tín hiệu trên nhiễu tương đương trung bình của hệ thống như sau [12]:

$$\bar{\gamma}_{\text{SC}} = \bar{\gamma} \left(1 + \frac{\sqrt{1 - \rho^2}}{2} \right) \quad (5.20)$$

với $\bar{\gamma}$ là tỷ số tín hiệu trên nhiễu của từng kênh.

Ví dụ 5.11

Hãy lập trình kiểm chứng công thức $\bar{\gamma}_{\text{SC}}$ ở (5.20).

Giải: Chương trình Matlab của Ví dụ 5.11 như sau:

```
clear all
clc
SNRdB = (0:1:5)';
Omega = 2;
SNR = 10.^ (SNRdB/10);

% SNR nguồn
gth = 2;

%% Mô phỏng
N = 10^6;
rho = 0.7;

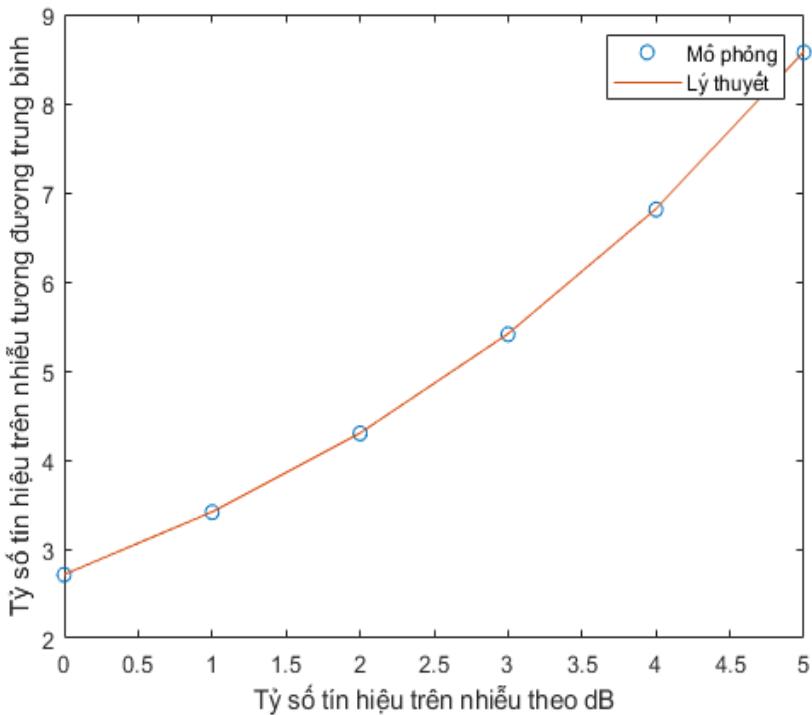
%% Tạo hệ số kênh truyền
% Cách 1
h1 = sqrt(Omega/2).*(randn(1,N) + 1i*randn(1,N));
n = sqrt(Omega/2).*(randn(1,N) + 1i*randn(1,N));
h2 = rho.*h1 + sqrt(1 - rho^2).*n;

% Tỷ số tín hiệu trên nhiễu
g1 = SNR.*abs(h1).^2;
g2 = SNR.*abs(h2).^2;

% Xác suất dùng
gSC = max(g1,g2);
gSC_s = mean(gSC,2);

% Lý thuyết
g_ = Omega.*SNR;
gSC_t = g_.*(1 + sqrt(1 - rho.^2)/2);

plot(SNRdB,gSC_s,'o',SNRdB,gSC_t,'-');
legend('Mô phỏng','Lý thuyết');
xlabel('Tỷ số tín hiệu trên nhiễu theo dB');
ylabel('Tỷ số tín hiệu trên nhiễu tương đương trung bình');
set(gcf,'color','white');
```



Hình 5.7: Tỷ số tín hiệu trên nhiễu tương đương trung bình của hệ thống SC trên kênh truyền fading Rayleigh tương quan.

5.3 Bài tập

1. Hãy dùng hàm edit để tìm hiểu hàm bsc(). Giải thích các dòng lệnh.
2. Xem xét hệ thống truyền gói tin trên kênh truyền nhị phân, mỗi gói tin có chiều dài là $N = 1000$ với xác suất lỗi bit $p = 10^{-3}$. Số lần truyền lại cho mỗi gói tin là 3. Hãy mô phỏng để tìm xác suất nhận đúng gói tin ở máy thu. So sánh với lý thuyết.
3. Hai kênh truyền nhị phân đối xứng có xác suất lỗi lần lượt là p_1 và p_2 kết nối tuần tự với nhau. Xác định xác suất lỗi của kênh truyền mới.
4. Hãy dùng hàm edit với hàm awgn() để tìm cách thức hàm awgn() cộng nhiễu vào tín hiệu. Giải thích các dòng lệnh và nhận xét.
5. Hãy mô phỏng kênh truyền nhiễu trắng với điều chế MPSK và kiểm chứng với lý thuyết.
6. Hãy mô phỏng kênh truyền nhiễu trắng với điều chế DPSK và kiểm chứng với lý thuyết.
7. Hãy mô phỏng kênh truyền nhiễu trắng với điều chế MQAM và kiểm chứng với lý thuyết.
8. Hãy mô phỏng để chứng minh rằng dung lượng của kênh truyền nhiễu trắng là chẵn trên của dung lượng kênh truyền fading trong cùng điều kiện.

9. Hãy mô phỏng kênh truyền log-normal với điều chế BPSK và so sánh với lý thuyết.
10. Hãy dùng hàm edit với hàm raylrnd() để tìm hiểu cách Matlab tạo ra biến ngẫu nhiên có phân bố Rayleigh. Giải thích và nhận xét hàm.
11. Hãy mô phỏng kênh truyền fading Rayleigh với điều chế MPSK và so sánh với lý thuyết.
12. Hãy mô phỏng kênh truyền fading Rayleigh với điều chế QMAM và so sánh với lý thuyết.
13. Hãy mô phỏng kênh truyền Nakagami- m với điều chế BPSK và so sánh với lý thuyết.
14. Hãy mô phỏng kênh truyền fading Rician với điều chế BPSK và so sánh với lý thuyết.
15. Hãy mô phỏng kỹ thuật lựa chọn anten phát 2 kênh khi kênh truyền là fading Rayleigh tương quan với hệ số tương quan ρ .
16. Hãy mô hình hóa độ lợi kênh truyền có K chặng tuyến tính trên 1 đường thẳng sử dụng mô hình kênh truyền suy hao đơn giản với giả sử tổng chiều dài chuẩn hóa giữa nút nguồn và nút đích là 1. So sánh xác suất dừng của hai hệ thống: truyền trực tiếp và truyền đa chặng, trên kênh truyền fading Rayleigh.

CHƯƠNG 6: ƯỚC TÍNH THAM SỐ VÀ ĐÁNH GIÁ HIỆU NĂNG

6.1 Các tham số hiệu năng của hệ thống thông tin

Hệ thống thông tin, theo mô hình tham chiếu hệ thống mở 7 lớp, thì mỗi lớp có các tham số hiệu năng riêng. Tuy nhiên trong giới hạn của cuốn sách này, chúng ta chỉ tập trung vào một số tham số quan trọng của hệ thống, đặc biệt ở lớp vật lý như sau.

6.1.1 Tỷ số tín hiệu trên nhiễu trung bình

Tỷ số tín hiệu trên nhiễu trung bình viết đầy đủ là tỷ số công suất tín hiệu trên công suất nhiễu trung bình là một tham số hiệu năng cơ bản của hệ thống thông tin vô tuyến, đặc biệt hữu dụng khi hệ thống sử dụng kỹ thuật phân tần.

Xem xét một hệ thống thông tin, chúng ta có tỷ số tín hiệu trên nhiễu trung bình được định nghĩa như sau:

$$\begin{aligned}\bar{\gamma} &= \int_0^{\infty} \gamma f_{\gamma}(\gamma) d\gamma \\ &= \frac{P\lambda}{N_0}\end{aligned}\tag{6.1}$$

với P là công suất của máy phát, λ là độ lợi kênh trung bình và N_0 là công suất nhiễu tại máy thu.

6.1.2 Xác suất dùng

Xác suất dùng là tham số hiệu năng quan trọng cho phép đánh giá hiệu năng của hệ thống thông tin mà không cần biết loại điều chế sử dụng. Xác suất dùng được định nghĩa là xác suất đối với lý thuyết và tỷ lệ đối với mô phỏng mà dung lượng chuẩn hóa của kênh truyền nhỏ hơn tốc độ dịch vụ mong muốn (hay yêu cầu) cho trước R , cụ thể:

$$\begin{aligned}OP &= \Pr[\log_2(1+\gamma) < R] \\ &= \Pr(\gamma < \gamma_{\text{th}}) \\ &= F_{\gamma}(\gamma_{\text{th}})\end{aligned}\tag{6.2}$$

với $F_{\gamma}(\gamma_{\text{th}})$ là hàm phân bố xác suất tích lũy của tỷ số tín hiệu trên nhiễu γ và $\gamma_{\text{th}} = 2^R - 1$ là ngưỡng dùng.

Hệ thống một anten phát và một anten thu (Single Input Single Output - SISO) hoạt động ở kênh truyền fading Rayleigh có xác suất dùng như sau:

$$OP = 1 - \exp\left(-\frac{\gamma_{th}}{\bar{\gamma}}\right). \quad (6.3)$$

Ở vùng tỷ lệ tín hiệu trên nhiễu cao, OP có thể xấp xỉ bằng cách áp dụng $e^{-x} = 1 - x$ cho x nhỏ, khi đó từ (6.3) ta có:

$$OP \approx \frac{\gamma_{th}}{\bar{\gamma}}. \quad (6.4)$$

6.1.3 Dung lượng dùng của kênh truyền

Dung lượng Shannon của kênh truyền là tham số chính của một hệ thống, cho biết tốc độ lý thuyết truyền thông tin tối đa của hệ thống mà không lỗi. Ở kênh truyền nhiễu trắng, dung lượng Shannon có công thức sau:

$$C = B \log_2(1 + \gamma). \quad (6.5)$$

Trong công thức (6.5), B là băng thông kênh truyền có đơn vị là Hz, thường $B = 1$ để tính dung lượng chuẩn trên 1 Hz băng thông.

Ở kênh truyền fading Rayleigh, dung lượng Shannon trung bình có thể tính theo công thức (6.6) sau khi áp dụng phép biến đổi (4.337.2) ở [13] như sau:

$$\begin{aligned} \bar{C} &= \int_0^{\infty} B \log_2(1 + \gamma) f_{\gamma}(\gamma) d\gamma \\ &= -\frac{B}{\ln 2} e^{-\frac{1}{\bar{\gamma}}} Ei\left(-\frac{1}{\bar{\gamma}}\right) \end{aligned} \quad (6.6)$$

với $Ei(x) = -\int_{-x}^{\infty} \frac{e^{-t}}{t} dt$ được định nghĩa tại mục 8.211 của [13] và $\bar{\gamma} = E\{\gamma\}$.

6.1.4 Xác suất lỗi bit

Xác suất lỗi bit là tham số hiệu năng quan trọng của hệ thống thông tin, liên quan trực tiếp đến chất lượng dịch vụ của hệ thống. Xác suất lỗi bit phải đi kèm với kiểu điều chế, mức điều chế và loại kênh truyền và cũng là tham số hiệu năng lý thuyết có mức độ tính toán phức tạp. Xác suất lỗi bit có mối liên hệ trực tiếp với xác suất lỗi symbol và tỷ lệ lỗi gógi.

Tỷ lệ lỗi bit là tỷ lệ số bit nhận sai với tổng số bit truyền trong khoảng thời gian quan sát. Tỷ lệ lỗi bit là một xấp xỉ của xác suất lỗi bit thông qua quá trình đo đạc thực tế hay mô phỏng Monte Carlo.

Xác suất lỗi bit của điều chế BPSK ở kênh truyền nhiễu trắng là [4]:

$$BER = Q\left(\sqrt{2\bar{\gamma}}\right). \quad (6.7)$$

Xác suất lỗi bit trung bình của điều chế BPSK ở kênh truyền fading Rayleigh được tính như sau:

$$\begin{aligned} \text{BER} &= \int_0^{\infty} Q\left(\sqrt{2\gamma}\right) f_{\gamma}(\gamma) d\gamma \\ &= \frac{1}{2} \left(1 - \sqrt{\frac{\bar{\gamma}}{\bar{\gamma} + 1}} \right). \end{aligned} \quad (6.8)$$

Ở vùng tỷ lệ trên nhiễu cao, áp dụng xấp xỉ $(1+x)^n \approx 1+nx$ cho giá trị x nhỏ, ta có thể xấp xỉ (6.8) như sau:

$$\text{BER} \approx \frac{1}{4\bar{\gamma}}. \quad (6.9)$$

Xác suất lỗi bit (và symbol) của những loại điều chế khác ở kênh truyền nhiễu trắng có thể tham khảo trong Bảng 6.1 của cuốn “Wireless Communications” của Andrea Goldsmith [4]. Khi đó, tỷ lệ lỗi bit trung bình của hệ thống ở kênh truyền fading Rayleigh có thể viết tổng quát như sau:

$$\begin{aligned} \text{BER} &= \int_0^{\infty} \alpha_m Q\left(\sqrt{\beta_m \gamma}\right) f_{\gamma_{\Sigma}}(\gamma) d\gamma \\ &= \frac{\alpha_m}{2} \left(1 - \sqrt{\frac{\beta_m \bar{\gamma}}{\beta_m \bar{\gamma} + 2}} \right) \end{aligned} \quad (6.10)$$

với α_m và β_m là các hệ số điều chế, phụ thuộc vào kiểu điều chế và mức điều chế.

6.2 Ước tính các tham số hiệu năng

6.2.1 Giới thiệu về phương pháp Monte Carlo

Monte Carlo là một phương pháp đánh giá/kiểm chứng tính chất hay đặc điểm hệ thống khảo sát bằng các biến giả ngẫu nhiên và hàm truyền trên máy tính. Yêu cầu khi áp dụng phương pháp Monte Carlo là phải biết được phân bố xác suất của dữ liệu và mô hình toán của hệ thống cần mô phỏng.

Phương pháp Monte Carlo cho hệ thống truyền thông thường qua ba bước như sau:

- Bước 1: Khởi tạo ngõ vào bằng các biến giả ngẫu nhiên.
- Bước 2: Đưa tín hiệu ngõ vào vào hàm thể hiện đặc tính của toàn hệ thống.
- Bước 3: Đo đặc kết quả đáp ứng ở ngõ ra và lặp lại hai bước trên đến khi kết quả mô phỏng là tin cậy.

Đặc điểm của phương pháp Monte Carlo khi ước lượng hiệu năng hệ thống viễn thông:

- Kết quả của phương pháp này càng chính xác (tiệm cận về kết quả đúng) khi số lần thử hay số lượng bước lặp tăng lên.
- Hiệu quả của phương pháp này tăng khi mức độ phức tạp của hệ thống tăng.
- Phù hợp để ước lượng hiệu năng các hệ thống truyền thông phức tạp, đặc biệt ở lớp vật lý và có thể chưa biết được hiệu năng chính xác.
- Thời gian mô phỏng phụ thuộc vào mức độ phức tạp và mức độ tính toán của mô hình mô phỏng. Đơn cử với các tham số hiệu năng như tỷ lệ lỗi bit, số lần thử phải ít nhất là 10^8 cho mức tỷ lệ lỗi bit là 10^{-7} .
- Khi số lần thử không đủ thì kết quả mô phỏng sẽ không chính xác, theo định lý giới hạn trung tâm.
- Có thể sử dụng kết quả mô phỏng để kiểm chứng hay định hướng cho các kết quả khi dùng phương pháp khác.

6.2.2 Ước tính tỷ số tín hiệu trên nhiễu trung bình

Tỷ số tín hiệu trên nhiễu trung bình có thể ước lượng bằng cách lấy trung bình của tỷ số tín hiệu trên nhiễu tức thời trong thời gian mô phỏng hay số lần mô phỏng. Gọi N là số lần mô phỏng, ta có

$$\bar{\gamma} = \frac{1}{N} \sum_{n=1}^N \gamma_n \quad (6.11)$$

với γ_n là tỷ số tín hiệu trên nhiễu tức thời tại lần thử n .

Ví dụ 6.1

Ước lượng tỷ số tín hiệu trên nhiễu trung bình của hệ thống SISO ở kênh truyền fading Rayleigh, có công suất phát là P , độ lợi kênh trung bình là Ω và so sánh với lý thuyết.

Giải: Mã nguồn Matlab cho Ví dụ 6.1 như sau:

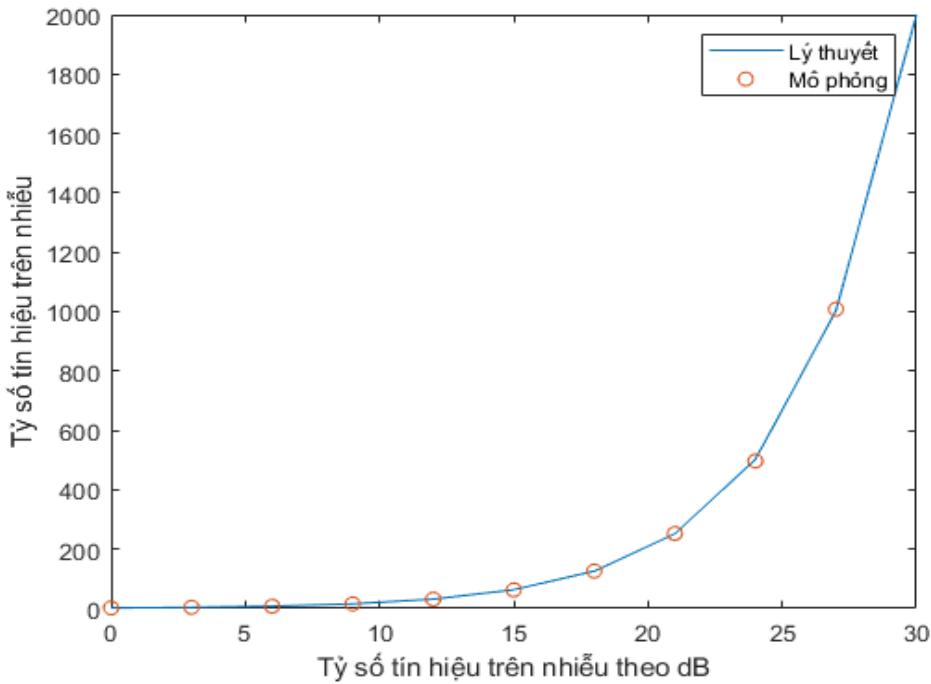
```
% Công suất phát theo dB
PdB = 0:3:30;
P = 10.^^(PdB/10);

% Độ lợi kênh trung bình
Omega = 2;

% Tỷ số tín hiệu trên nhiễu trung bình
gamma_a = P.*Omega;

% Mô phỏng
N = 10^4;
h = sqrt(Omega/2).*(randn(length(PdB),N) + 1i*randn(length(PdB),N));
gamma_s = sum(P(:).*abs(h).^2,2)/N;
```

```
% Vẽ đồ thị
figure(1);
set(gcf,'color','White');
plot(PdB,gamma_a,'-',PdB, gamma_s,'o');
legend('Lý thuyết', 'Mô phỏng');
xlabel('Tỷ số tín hiệu trên nhiễu trên nhiễu theo dB');
ylabel('Tỷ số tín hiệu trên nhiễu');
axis([0 30 0 2000])
```



Hình 6.1 Tỷ số tín hiệu trên nhiễu trung bình của hệ thống SISO trên kênh truyền fading Rayleigh của Ví dụ 6.1.

Ví dụ 6.2

Biết rằng tỷ số tín hiệu trên nhiễu trung bình của hệ thống kết hợp lựa chọn (selection combiner) hai kênh là $\bar{\gamma}_{SC} = \frac{3}{2}\bar{\gamma}$ với $\bar{\gamma}$ là tỷ số tín hiệu trên nhiễu trung bình của kênh. Hãy viết chương trình mô phỏng hệ thống và so sánh với kết quả lý thuyết.

Giải: Chương trình Matlab để xuất như sau.

```
% Công suất phát theo dB
PdB = 0:1:10;
P = 10.^ (PdB/10);
% Độ lợi kênh trung bình
Omega = 2;

%% Phân tích lý thuyết
g_ = P.*Omega;
gS_a = 1.5*g_;
```

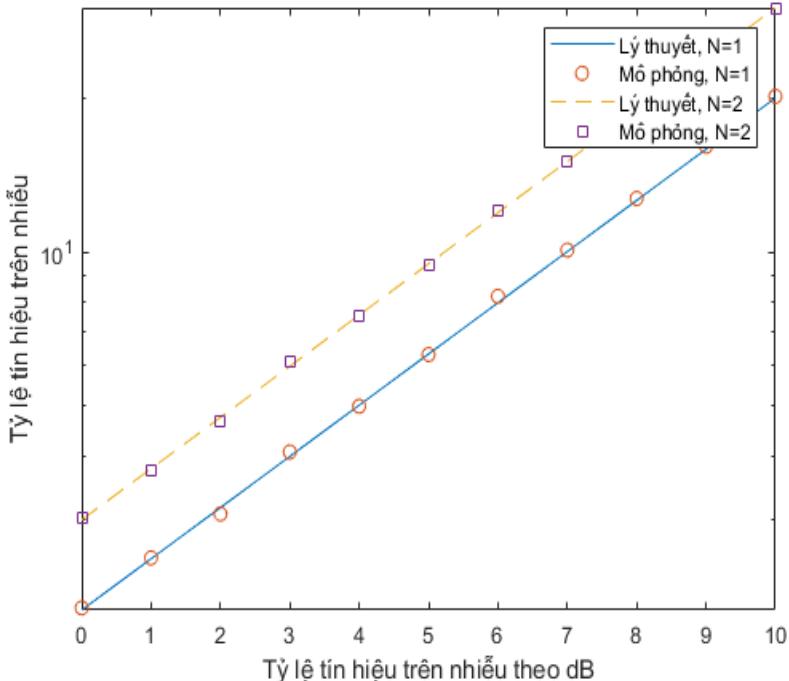
```

%% Thực hiện mô phỏng
% Số lượng phép thử
N = 10^4;
h1 = sqrt(Omega/2).*randn(length(PdB),N) + 1i*randn(length(PdB),N);
h2 = sqrt(Omega/2).*randn(length(PdB),N) + 1i*randn(length(PdB),N);

% Kỹ thuật phân tập thu lựa chọn anten
g_s = sum(P(:).*abs(h1).^2,2)/N;
gS_s = sum(P(:).*max(abs(h1).^2,abs(h2).^2),2)/N;

% Vẽ đồ thị
figure(1);
set(gcf,'color','White');
semilogy(PdB,g_s,'-',PdB, g_s,'o',PdB,gS_s,'--',PdB, gS_s,'s');
legend('Lý thuyết, N=1','Mô phỏng, N=1','Lý thuyết, N=2','Mô phỏng, N=2');
xlabel('Tỷ lệ tín hiệu trên nhiễu theo dB');
ylabel('Tỷ lệ tín hiệu trên nhiễu');

```



Hình 6.2: Tỷ số tín hiệu trên nhiễu trung bình của hệ thống SC hai kênh trên kênh truyền fading Rayleigh ở Ví dụ 6.2.

Ví dụ 6.3

Biết rằng tỷ số tín hiệu trên nhiễu trung bình của hệ thống kết hợp theo tỷ lệ tối ưu (maximal ratio combiner) N kênh là $\gamma_{\Sigma} = \sum_{n=1}^N \bar{\gamma}_n$ với $\bar{\gamma}_n$ là tỷ số tín hiệu trên nhiễu trung bình của kênh thứ n . Hãy viết chương trình mô phỏng hệ thống và so sánh với kết quả lý thuyết.

Giải: Chúng ta xem xét kỹ thuật MRC 3 kênh với độ lợi kênh truyền của từng kênh lần lượt là [1 2 3].

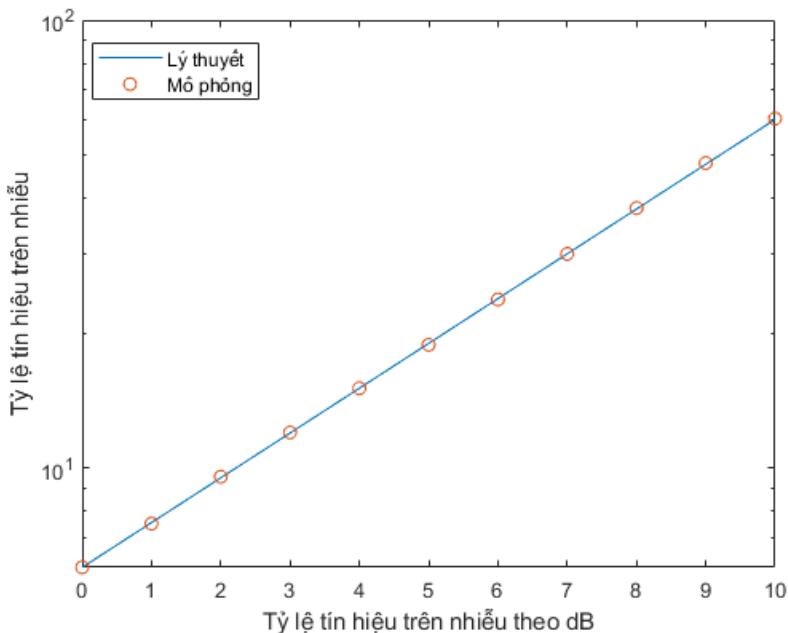
```
% Công suất phát theo dB
PdB = 0:1:10;
P = 10.^{PdB/10};
% Độ lợi kênh truyền trung bình
Omega = [1 2 3]; %
N = length(Omega);

% Lý thuyết
g_ = Omega(:).*P;
gMRC_a = sum(g_,1);

% Mô phỏng
N = 10^4; % Số luồng phép thử
h1 = sqrt(Omega(1)/2).*(randn(length(PdB),N) + 1i*randn(length(PdB),N));
h2 = sqrt(Omega(2)/2).*(randn(length(PdB),N) + 1i*randn(length(PdB),N));
h3 = sqrt(Omega(3)/2).*(randn(length(PdB),N) + 1i*randn(length(PdB),N));

% MRC
gMRC_s = P(:).*sum(abs(h1).^2 + abs(h2).^2 + abs(h3).^2,2)/N;

% Vẽ
figure(1);
set(gcf,'color','White');
semilogy(PdB,gMRC_a,'-',PdB,gMRC_s,'o');
legend('Lý thuyết', 'Mô phỏng');
xlabel('Tỷ lệ tín hiệu trên nhiễu theo dB');
ylabel('Tỷ lệ tín hiệu trên nhiễu');
```



Hình 6.3: Tỷ lệ tín hiệu trên nhiễu trung bình của hệ thống MRC trên kênh truyền fading Rayleigh.

6.2.3 Uớc tính dung lượng hệ thống

Tương tự như tỷ số tín hiệu trên nhiễu trung bình, dung lượng hệ thống tại nút thu có thể được ước lượng bằng cách lấy trung bình của dung lượng trên nhiễu tức thời theo số lần thử. Dung lượng trung bình của hệ thống khi thực hiện mô phỏng là:

$$\bar{C} = \frac{1}{N} \sum_{n=1}^N \log_2(1 + \gamma_n). \quad (6.12)$$

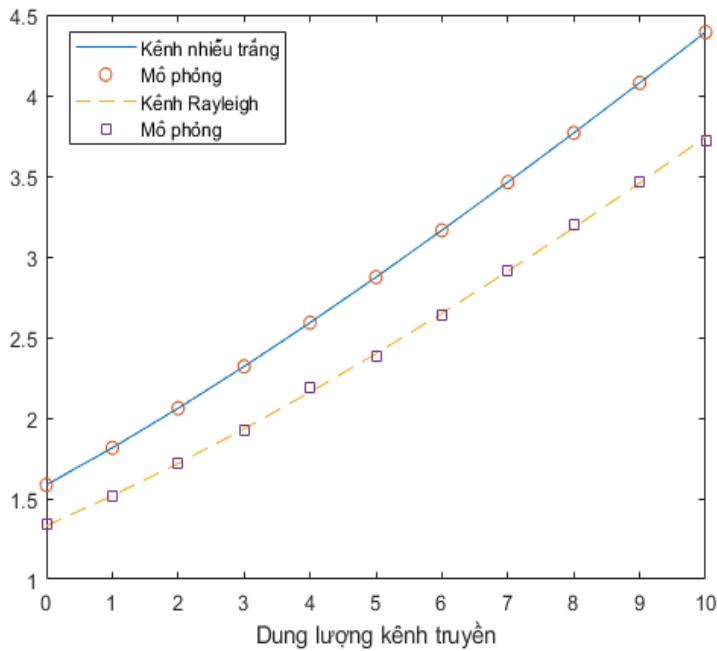
Ví dụ 6.4

Sử dụng mô hình hệ thống và kênh truyền tương tự như Ví dụ 6.1, ước lượng dung lượng hệ thống ở:

- Kênh truyền nhiễu trắng.
- Kênh truyền fading Rayleigh.
- So sánh với kết quả lý thuyết.

Giải: Trong Ví dụ 6.4, chúng ta sẽ lần lượt sử dụng các công thức (6.5) và (6.6) để tính toán dung lượng kênh truyền lý thuyết của kênh truyền nhiễu trắng và kênh truyền fading. Riêng đối với kết quả mô phỏng, chúng ta sử dụng công thức (6.12).

```
% Công suất phát theo dB  
PdB = 0:1:10;  
P = 10.^*(PdB/10);  
% Độ lợi kênh truyền  
Omega = 2;  
B = 1;  
  
% Kết quả phân tích  
g_ = P.*Omega;  
CA_a = log2(1 + g_);  
CR_a = -1/log(2).*exp(1./g_).*real(-expint(1./g_));  
  
% Mô phỏng  
% Số lượng phép thử  
N = 10^4;  
h = sqrt(Omega/2).*(randn(length(PdB),N) + 1i*randn(length(PdB),N));  
n = sqrt(1/2).*(randn(length(PdB),N) + 1i*randn(length(PdB),N));  
%  
g_A = P(:).*Omega.*ones(length(PdB),N);  
g_R = P(:).*abs(h).^2;  
  
CA_s = sum(B.*log2(1+ g_A),2)/N;  
CR_s = sum(B.*log2(1+ g_R),2)/N;  
  
% Vẽ đồ thị  
figure(1);  
set(gcf,'color','White');  
plot(PdB,CA_a,'-',PdB, CA_s,'o',PdB,CR_a,'--',PdB, CR_s,'s');  
legend('Kênh nhiễu trắng','Mô phỏng','Kênh Rayleigh','Mô phỏng');  
xlabel('Dung lượng kênh truyền');
```



Hình 6.4: Dung lượng kênh truyền nhiễu trắng và fading Rayleigh.

Từ Hình 6.4, ta có thể thấy rằng kết quả lý thuyết và kết quả mô phỏng gần như trùng khớp với nhau ở toàn bộ vùng tỷ lệ tín hiệu trên nhiễu xem xét và dung lượng của hệ thống hoạt động ở kênh truyền nhiễu trắng lớn hơn dung lượng của hệ thống ở kênh truyền fading Rayleigh ở cùng điều kiện.

Một lỗi sai thường mắc phải trong khi tính dung lượng kênh truyền là khi sử dụng hàm `expint()` trong Matlab. Định nghĩa của hàm `expint()` trong Matlab khác với định nghĩa của hàm `expint()` trong phần mềm Mathematica hay trong các sách tra cứu toán học. Khi không để ý, sẽ dẫn đến kết quả lý thuyết và kết quả mô phỏng không khớp.

6.2.4 Ước tính xác suất dùng

Trong mô phỏng, xác suất dùng là tỷ lệ số lần mà dung lượng Shannon của hệ thống nhỏ hơn tốc độ truyền mong muốn trên tổng số lần truyền (N), cụ thể:

$$OP = \frac{N_f}{N} \quad (6.13)$$

với N_f là số lần mà $\log_2(1 + \gamma_n) < R$.

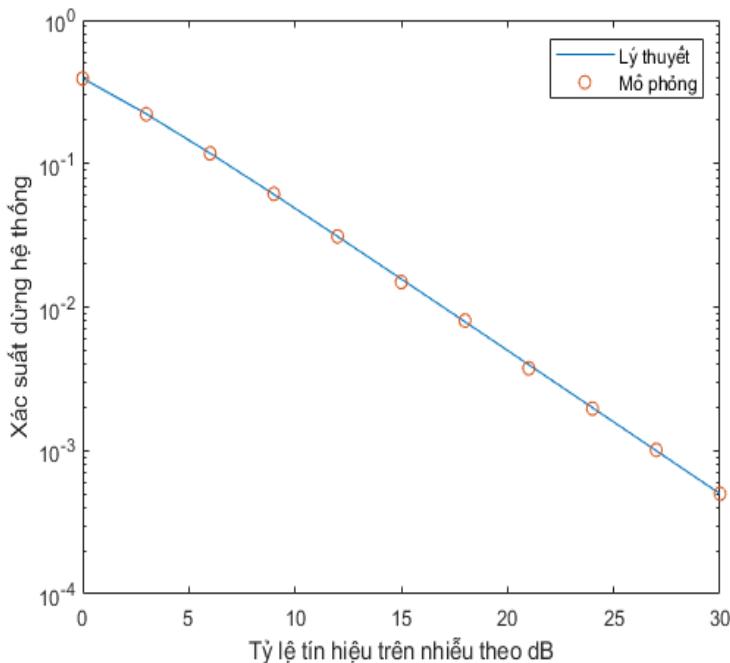
Ví dụ 6.5

Sử dụng mô hình hệ thống và kênh truyền tương tự như Ví dụ 6.1, mô phỏng xác suất dùng của hệ thống ở kênh truyền fading Rayleigh.

Giải: Chúng ta sẽ thực hiện mô phỏng xác suất dùng cho hệ thống từ 0 đến 30 dB với bước khảo sát là 3dB. Với kết quả lý thuyết, chúng ta sử dụng công thức (6.3). Với kết quả mô phỏng, chúng ta thực hiện các bước sau:

- Bước 1: Tạo độ lợi kênh truyền.
- Bước 2: Tính tỷ số tín hiệu trên nhiễu.
- Bước 3: Đếm số lần dung lượng kênh truyền nhỏ hơn tốc độ truyền cho trước và tính OP.
- Bước 4: Vẽ đồ thị OP lý thuyết và mô phỏng.

```
% Công suất phát theo dB
PdB = 0:3:30;
P = 10.^PdB/10;
% Độ lợi kênh truyền trung bình
Omega = 2;
R = 1;
gth = 2^R - 1;
% Phân tích lý thuyết
gamma_ = P.*Omega;
OP_a = 1- exp(-gth./gamma_);
% Mô phỏng
% Số lượng phép thử
N = 10^5;
h = sqrt(Omega/2).*(randn(length(PdB),N) + 1i*randn(length(PdB),N));
gamma = P(:).*abs(h).^2;
OP_s = sum(log2(1 + gamma) < R,2)/N;
% Vẽ hình
figure(1);
set(gcf,'color','White');
semilogy(PdB,OP_a,'-',PdB, OP_s,'o');
legend('Lý thuyết','Mô phỏng');
xlabel('Tỷ lệ tín hiệu trên nhiễu theo dB');
ylabel('Xác suất dừng hệ thống');
```



Hình 6.5: Xác suất dừng hệ thống SISO ở kênh truyền fading Rayleigh của Ví dụ 6.5.

6.2.5 Ước tính tỷ lệ lỗi bit/symbol

Khi ước tính tỷ lệ lỗi bit, chúng ta cần tính tỷ lệ của số lượng bit sai (N_e) nhận tại máy thu trên tổng số lượng bit phát đi từ máy phát (N_t). Khi đó, ta có:

$$\text{BER} = \frac{N_e}{N_t}. \quad (6.14)$$

Áp dụng cách tương tự như vậy để tính tỷ lệ lỗi symbol, cụ thể:

$$\text{SER} = \frac{N_e}{N_t}. \quad (6.15)$$

Trong (6.15), N_e là số lượng symbol sai và N_t là tổng số lượng symbol phát. Khi hệ thống sử dụng mã Gray, ta có thể xấp xỉ SER từ BER như sau:

$$\text{SER} \approx \frac{\text{BER}}{\log_2 M} \quad (6.16)$$

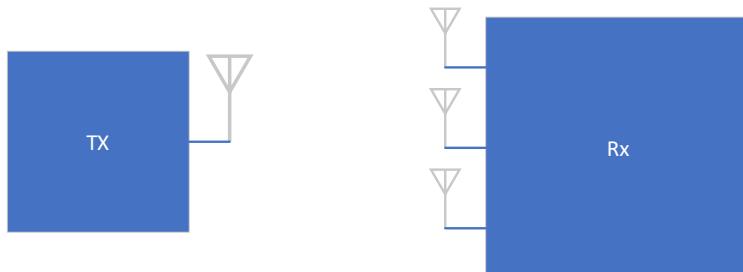
với M là số mức điều chế mà hệ thống sử dụng.

6.3 Thực hiện mô phỏng hệ thống viễn thông

6.3.1 Mô phỏng hệ thống phân tập thu

6.3.1.1 Hệ thống kết hợp lựa chọn (Selection combiner)

Trong phần này, chúng ta sẽ thực hiện mô phỏng hệ thống phân tập thu lựa chọn kết hợp (selection combining). Xem xét một hệ thống phân tập thu như Hình 6.6 với một anten phát và N anten thu.



Hình 6.6: Hệ thống phân tập thu sử dụng kỹ thuật kết hợp lựa chọn.

Giả sử công suất phát của máy phát là P , tín hiệu nhận được tại máy thu là:

$$r_i = \sqrt{P} h_i x + n_i \quad (6.17)$$

với h_i là hệ số kênh truyền từ anten phát đến anten thứ i của máy thu, n_i là nhiễu trắng lại anten máy thu có công suất là N_0 . Tỷ số tín hiệu trên nhiễu tại anten thứ i có dạng như sau:

$$\gamma_i = \frac{P|h_i|^2}{N_0}. \quad (6.18)$$

Ở kênh truyền fading Rayleigh, hàm CDF và PDF của γ_i lần lượt có dạng như sau:

$$f_{\gamma_i}(\gamma) = \frac{1}{\bar{\gamma}} e^{-\frac{\gamma}{\bar{\gamma}}}, \quad (6.19)$$

$$F_{\gamma_i}(\gamma) = 1 - e^{-\frac{\gamma}{\bar{\gamma}}}. \quad (6.20)$$

Giả sử rằng các kênh truyền từ máy phát đến máy thu là giống nhau nên tỷ số tín hiệu trên nhiễu trung bình của các nhánh là bằng nhau, cụ thể là:

$$\bar{\gamma}_1 = \bar{\gamma}_2 = L = \bar{\gamma}_N = \bar{\gamma}. \quad (6.21)$$

Máy thu sử dụng kỹ thuật kết hợp lựa chọn, gọi γ_Σ là tỷ số tín hiệu trên nhiễu tương đương sau bộ kết hợp, ta có:

$$\gamma_\Sigma = \max_{i=1,\dots,N} \gamma_i. \quad (6.22)$$

Giả sử khoảng cách giữa các anten thu ở máy thu là đủ lớn và do đó các γ_i là độc lập với nhau, ta có hàm phân bố xác suất tích lũy cho γ_Σ là như sau:

$$\begin{aligned} F_{\gamma_\Sigma}(\gamma) &= \prod_{i=1}^N F_{\gamma_i}(\gamma) \\ &= \left(1 - e^{-\frac{\gamma}{\bar{\gamma}}}\right)^N. \end{aligned} \quad (6.23)$$

Dẫn đến hàm PDF cho γ_Σ là như sau:

$$\begin{aligned} f_{\gamma_\Sigma}(\gamma) &= \frac{d}{d\gamma} \left[\prod_{i=1}^N F_{\gamma_i}(\gamma) \right] \\ &= \sum_{n=1}^N (-1)^{n-1} \binom{N}{n} \frac{n}{\bar{\gamma}} e^{-\frac{n\gamma}{\bar{\gamma}}}. \end{aligned} \quad (6.24)$$

Sử dụng biến đổi nhị phân, ta có thể viết $F_{\gamma_\Sigma}(\gamma)$ như sau:

$$F_{\gamma_\Sigma}(\gamma) = 1 - \sum_{n=1}^N (-1)^{n-1} \binom{N}{n} \exp\left(-\frac{n\gamma}{\bar{\gamma}}\right). \quad (6.25)$$

Xác suất dừng của hệ thống SC N anten có thể được tính như sau:

$$\begin{aligned}
 \text{OP} &= F_{\gamma_{\Sigma}}(\gamma_{th}) \\
 &= \left[1 - \exp\left(-\frac{\gamma_{th}}{\bar{\gamma}}\right) \right]^N \\
 &= 1 - \sum_{n=1}^N (-1)^{n-1} \binom{N}{n} \exp\left(-\frac{n\gamma_{th}}{\bar{\gamma}}\right).
 \end{aligned} \tag{6.26}$$

Ví dụ 6.6

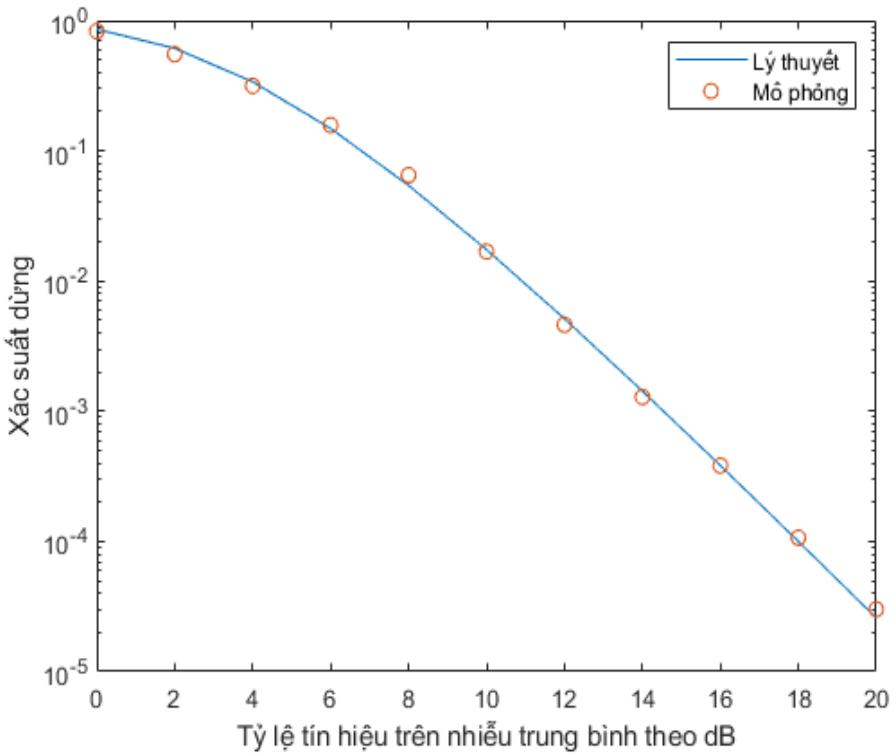
Hãy mô phỏng hệ thống phân tập thu lựa chọn anten thu (selection combining) với 3 anten và vẽ đồ thị xác suất dùng theo tỷ số tín hiệu trên nhiễu theo dB.

Giải: Ta sẽ sử dụng công thức (6.26) cho kết quả lý thuyết. Chúng ta sẽ chọn số bit mô phỏng gấp 30 lần nghịch đảo của xác suất dùng lý thuyết. Do số lần thử là một số nguyên dương, nên chúng ta dùng hàm ceil() để làm tròn số.

```
% Tham số kênh truyền và hệ thống
N = 3;
SNRdB = 0:2:20;
SNR = 10.^{SNRdB/10};
gth = 3;
lambda = 1;

% Lý thuyết
g = lambda*SNR;
OP_a = (1 - exp(-gth./g)).^N;

% Mô phỏng
Flen = ceil(30./OP_a);
for idx = 1:length(SNRdB)
    h = sqrt(SNR(idx)).*lambda/2).*(randn(N,Flen(idx)) + 1i*randn(N,Flen(idx)));
    gmax = max(abs(h).^2,[],1);
    OP_s(idx) = sum(gmax < gth)./Flen(idx);
end
semilogy(SNRdB,OP_a,'-',SNRdB,OP_s,'o');
xlabel('Tỷ lệ tín hiệu trên nhiễu trung bình theo dB');
ylabel('Xác suất dùng');
legend('Lý thuyết','Mô phỏng');
```



Hình 6.7: Xác suất dùng hệ thống của hệ thống SC 3 kênh ở kênh truyền fading Rayleigh.

Ví dụ 6.7

Mô phỏng tỷ lệ lỗi bit của điều chế BPSK trên kênh truyền fading Rayleigh với phân tập thu SC.

Giải: Từ công thức (6.24), ta có tỷ lệ lỗi bit của điều chế BPSK được tính như sau:

$$\begin{aligned} \text{BER} &= \int_0^\infty Q(\sqrt{2\gamma}) f_{\gamma_\Sigma}(\gamma) d\gamma \\ &= \sum_{n=1}^N (-1)^{n-1} \binom{N}{n} \frac{1}{2} \left(1 - \sqrt{\frac{\bar{\gamma}}{n + \bar{\gamma}}} \right). \end{aligned} \quad (6.27)$$

```
% Công suất phát theo dB
clear all
SNRdB = (0:2:20)';
SNR = 10.^{(SNRdB/10)};
K = length(SNRdB);

% Số lượng anten thu
N = 2;
% Độ lợi kênh truyền - 3 kênh giống nhau
Omega = 2;
% Điều chế BPSK
M = 2;
```

```

% Lý thuyết
g_ = log2(M).*Omega.*SNR;
G = sqrt(g_./(1 + g_));
BER_a = zeros(size(SNRdB));

for n = 1:N
    BER_a = BER_a + (-1).^(n-1).*nchoosek(N,n).*1/2.*(1 - sqrt(g_./(n + g_)));
end

% Mô phỏng
NoB = 10^5;

BER_s = zeros(size(SNRdB));
for idx = 1:length(SNRdB)

    s = randi([0 M-1],1,NoB);
    % Điều chế máy phát sử dụng mã Gray
    Tx = pskmod(s,M,0,'gray');
    Ps = log2(M).*SNR(idx);
    % Tạo kênh truyền
    h = sqrt(Ps.*Omega/2).* (randn(N,NoB) + 1i*randn(N,NoB));

    % Phân tập thu SC
    [h_,Imax] = max(abs(h),[],1);
    hmax = h(h_==abs(h))';

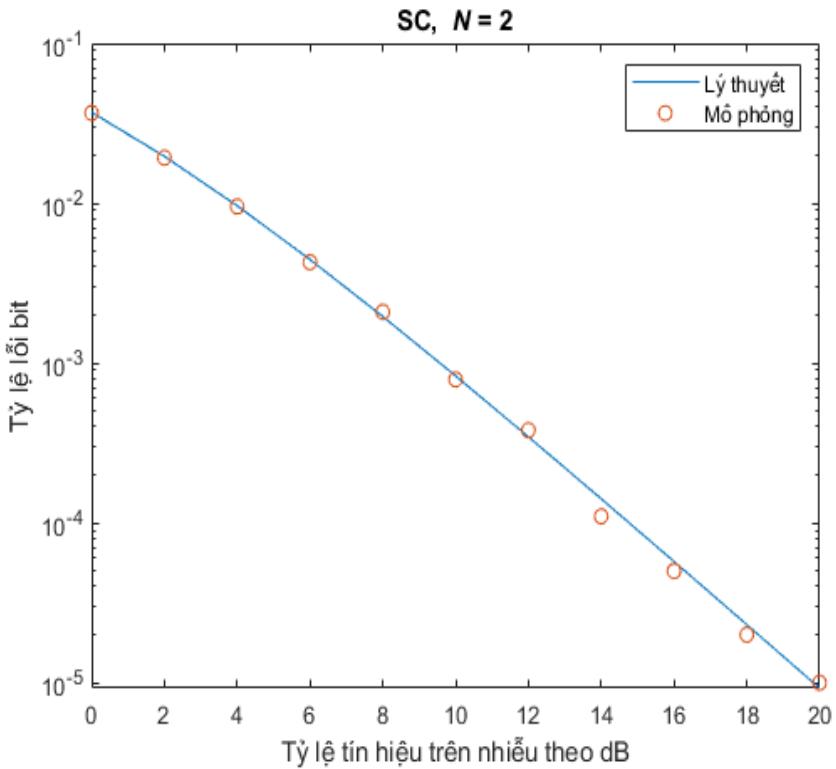
    % Tín hiệu tại máy thu
    n = sqrt(1/2).* (randn(1,NoB) + 1i*randn(1,NoB));
    Rx = hmax.*Tx + n;

    % Giải điều chế
    Tx_ = Rx./hmax;
    s_ = pskdemod(Tx_,M,0,'gray');

    % Tính tỷ lệ lỗi bit
    BER_s(idx) = biterr(s,s_)/(log2(M)*NoB);
end

% Vẽ
figure(1);
set(gcf,'color','White');
semilogy(SNRdB,BER_a,'-',SNRdB,BER_s,'o');
legend('Lý thuyết','Mô phỏng');
xlabel('Tỷ lệ tín hiệu trên nhiễu theo dB');
ylabel('Tỷ lệ lỗi bit');
title('SC, {ít N} = 2')
set(gcf,'color','w');

```



Hình 6.8: Tỷ lệ lỗi bit của hệ thống SC cho điều chế BPSK
ở kênh truyền fading Rayleigh của Ví dụ 6.7.

6.3.1.2 Kỹ thuật kết hợp phân tập theo tỷ lệ tối ưu

Lý thuyết của kỹ thuật kết hợp phân tập theo tỷ lệ tối ưu (MRC) có thể tham khảo ở mục 7.2.4 của sách Goldsmith. Phần này chỉ trình bày lại các công thức hiệu năng chính của MRC để phục vụ cho phần mô phỏng trên Matlab.

Gọi r_m là tín hiệu thu được tại anten thứ m , ta có:

$$r_m = \sqrt{P}h_m x + n_m \quad (6.28)$$

với x là tín hiệu phát sau điều chế với $E\{|x|^2\} = 1$; P là công suất phát ở máy thu, h_m là hệ số kênh truyền từ máy phát đến anten máy thu thứ m với $E\{|h_m|^2\} = \Omega \quad \forall m$ và n_m là nhiễu trăng tại anten thứ m với trung bình bằng không và phương sai N_0 .

Giả sử máy thu biết thông tin kênh truyền, máy thu sẽ thực hiện phục hồi x là \hat{x} như sau:

$$\hat{x} = \frac{\sum_{m=1}^M (\sqrt{P}h_m)^* r_m}{P \sum_{m=1}^M |h_m|^2} = x + \frac{\sum_{m=1}^M (\sqrt{P}h_m)^* n_m}{P \sum_{m=1}^M |h_m|^2}. \quad (6.29)$$

Gọi $\gamma_1, \dots, \gamma_M$ là tỷ số tín hiệu trên nhiễu tại anten số 1 đến anten số M , ta có tỷ số tín hiệu trên nhiễu tương đương sau bộ kết hợp theo tỷ số tối ưu MRC là:

$$\gamma_{\Sigma} = \sum_{m=1}^M \gamma_m. \quad (6.30)$$

Xem xét kênh truyền fading Rayleigh độc lập và đồng nhất, cụ thể là:

$$\bar{\gamma}_m = \bar{\gamma} \quad \forall m. \quad (6.31)$$

Hàm PDF của γ_{Σ} có dạng như sau [14]:

$$f_{\gamma_{\Sigma}}(\gamma) = \frac{\gamma^{M-1} e^{-\frac{\gamma}{\bar{\gamma}}}}{\bar{\gamma}^M (M-1)!}, \quad \gamma \geq 0. \quad (6.32)$$

Xác suất của hệ thống MRC với M anten là:

$$OP = \Pr(\gamma_{\Sigma} \leq \gamma_{th}) = \int_0^{\gamma_{th}} f_{\gamma_{\Sigma}}(\gamma) d\gamma = 1 - e^{-\frac{\gamma_{th}}{\bar{\gamma}}} \sum_{k=1}^M \frac{1}{(k-1)!} \left(\frac{\gamma_{th}}{\bar{\gamma}} \right)^{k-1}. \quad (6.33)$$

với γ_{th} là ngưỡng dừng cho trước của hệ thống.

Giả sử hệ thống sử dụng điều chế BPSK, ta có tỷ lệ lỗi bit trung bình như sau [4]:

$$BER = \left(\frac{1-\Gamma}{2} \right)^M \sum_{m=0}^{M-1} \binom{M-1+m}{m} \left(\frac{1+\Gamma}{2} \right)^m \quad (6.34)$$

$$\text{với } \Gamma = \sqrt{\frac{\bar{\gamma}}{1+\bar{\gamma}}}. \quad$$

Ví dụ 6.8

Mô phỏng xác suất dừng của hệ thống MRC với số lượng anten M cho trước ở kênh truyền fading Rayleigh. Kiểm chứng với công thức lý thuyết.

Giải: Chúng ta sẽ áp dụng công thức (6.33) để tính xác suất dừng của hệ thống ở kênh truyền fading Rayleigh.

```
% Thông số kênh truyền và hệ thống
```

```
M = 2;
```

```
SNRdB = 0:2:20;
```

```
SNR = 10.^^(SNRdB/10);
```

```
gth = 3;
```

```
lambda = 1;
```

```
% Lý thuyết
```

```
g = lambda*SNR;
```

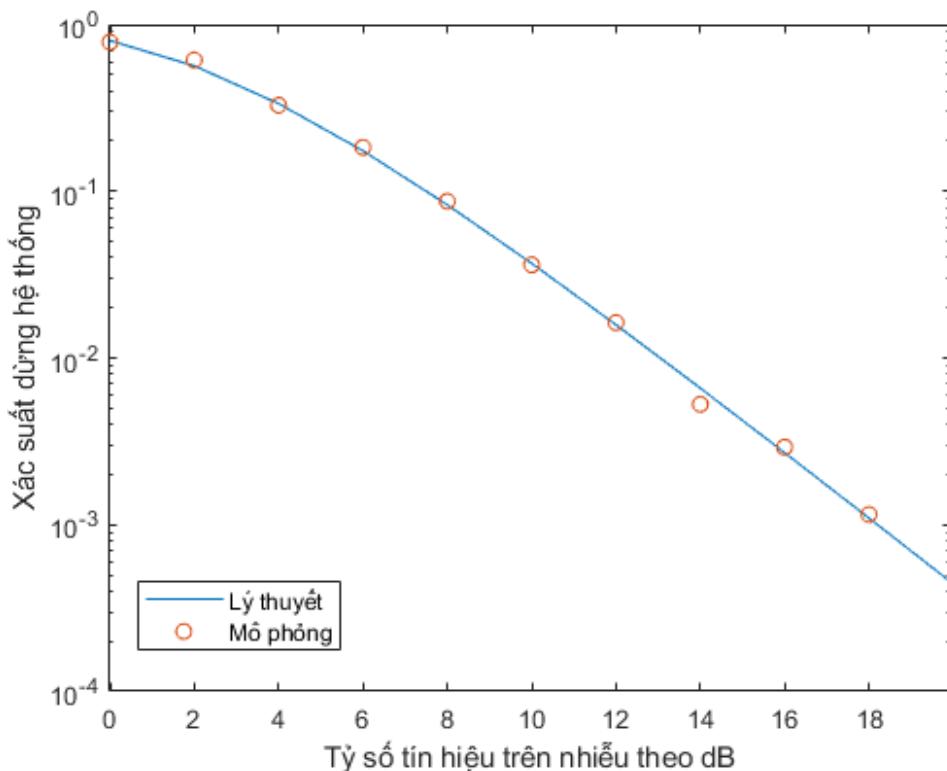
```
OP_a = zeros(size(SNRdB));
```

```

for k=1:M
    OP_a = OP_a + 1/factorial(k-1).*(gth./g).^(k-1);
end
OP_a = 1 - exp(-gth./g).*OP_a;

% Mô phỏng
Flen = ceil(100./OP_a);
for idx = 1:length(SNRdB)
    h = sqrt(SNR(idx).*lambda/2).*((randn(M,Flen(idx)) + 1i*randn(M,Flen(idx))));
    gSum = sum(abs(h).^2,1);
    OP_s(idx) = sum(gSum < gth)./Flen(idx);
end
semilogy(SNRdB,OP_a,'-',SNRdB,OP_s,'o');
xlabel('Tỷ số tín hiệu trên nhiễu theo dB');
ylabel('Xác suất dùng hệ thống');
legend('Lý thuyết','Mô phỏng');

```



Hình 6.9: Xác suất dùng của hệ thống kết hợp phân tập MRC với 2 anten ở kênh truyền fading Rayleigh.

Ví dụ 6.9

Mô phỏng tính tỷ lệ lỗi bit trung bình của hệ thống MRC cho điều chế BPSK.

- a. So sánh với công thức lý thuyết.
- b. Nhận xét.

Giải: Chúng ta dùng công thức (6.34) để tính xác suất lỗi bit lý thuyết.

```

% Tỷ số tín hiệu trên nhiễu theo dB
SNRdB = (0:2:20)';
SNR = 10.^{SNRdB/10};
K = length(SNRdB);

% Số lượng anten thu
N = 2;
% Độ lợi kênh truyền - 3 kênh giống nhau
Omega = 2;
% Điều chế BPSK
M = 2;

% Lý thuyết
g_ = log2(M).*Omega.*SNR;
G = sqrt(g_./(1 + g_));
BER_a = zeros(size(SNRdB));
for m = 0:N-1
    BER_a = BER_a + nchoosek(N - 1 + m,m).*((1 + G)/2).^m;
end
BER_a = ((1 - G)/2).^N.*BER_a;

% Mô phỏng
NoB = 10^5;

BER_s = zeros(size(SNRdB));
for idx = 1:length(SNRdB)
    s = randi([0 M-1],1,NoB);
    % Điều chế máy phát sử dụng mã Gray
    Tx = pskmod(s,M,0,'gray');
    Ps = log2(M).*SNR(idx);
    % Tạo kênh truyền
    h = sqrt(Ps.*Omega/2).*(randn(N,NoB) + 1i*randn(N,NoB));
    n = sqrt(1/2).*(randn(N,NoB) + 1i*randn(N,NoB));

    % Tín hiệu tại máy thu
    Rx = h.*Tx + n;

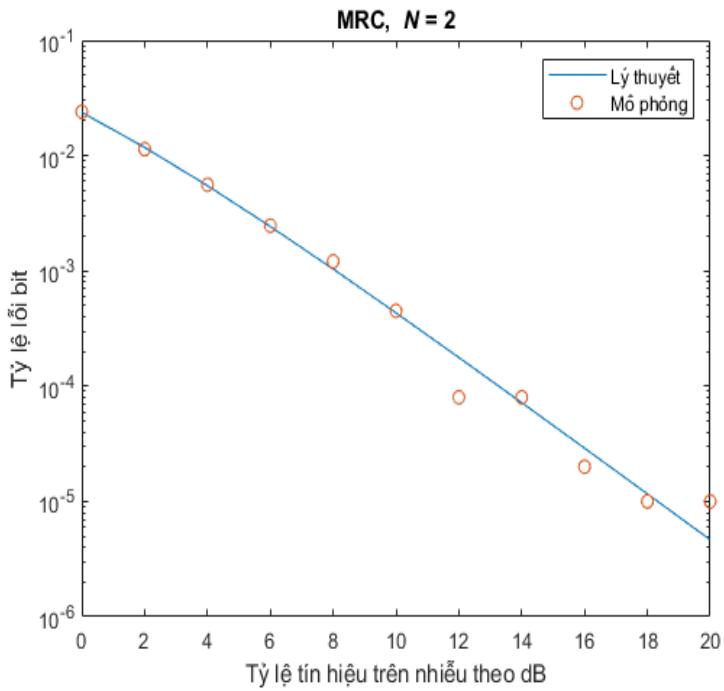
    % Phân tập thu MRC
    Tx_ = sum(conj(h).*Rx,1)./sum(abs(h).^2,1);

    % Giải điều chế
    s_ = pskdemod(Tx_,M,0,'gray');

    % Tính tỷ lệ lỗi bit
    BER_s(idx) = biterr(s,s_)/(log2(M)*NoB);
end

% Vẽ
figure(1);
set(gcf,'color','White');
semilogy(SNRdB,BER_a,'-',SNRdB,BER_s,'o');
legend('Lý thuyết','Mô phỏng');
xlabel('Tỷ lệ tín hiệu trên nhiễu theo dB');
ylabel('Tỷ lệ lỗi bit');
title('MRC, {|it N| = 2}')
set(gcf,'color','w');

```

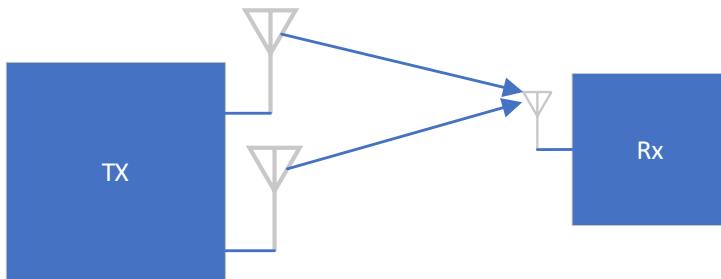


Hình 6.10: Tỷ lệ lỗi bit của hệ thống MRC với 1 anten phát và 2 anten thu cho điều chế BPSK ở kênh truyền fading Rayleigh.

6.3.2 Mô phỏng hệ thống mã không gian thời gian (phân tập phát)

Mã không gian thời gian là một kỹ thuật phân tập phát cho phép máy phát có nhiều anten phát có thể đạt được độ lợi phân tập như hệ thống phân tập thu. Trong phần này, chúng ta sẽ thực hiện mô phỏng kỹ thuật phân tập phát Alamouti và kiểm chứng với kết quả phân tích lý thuyết [15].

Xem xét mô hình hệ thống mã không gian thời gian Alamouti với 2 anten phát và 1 anten thu như Hình 6.11.



Hình 6.11: Mô hình mã không gian thời gian Alamouti

Gọi s_1 và s_2 lần lượt là tín hiệu truyền từ anten 1 và anten 2 trong khe thời gian thứ nhất. Tuy nhiên, trong khe thời gian thứ hai, anten 1 sẽ truyền $-s_2^*$ và anten 2 sẽ truyền s_1^* với $(\cdot)^*$ là toán tử lấy liên hiệp phức. Bảng 6.1 trình bày cụ thể theo thời gian và không gian của kỹ thuật truyền mã Alamouti.

Bảng 6.1: Bảng mã Alamouti theo không gian và theo thời gian.

	Khe thời gian 1	Khe thời gian 2
Anten 1	s_1	$-s_2^*$
Anten 2	s_2	s_1^*

Gọi h_1 và h_2 lần lượt là hệ số kênh truyền từ anten thứ nhất và thứ hai đến máy thu. Giả sử rằng kênh truyền là không thay đổi trong hai khe thời gian liên tiếp, nghĩa là:

$$h_1(t) = h_1(t+T) \quad (6.35)$$

$$h_2(t) = h_2(t+T) \quad (6.36)$$

với T là chu kỳ của một symbol.

Ta có thể viết tín hiệu nhận tại máy thu ở khe thời gian thứ nhất như sau:

$$r_1 = h_1 s_1 + h_2 s_2 + n_1. \quad (6.37)$$

Tương tự như vậy, ta có tín hiệu nhận tại khe thời gian thứ hai là:

$$r_2 = h_2 (-s_2^*) + h_1 s_1^* + n_2. \quad (6.38)$$

Tại máy thu, giả sử rằng máy thu có thông tin kênh truyền hoàn hảo, nên ta có tín hiệu kết hợp tại máy thu như sau:

$$\%_1 = h_1^* r_1 + h_2^* r_2 \quad (6.39)$$

$$\%_2 = h_2^* r_1 - h_1^* r_2 \quad (6.40)$$

Thay thế (6.37) và (6.38) lần lượt vào (6.39) và (6.40), sau quá trình biến đổi phù hợp, ta viết lại $\%_1$ và $\%_2$ như sau:

$$\%_1 = \left(|h_1|^2 + |h_2|^2 \right) s_1 + h_1^* n_1 + h_2^* n_2, \quad (6.41)$$

$$\%_2 = \left(|h_1|^2 + |h_2|^2 \right) s_2 - h_1^* n_2 + h_2^* n_1. \quad (6.42)$$

Tại máy thu, tín hiệu đầu vào của bộ giải điều chế cho s_1 và s_2 lần lượt sẽ là:

$$\hat{s}_1 = \frac{\hat{s}_1}{|h_1|^2 + |h_2|^2} = s_1 + \frac{h_1^* n_1 + h_2^* n_2}{|h_1|^2 + |h_2|^2}, \quad (6.43)$$

$$\hat{s}_2 = \frac{\%_2}{|h_1|^2 + |h_2|^2} = s_2 + \frac{-h_1^* n_2 + h_2^* n_1}{|h_1|^2 + |h_2|^2}. \quad (6.44)$$

Tỷ số tín hiệu trên nhiễu của s_1 và s_2 là:

$$\gamma_{\Sigma} = \frac{|h_1|^2 + |h_2|^2}{N_0}. \quad (6.45)$$

Do tỷ số tín hiệu trên nhiễu của mã không gian thời gian Alamouti tương tự như với bộ kết hợp phân tập thu theo tỷ lệ tối ưu MRC, nên có cùng công thức tính xác suất dùng và tỷ lệ lỗi bit trung bình. Điểm khác biệt duy nhất của hai hệ thống là công suất phát, với hệ thống MRC dùng công suất P cho 1 anten phát, thì hệ thống Alamouti là $P/2$ cho từng anten phát. Do đó, xác suất dùng hay tỷ lệ lỗi bit của hệ thống Alamouti sẽ nhỏ hơn hệ thống MRC 3 dB.

Ví dụ 6.10

Mô phỏng hệ thống mã không gian thời gian hai anten phát và một anten thu với điều chế BPSK ở kênh truyền fading Rayleigh. Tính tỷ lệ lỗi bit trung bình của hệ thống và so sánh với lý thuyết.

Giải: Chương trình Matlab cho mã không gian thời gian Alamouti như sau và kết quả là Hình 6.12.

```
% Tỷ số tín hiệu trên nhiễu theo dB
SNRdB = 0:1:10;
SNR = 10.^ (SNRdB/10);
% Độ lợi kênh truyền trung bình
Omega = 2;
% Mức điều chế, BPSK M=2
M = 2;
% Hai anten phát
K = 2;

% Lý thuyết
G = sqrt(Omega.*SNR/2./(1 + Omega.*SNR/2));
BER_a = zeros(size(SNRdB));
for m = 0:K-1
    BER_a = BER_a + nchoosek(K - 1 + m,m).*((1 + G)/2).^m;
end
BER_a = ((1 - G)/2).^K.*BER_a;

% Mô phỏng
N = 10^5;
BER_s = zeros(size(SNRdB));
for idx = 1:length(SNRdB)
    s = randi([0 M-1],1,2*N);

    % Transmitter
    Tx = pskmod(s,M,0,'gray');
    s1 = Tx(1:2:end);
    s2 = Tx(2:2:end);

    % Channel
    Ps = log2(M).*SNR(idx)/2;
    h1 = sqrt(Ps.*Omega/2).*(randn(1,N) + 1i*randn(1,N));
    h2 = sqrt(Ps.*Omega/2).*(randn(1,N) + 1i*randn(1,N));
    Rx = h1.*s1 + h2.*s2;
    BER_s(idx) = sum(Rx~=s)/N;
end
```

```

h2 = sqrt(Ps.*Omega/2).*(randn(1,N) + 1i*randn(1,N));

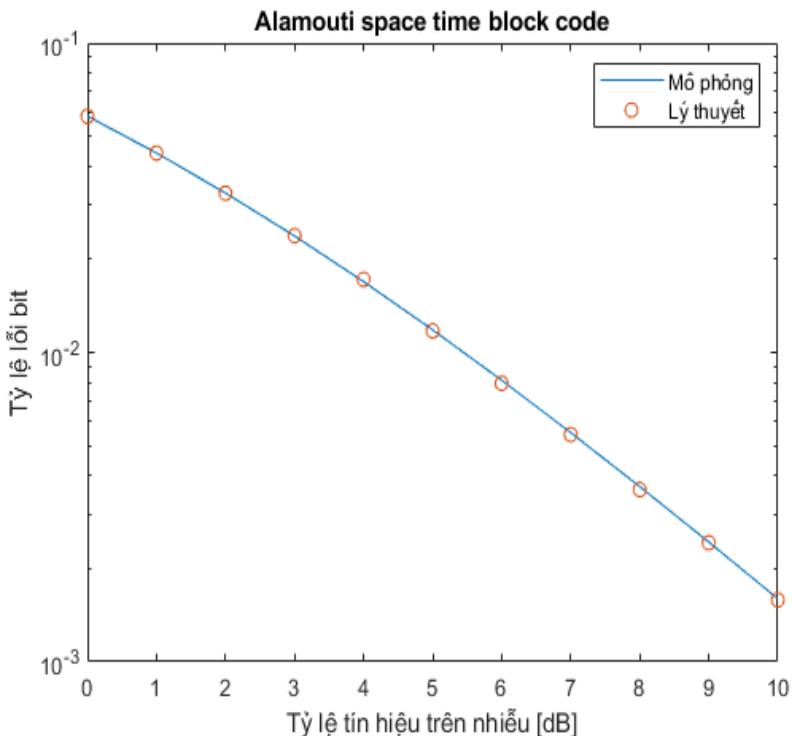
n1 = sqrt(1/2).*(randn(1,N) + 1i*randn(1,N));
n2 = sqrt(1/2).*(randn(1,N) + 1i*randn(1,N));
% Receiver
r1 = h1.*s1 + h2.*s2 + n1;
r2 = h1.*(-conj(s2)) + h2.*conj(s1) + n2;
r1_ = (conj(h1).*r1 + h2.*conj(r2))./(abs(h1).^2 + abs(h2).^2);
r2_ = (conj(h2).*r1 - h1.*conj(r2))./(abs(h1).^2 + abs(h2).^2);

s1_ = pskdemod(r1_,M,0,'gray');
s2_ = pskdemod(r2_,M,0,'gray');

%
s_(1:2:2*N) = s1_;
s_(2:2:2*N) = s2_;
BER_s(idx) = BER_s(idx) + biterr(s,s_)/(log2(M)*2*N);
end

% Vẽ đồ thị
figure(1);
set(gcf,'color','White');
semilogy(SNRdB,BER_a,'-',SNRdB,BER_s,'o');
legend('Mô phỏng','Lý thuyết');
xlabel('Tỷ lệ tín hiệu trên nhiễu [dB]');
ylabel('Tỷ lệ lỗi bit');
title('Alamouti space time block code')

```



Hình 6.12: Tỷ lệ lỗi bit của hệ thống mã không gian thời gian với 2 anten phát và 1 anten thu và điều chế BPSK ở kênh truyền fading Rayleigh.

6.3.3 Mô phỏng hệ thống truyền thích ứng

Kỹ thuật truyền thích ứng là một trong những kỹ thuật quan trọng cho các hệ thống vô tuyến thé hệ mới, cho phép cải thiện hiệu suất phô tần một cách đáng kể. Nói một cách khác, với một băng thông cho trước, hệ thống không sử dụng kỹ thuật điều chế thích ứng chỉ sử dụng một kiểu điều chế với một mức điều chế như khi thiết kế bát kẽ chất lượng kênh truyền tốt hay xấu. Ý tưởng của kỹ thuật truyền thích ứng là cho phép điều chỉnh loại điều chế cũng như mức điều chế để tối đa hiệu suất phô tần miễn là tỷ lệ lỗi bit trung bình của hệ thống nhỏ hơn một giá trị cho trước gọi là BER_T [16-20].

Hệ thống điều chế thích ứng về cơ bản có mô hình tương tự như hệ thống không truyền thích ứng, ngoại trừ đường truyền hồi tiếp giữa máy thu và máy phát, nghĩa là máy thu sử dụng tỷ số tín hiệu trên nhiễu thu được và lựa chọn mức điều chế hợp và hồi tiếp thông tin về mức điều chế thích hợp về máy phát. Để phân tích đơn giản, đường truyền hồi tiếp giữa máy thu và máy phát giả sử không trễ và không lỗi. Tuy nhiên, ngay cả khi xem xét đường truyền hồi tiếp có lỗi và có trễ, hiệu năng của hệ thống bị suy giảm và có thể bù bằng tăng công suất phát.

Giả sử hệ thống có $K+1$ chế độ truyền bao gồm: không truyền, BPSK, ..., M_k PSK, ..., M_K PSK với $M_k = 2^k$ và $k = 1, \dots, K$. Tương ứng với $K+1$ chế độ truyền chúng ta có các ngưỡng SNR phân định các chế độ truyền như sau:

Bảng 6.2: Mức ngưỡng của các chế độ truyền thích ứng.

k	0	1	...	k	...	K
Ngưỡng SNR	$\gamma_T^0 = 0$	$\gamma_T^1 = 0$		γ_T^k		$\gamma_T^{K+1} = \infty$
Chế độ truyền	Không truyền	BPSK		2^k PSK		2^K PSK

Tín hiệu thu tại máy thu có dạng như sau:

$$r = \sqrt{P}hx + n \quad (6.46)$$

với P là công suất của máy phát, h là hệ số đường truyền với $E\{|h|^2\} = \lambda$, x là tín hiệu sau điều chế với $E\{|x|^2\} = 1$. Tỷ số tín hiệu trên nhiễu tại máy thu là:

$$\gamma = \frac{P|h|^2}{N_0}. \quad (6.47)$$

Sau khi ước lượng tỷ số tín hiệu trên nhiễu nhận được, máy thu so sánh γ với các giá trị ngưỡng γ_T^k và chọn chế độ truyền phù hợp để tối đa hiệu suất phô tần. Máy thu sẽ hồi tiếp chế độ truyền về cho máy phát và máy phát sẽ phát chế độ truyền được đã được lựa chọn.

Giả sử hệ thống có $K+1$ chế độ truyền, thì số lượng bit cần hồi tiếp là $\lceil K+1 \rceil$ với $\lceil \cdot \rceil$ là phép toán lấy phần nguyên bên phải gần nhất.

Trong kỹ thuật truyền thích ứng, việc xác định giá trị γ_T^k là rất quan trọng. Dựa trên mục tiêu tối đa hiệu suất sử dụng phổ tần nhưng phải đảm bảo tỷ lệ lỗi bit trung bình của hệ thống phải nhỏ hơn hoặc bằng giá trị BER_T , cách đơn giản là cho tỷ lệ lỗi bit của tất cả các chế độ truyền nhỏ hơn BER_T , cụ thể:

$$\alpha_k Q\left(\sqrt{\beta_k \gamma_T^k}\right) = \text{BER}_T \quad (6.48)$$

với α_k và β_k là hệ số điều chế của M_k PSK với $M_k = 2^k$, xác định như sau:

$$\alpha_k = \frac{2}{\log_2 M_k} \quad (6.49)$$

và

$$\beta_k = 2 \sin^2\left(\frac{\pi}{M}\right) \log_2 M_k. \quad (6.50)$$

Do đó, ta có thể tính γ_T^k như sau:

$$\gamma_T^k = \frac{1}{\beta_k} \left[Q^{-1}\left(\frac{\text{BER}_T}{\alpha_k}\right) \right]^2. \quad (6.51)$$

Tiếp theo, chúng ta sẽ đánh giá hiệu năng của hệ thống truyền thích ứng bao gồm: xác suất của các chế độ truyền, xác suất dừng hệ thống, tỷ lệ lỗi bit trung bình và hiệu suất phổ tần.

Xác suất chế độ truyền: Chúng ta có $K+1$ chế độ truyền, xác suất chế độ truyền thứ k , π_k với $k = 0, \dots, K$ là:

$$\pi_k = \Pr\left(\gamma_T^k \leq \gamma \leq \gamma_T^{k+1}\right). \quad (6.52)$$

Ở kênh truyền fading Rayleigh, ta có thể tính xác suất chế độ truyền thứ k như sau:

$$\pi_k = \int_{\gamma_T^k}^{\gamma_T^{k+1}} f_\gamma(\gamma) d\gamma = \exp\left(-\frac{\gamma_T^k}{\bar{\gamma}}\right) - \exp\left(-\frac{\gamma_T^{k+1}}{\bar{\gamma}}\right). \quad (6.53)$$

Ta dễ dàng thấy rằng:

$$\sum_{k=0}^K \pi_k = 1. \quad (6.54)$$

Xác suất dùng của hệ thống truyền thích ứng là tương ứng với xác suất truyền ché độ đầu tiên, π_0 , cụ thể là:

$$OP = \pi_0 = 1 - \exp\left(-\frac{\gamma_T^1}{\bar{\gamma}}\right). \quad (6.55)$$

Hiệu suất phổ tần (Spectral Efficiency): Khi hệ thống dùng nhiều mức điều chế, ta có hiệu suất phổ tần hệ thống như sau:

$$ASE = \sum_{k=0}^K \pi_k \log_2 M_k. \quad (6.56)$$

Từ công thức (6.56), ta dễ dàng nhận thấy rằng:

$$ASE \leq \log_2 M_K \quad (6.57)$$

khi mà $\pi_k \leq 1 \forall k$.

Tỷ lệ lỗi bit trung bình: Chúng ta có $K+1$ ché độ truyền, mỗi ché độ truyền sử dụng mức điều chế khác nhau và có tỷ lệ lỗi bit khác nhau, do đó tỷ lệ lỗi bit trung bình của hệ thống là trung bình theo trọng số của các tỷ lệ lỗi bit tương ứng với các ché độ truyền:

$$BER = \frac{\sum_{k=0}^K \pi_k BER_k}{\sum_{k=0}^K \pi_k \log_2 M_k} \quad (6.58)$$

với

$$\begin{aligned} BER_k &= \int_{\gamma_T^k}^{\gamma_T^{k+1}} \alpha_k Q\left(\sqrt{\beta_k \gamma}\right) f_\gamma(\gamma) d\gamma \\ &= \frac{\alpha_k}{2} \int_{\gamma_T^k}^{\gamma_T^{k+1}} \operatorname{erfc}\left(\sqrt{\frac{\beta_k \gamma}{2}}\right) \frac{1}{\bar{\gamma}} e^{-\frac{\gamma}{\bar{\gamma}}} d\gamma \\ &= \frac{\alpha_k}{2} \left[\sqrt{\frac{\bar{\gamma} \beta_k}{\bar{\gamma} \beta_k + 2}} \operatorname{erf}\left(\sqrt{\frac{\gamma_T^k (\bar{\gamma} \beta_k + 2)}{2 \bar{\gamma}}}\right) + e^{-\frac{\gamma_T^k}{\bar{\gamma}}} \operatorname{erfc}\left[\sqrt{\frac{\beta_k \gamma_T^k}{2}}\right] \right] \\ &\quad - \frac{\alpha_k}{2} \left[\sqrt{\frac{\bar{\gamma} \beta_k}{\bar{\gamma} \beta_k + 2}} \operatorname{erf}\left(\sqrt{\frac{\gamma_T^{k+1} (\bar{\gamma} \beta_k + 2)}{2 \bar{\gamma}}}\right) + e^{-\frac{\gamma_T^{k+1}}{\bar{\gamma}}} \operatorname{erfc}\left[\sqrt{\frac{\beta_k \gamma_T^{k+1}}{2}}\right] \right]. \end{aligned} \quad (6.59)$$

Ví dụ 6.11

Mô phỏng hệ thống truyền thích ứng với $K+1$ ché độ truyền sử dụng điều chế MPSK ở kênh truyền fading Rayleigh. So sánh với kết quả phân tích lý thuyết và nhận xét.

Giải:

```
clear all
% Tỷ số tín hiệu trên nhiễu theo symbol
EsNo = 0:2:30;
SNR = 10.^{(EsNo/10)};
% Độ lợi kênh truyền trung bình
Omega = 2;
% Số chế độ truyền
K = 4;
% Mức ngưỡng tỷ lệ lỗi bit cho phép của hệ thống
BERT=10^-2;

% Lý thuyết
m = 1:K-1;
M = 2.^{(1:K-1)};

alpha = 2./log2(M);
beta = 2.*sin(pi./M).^2;
gT = 1./beta.*{(qfuncinv(BERT./alpha)).^2};
gT = [0 gT Inf];
g = Omega.*SNR;

% Xác suất chế độ truyền
MP_a = zeros(K,length(EsNo));
for k = 1:K
    MP_a(k,:) = exp(-gT(k)./g) - exp(-gT(k+1)./g);
end

% Xác suất dùng hệ thống
OP_a = MP_a(1,:);

% Hiệu suất phô tần
ASE_a = zeros(size(EsNo));
for k = 2:K
    ASE_a = ASE_a + log2(M(k-1)).*MP_a(k,:);
end

% Tỷ lệ lỗi bit
BERk_a = zeros(K-1,length(EsNo));
for k = 1:K-1
    BERk_a(k,:) = alpha(k)./2.*{(sqrt(beta(k).*g)./(beta(k).*g + 2)).*erf(sqrt(gT(k+1).*{(beta(k).*g + 2)./(2.*g)})) ...
        + exp(-gT(k+1)./g).*erfc(sqrt(beta(k).*gT(k+1)/2)))...
        - alpha(k)./2.*{(sqrt(beta(k).*g)./(beta(k).*g + 2)).*erf(sqrt(gT(k+2).*{(beta(k).*g + 2)./(2.*g)})) ...
        + exp(-gT(k+2)./g).*erfc(sqrt(beta(k).*gT(k+2)/2)))};
end
ABER_a = sum(log2(M)'.*BERk_a,1)./ASE_a;

% Mô phỏng
% Số lượng symbol sẽ truyền
NoS = 10^5;
% Khởi tạo các biến
ABER_s = zeros(size(EsNo));
ASE_s = zeros(size(EsNo));
```

```

MP_s = zeros(K,length(EsNo));

for idx = 1:length(EsNo)

    Ps = SNR(idx);
    h = sqrt(Ps.*Omega/2).*(randn(1,NoS) + 1i*randn(1,NoS));
    g = abs(h).^2;

    % Lựa chọn chế độ truyền dựa trên tỷ số tín hiệu trên nhiễu và ngưỡng SNR
    MT = zeros(1,NoS);
    for k = 1:length(gT)-1
        MT = MT + k*((g >= gT(k))&(g < gT(k+1)));
    end

    % Xác suất của các chế độ truyền
    for k=1:K
        MP_s(k,idx) = sum(MT==k)/NoS;
    end

    % Điều chỉnh
    s = zeros(size(MT));
    for k = 2:K
        s = s + (MT==k).*randi([0 M(k-1)-1],1,NoS);
    end

    % Truyền tín hiệu
    Tx = zeros(size(MT));
    for k = 2:K
        Tx = Tx + (MT==k).*pskmod(s.*(MT==k),M(k-1),0,'gray');
    end

    % Nhận tín hiệu
    n = sqrt(1/2).*randn(1,NoS) + 1i*randn(1,NoS);
    Rx = h.*Tx + n;

    s_ = zeros(size(s));
    NoB = zeros(size(s));

    for k = 2:K
        s_ = s_ + (MT==k).*pskdemod((MT==k).*Rx./h,M(k-1),0,'gray');
        NoB = NoB + (MT==k).*log2(M(k-1));
    end

    % Tính tỷ lệ lỗi bit
    ASE_s(idx) = sum(log2(M)'.*MP_s(2:end,idx),1);
    ABER_s(idx) = biterr(s,s_)/sum(NoB);
end

OP_s = MP_s(1,:);

% Vẽ hình so sánh
figure(1);
set(gcf,'color','White');
plot(EsNo,MP_a,'-',EsNo,MP_s,'o');

```

```

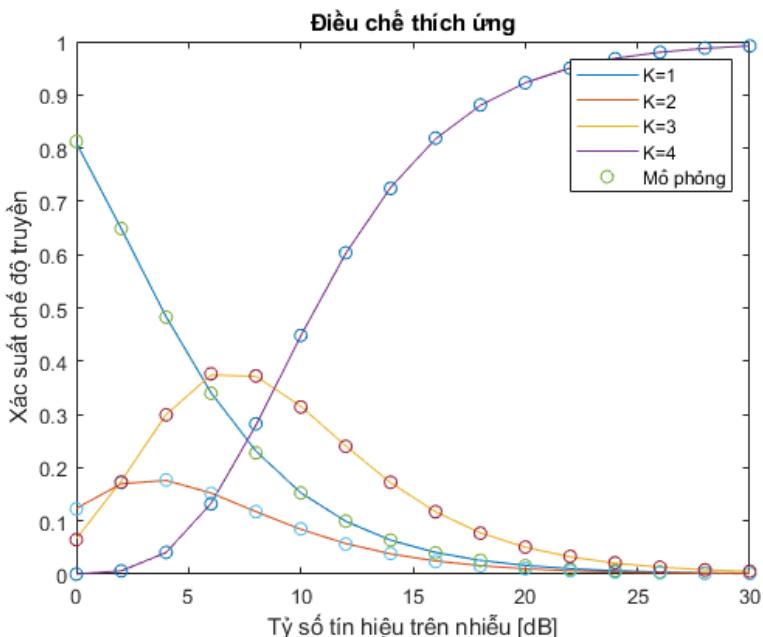
legend('K=1','K=2','K=3','K=4','Mô phỏng');
xlabel('Tỷ số tín hiệu trên nhiễu [dB]');
ylabel('Xác suất chế độ truyền');
title('Điều chế thích ứng')

% Vẽ hình so sánh
figure(2);
set(gcf,'color','White');
semilogy(EsNo,OP_a,'-',EsNo,OP_s,'o');
legend('Lý thuyết','Mô phỏng');
xlabel('Tỷ số tín hiệu trên nhiễu [dB]');
ylabel('Xác suất dùng');
title('Điều chế thích ứng')

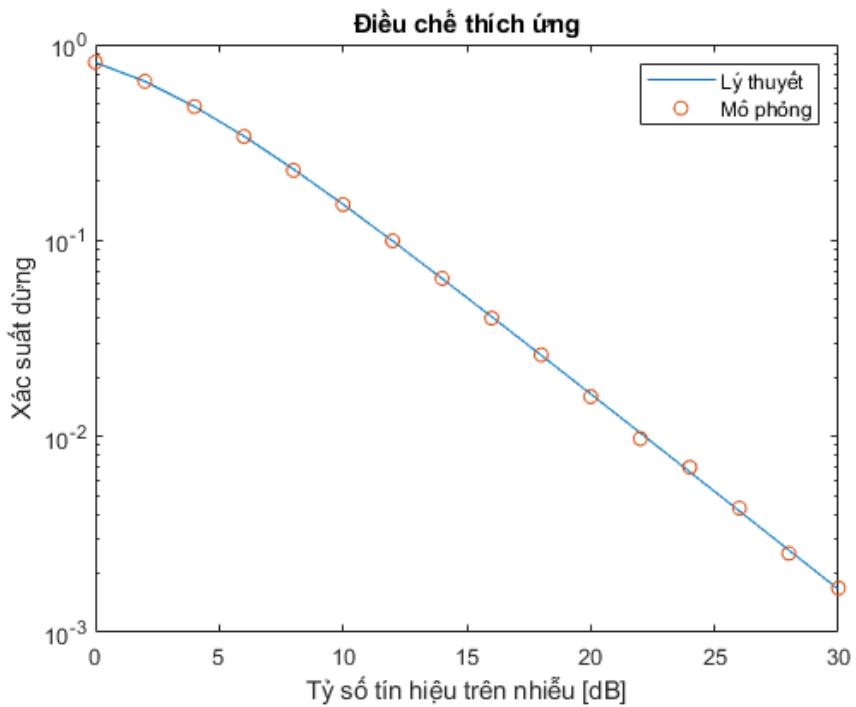
% Vẽ hình so sánh
figure(3);
set(gcf,'color','White');
plot(EsNo,ASE_a,'-',EsNo,ASE_s,'o');
legend('Lý thuyết','Mô phỏng');
xlabel('Tỷ số tín hiệu trên nhiễu [dB]');
ylabel('Hiệu suất phổ tán');
title('Điều chế thích ứng')

% Vẽ hình so sánh
figure(4);
set(gcf,'color','White');
semilogy(EsNo,ABER_a,'-',EsNo,ABER_s,'o');
legend('Lý thuyết','Mô phỏng');
xlabel('Tỷ số tín hiệu trên nhiễu [dB]');
ylabel('Tỷ lệ lỗi bit trung bình');
title('Điều chế thích ứng')

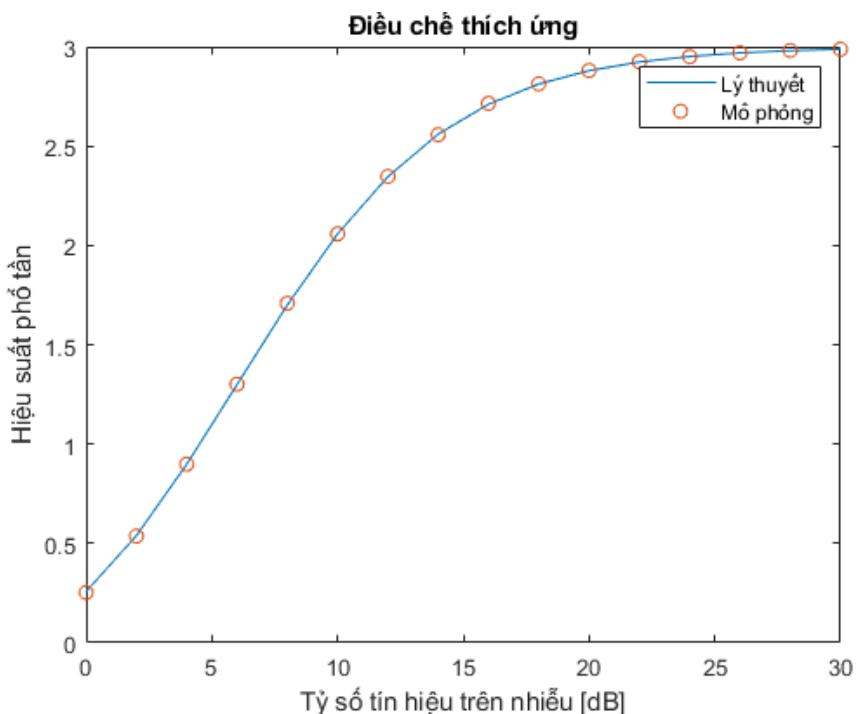
```



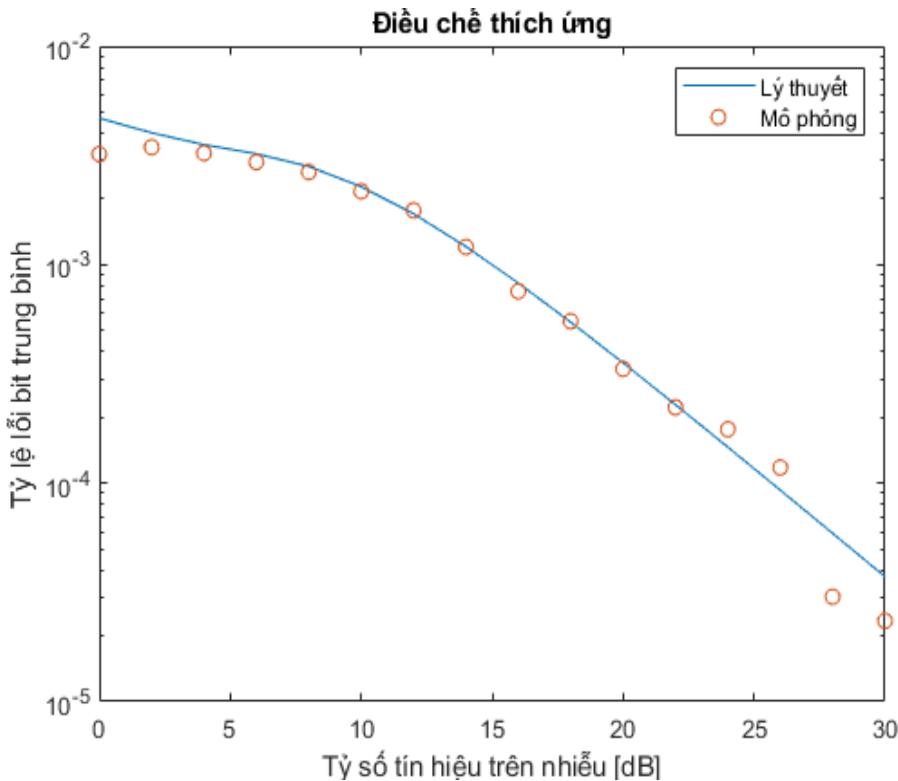
Hình 6.13: Xác suất của các chế độ truyền của hệ thống truyền thích ứng ở kênh truyền fading Rayleigh.



Hình 6.14: Xác suất đúng của hệ thống truyền thích ứng ở kênh truyền fading Rayleigh.



Hình 6.15: Hiệu suất phô tần của hệ thống truyền thích ứng sử dụng MPSK ở kênh truyền fading Rayleigh.



Hình 6.16: Tỷ lệ lỗi bit trung bình của hệ thống truyền thích ứng dùng MPSK ở kênh truyền fading Rayleigh.

6.3.4 Mô phỏng hệ thống truyền giải mã và chuyển tiếp hai chặng

Kỹ thuật chuyển tiếp là một trong những kỹ thuật hiệu quả để mở rộng vùng phủ sóng, cải thiện hiệu năng mạng và hạn chế can nhiễu. Kỹ thuật chuyển tiếp ngày nay được áp dụng trong các mạng vô tuyến thế hệ mới.

Mô hình hệ thống chuyển tiếp bao gồm 3 nút mạng: nút nguồn, nút chuyển tiếp và nút đích với hai chặng. Quá trình truyền thông sẽ lần lượt từ nút nguồn đến nút chuyển tiếp trong khe thời gian thứ 1 và nút chuyển tiếp sẽ chuyển tiếp dữ liệu nhận được đến nút đích dùng kỹ thuật giải mã và chuyển tiếp trong khe thời gian thứ 2.

Gọi h_1 và h_2 lần lượt là hệ số kênh truyền của chặng 1 và 2, ta có tín hiệu nhận được tại chặng 1 và 2 là:

$$r_i = P_i h_i x_i + n_i \quad (6.60)$$

với P_i là công suất phát ở chặng i , x_i là tín hiệu điều chế ở chặng i với $E\{|x_i|^2\} = 1$ và n_i là nhiễu trắng ở máy thu có trung bình bằng không và phương sai N_0 .

Tỷ số tín hiệu trên nhiễu của chặng i là:

$$\gamma_i = \frac{P_i |h_i|^2}{N_0}. \quad (6.61)$$

Ở kênh truyền fading Rayleigh, hàm PDF và CDF của γ_i là:

$$f_{\gamma_i}(\gamma) = \frac{1}{\bar{\gamma}_i} e^{-\frac{\gamma}{\bar{\gamma}_i}}, \quad (6.62)$$

$$F_{\gamma_i}(\gamma) = 1 - e^{-\frac{\gamma}{\bar{\gamma}_i}}, \quad (6.63)$$

6.3.4.1 Phân tích chính xác

Với chuyển tiếp cố định, hệ thống chuyển tiếp hai chặng sẽ được xem là dừng nếu chặng 1 sẽ bị dừng hoặc chặng 1 không bị dừng và chặng 2 bị dừng. Do đó, ta có thể viết xác suất dừng của hệ thống như sau:

$$\begin{aligned} OP &= \Pr(\gamma_1 < \gamma_{th}) + \Pr(\gamma_1 \geq \gamma_{th}) \Pr(\gamma_2 < \gamma_{th} \mid \gamma_1 \geq \gamma_{th}) \\ &= \Pr(\gamma_1 < \gamma_{th}) + \Pr(\gamma_2 < \gamma_{th}, \gamma_1 \geq \gamma_{th}) \end{aligned} \quad (6.64)$$

với γ_{th} là ngưỡng dừng của hệ thống.

Giả sử là kênh truyền của hai chặng là độc lập với nhau, ta có thể viết lại (6.64) như sau:

$$OP = \Pr(\gamma_1 < \gamma_{th}) + \Pr(\gamma_2 < \gamma_{th}) \Pr(\gamma_1 \geq \gamma_{th}) \quad (6.65)$$

Từ (6.65), ta có dạng đóng của OP như sau:

$$OP = 1 - e^{-\frac{\gamma_{th}}{\bar{\gamma}_1}} + \left(1 - e^{-\frac{\gamma_{th}}{\bar{\gamma}_2}}\right) e^{-\frac{\gamma_{th}}{\bar{\gamma}_1}}. \quad (6.66)$$

Khác với xác suất dừng hệ thống, ta cũng có thể tính tỷ lệ lỗi bit trung bình của hai chặng như sau:

$$BER = BER_1(1 - BER_2) + (1 - BER_1) BER_2. \quad (6.67)$$

Công suất tính xác suất lỗi bit trung bình khác với công thức tính xác suất dừng vì chúng ta phải loại bỏ trường hợp chặng 1 và chặng 2 cùng giải mã sai nhưng dẫn đến giải mã đúng ở phía đầu cuối. Giả sử hệ thống sử dụng điều chế kết hợp, ta có BER_i , được tính như sau:

$$BER_i = \int_0^{\infty} \alpha_m Q\left(\sqrt{\beta_m \gamma}\right) f_{\gamma_i}(\gamma) d\gamma = \frac{\alpha_m}{2} \left(1 - \sqrt{\frac{\beta_m \bar{\gamma}_i}{\beta_m \bar{\gamma}_i + 2}}\right) \quad (6.68)$$

với α_m và β_m là các hệ số phụ thuộc vào loại và mức điều chế, có thể tham khảo ở Bảng 6.1 của [4]. Ví dụ với điều chế BPSK, ta có $\alpha_m = 1, \beta_m = 2$.

6.3.4.2 Phân tích xấp xỉ

Một kỹ thuật xấp xỉ cho hệ thống chuyển tiếp cố định dựa trên quan sát rằng tỷ số tín hiệu trên nhiễu tương đương của hệ thống chuyển tiếp sẽ bằng tỷ số tín hiệu trên nhiễu của chặng yếu nhất, cụ thể ta có thể viết như sau:

$$\gamma_{\Sigma} = \min(\gamma_1, \gamma_2). \quad (6.69)$$

Giả sử kênh truyền của hai chặng độc lập thống kê, ta có thể viết hàm CDF của γ_{Σ} như sau:

$$\begin{aligned} F_{\gamma_{\Sigma}}(\gamma) &= \Pr[\min(\gamma_1, \gamma_2) \leq \gamma] \\ &= 1 - \Pr[\min(\gamma_1, \gamma_2) > \gamma] \\ &= 1 - \Pr[\gamma_1 > \gamma, \gamma_2 > \gamma] \\ &= 1 - \Pr(\gamma_1 > \gamma) \Pr(\gamma_2 > \gamma) \\ &= 1 - \exp\left[-\gamma\left(\frac{1}{\bar{\gamma}_1} + \frac{1}{\bar{\gamma}_2}\right)\right]. \end{aligned} \quad (6.70)$$

Hàm PDF của γ_{Σ} có được bằng cách đạo hàm hàm CDF từ (6.70) như sau:

$$f_{\gamma_{\Sigma}}(\gamma) = \left(\frac{1}{\bar{\gamma}_1} + \frac{1}{\bar{\gamma}_2}\right) \exp\left[-\gamma\left(\frac{1}{\bar{\gamma}_1} + \frac{1}{\bar{\gamma}_2}\right)\right]. \quad (6.71)$$

Xác suất dừng của hệ thống có thể tính dựa vào hàm CDF của γ_{Σ} như sau:

$$\text{OP} = F_{\gamma_{\Sigma}}(\gamma_{th}) = 1 - \exp\left[-\gamma_{th}\left(\frac{1}{\bar{\gamma}_1} + \frac{1}{\bar{\gamma}_2}\right)\right]. \quad (6.72)$$

Giả sử hệ thống dùng điều chế kết hợp, ta có tỷ lệ lỗi bit trung bình của hệ thống như sau:

$$\text{BER} = \int_0^{\infty} \alpha_m Q\left(\sqrt{\beta_m \gamma}\right) f_{\gamma_{\Sigma}}(\gamma) d\gamma = \frac{\alpha_m}{2} \left(1 - \sqrt{\frac{\beta_m \bar{\gamma}_{\Sigma}}{\beta_m \bar{\gamma}_{\Sigma} + 2}}\right) \quad (6.73)$$

$$\text{với } \gamma_{\Sigma} = \left(\frac{1}{\bar{\gamma}_1} + \frac{1}{\bar{\gamma}_2}\right)^{-1} = \frac{\bar{\gamma}_1 \bar{\gamma}_2}{\bar{\gamma}_1 + \bar{\gamma}_2}.$$

Ví dụ 6.12

Mô phỏng hệ thống chuyển tiếp hai chặng sử dụng kỹ thuật giải mã và chuyển tiếp cố định qua hai tham số hiệu năng là xác suất dừng hệ thống và tỷ lệ lỗi bit cho điều chế BPSK ở kênh truyền fading Rayleigh.

- a. So sánh với kết quả lý thuyết chính xác.
- b. So sánh với kết quả lý thuyết xấp xỉ.

Giải: Mã nguồn Matlab của Ví dụ 6.12 là như sau:

```
clear all
% Tỷ số tín hiệu trên nhiễu
EsNo = 0:2:30;
SNR = 10.^{EsNo/10};
Omega = [1 10];
g1_ = Omega(1).*SNR;
g2_ = Omega(2).*SNR;
gth = 1;

OPe_a = 1 - exp(-gth./g1_) + exp(-gth./g1_).*(1 - exp(-gth./g2_));
BER1 = 1/2.*{(1 - sqrt(g1_./(1 + g1_)))};
BER2 = 1/2.*{(1 - sqrt(g2_./(1 + g2_)))};
BERe_a = BER1.*{(1 - BER2) + (1 - BER1).*BER2};

g12_ = g1_.*g2_./(g1_ + g2_);
OPa_a = 1 - exp(-gth./g12_);
BERa_a = 1/2.*{(1 - sqrt(g12_./(1 + g12_)))};

% Mô phỏng
N = 10^5;
M = 2;

x1 = randi([0 1],length(EsNo),N);
Tx1 = pskmod(x1,M,0,'gray');
h1 = sqrt(SNR(:).*Omega(1)/2).*{(randn(1,N) + 1i.*randn(1,N))};
n1 = sqrt(1/2).*{(randn(length(EsNo),N) + 1i.*randn(length(EsNo),N))};

Rx1 = h1.*Tx1 + n1;
x1_ = pskdemod(Rx1./h1,M,0,'gray');

Tx2 = pskmod(x1_,M,0,'gray');
h2 = sqrt(SNR(:).*Omega(2)/2).*{(randn(1,N) + 1i.*randn(1,N))};
n2 = sqrt(1/2).*{(randn(length(EsNo),N) + 1i.*randn(length(EsNo),N))};
Rx2 = h2.*Tx2 + n2;
x2_ = pskdemod(Rx2./h2,M,0,'gray');

BER_s = sum(x1~=x2_,2)/(log2(M).*N);

g1 = abs(h1).^2;
g2 = abs(h2).^2;
OP_s = sum((g1 < gth)|(g1 >= g1)&(g2 < gth),2)/N;

% Vẽ xác suất dùng hệ thống
figure(1);
set(gcf,'color','White');
semilogy(EsNo,OPe_a,'-',EsNo,OPa_a,'--',EsNo,OP_s,'o');
legend('Chính xác','Xáp xi','Mô phỏng');
xlabel('Tỷ lệ tín hiệu trên nhiễu [dB]');
ylabel('Xác suất dùng');
title('Chuyển tiếp hai chặng');

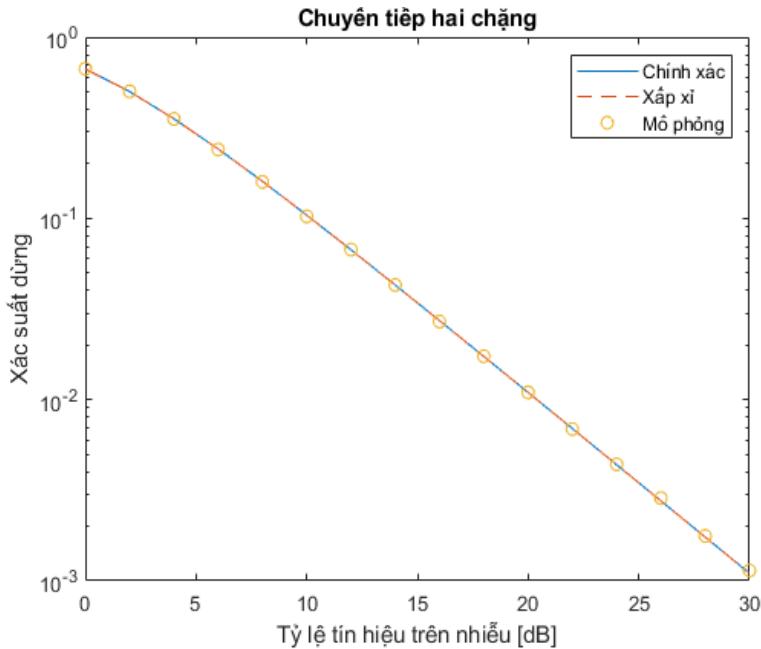
% Vẽ tỷ lệ lỗi bit hệ thống
figure(2);
set(gcf,'color','White');
semilogy(EsNo,BERe_a,'-',EsNo,BERa_a,'.',EsNo,BER_s,'o');
```

```

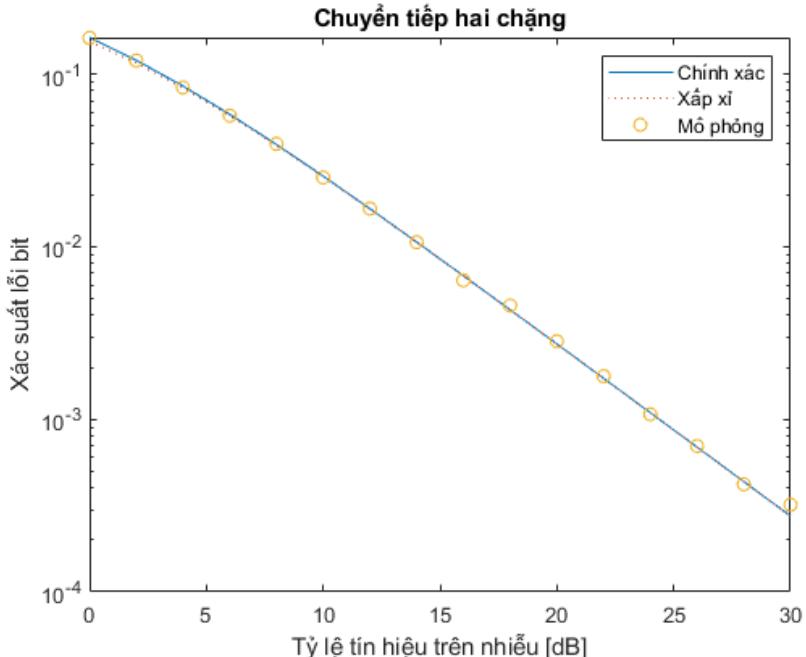
legend('Chính xác','Xấp xỉ','Mô phỏng');
xlabel('Tỷ lệ tín hiệu trên nhiễu [dB]');
ylabel('Xác suất lỗi bit');
title('Chuyển tiếp hai chặng');

```

Kết quả mô phỏng của mã nguồn trên là Hình 6.17 và Hình 6.18 như sau:



Hình 6.17: Xác suất dừng của hệ thống chuyển tiếp hai chặng có định
ở kênh truyền fading Rayleigh.



Hình 6.18: Tỷ lệ lỗi bit của hệ thống chuyển tiếp hai chặng có định
ở kênh truyền fading Rayleigh.

6.4 Bài tập

1. Xem xét hệ thống phân tập thu kết hợp lựa chọn với hai nhánh hoạt động ở kênh truyền fading Rayleigh, giả sử rằng $\bar{\gamma}_1 \neq \bar{\gamma}_2$ và các kênh truyền là độc lập. Hãy:
 - a. Phân tích xác suất dừng của hệ thống.
 - b. Phân tích tỷ lệ lỗi bit trung bình của hệ thống với điều chế MPSK.
 - c. Phân tích dung lượng Shannon của hệ thống.
 - d. Viết chương trình mô phỏng trên Matlab kiểm chứng kết quả ở các câu trên.
2. Xem xét hệ thống phân tập thu kết hợp lựa chọn với hai nhánh hoạt động ở kênh truyền fading Nakagami- m , giả sử rằng $\bar{\gamma}_1 = \bar{\gamma}_2$ và các kênh truyền là độc lập. Hãy:
 - a. Phân tích xác suất dừng của hệ thống.
 - b. Phân tích tỷ lệ lỗi bit trung bình của hệ thống với điều chế MPSK.
 - c. Phân tích dung lượng Shannon của hệ thống.
 - d. Viết chương trình mô phỏng trên Matlab kiểm chứng kết quả ở các câu trên.
3. Xem xét hệ thống phân tập thu kết hợp lựa chọn có N nhánh hoạt động ở kênh truyền Rayleigh fading, giả sử rằng $\bar{\gamma}_1 \neq \dots \neq \bar{\gamma}_N$ và các kênh truyền là độc lập.
 - a. Phân tích xác suất dừng của hệ thống.
 - b. Phân tích tỷ lệ lỗi bit trung bình của hệ thống với điều chế MPSK.
 - c. Phân tích dung lượng Shannon của hệ thống.
 - d. Viết chương trình mô phỏng trên Matlab kiểm chứng kết quả ở các câu trên.
4. Xem xét hệ thống MRC với hai nhánh hoạt động ở kênh truyền fading Nakagami- m , giả sử rằng $\bar{\gamma}_1 = \bar{\gamma}_2$. Hãy:
 - a. Phân tích xác suất dừng của hệ thống.
 - b. Phân tích tỷ lệ lỗi bit trung bình của hệ thống với điều chế MPSK.
 - c. Phân tích dung lượng Shannon của hệ thống.
 - d. Viết chương trình mô phỏng trên Matlab kiểm chứng kết quả ở các câu trên.
5. Xem xét hệ thống MRC với hai nhánh hoạt động ở kênh truyền fading Rayleigh, giả sử rằng $\bar{\gamma}_1 \neq \bar{\gamma}_2$. Hãy:
 - a. Phân tích xác suất dừng của hệ thống.
 - b. Phân tích tỷ lệ lỗi bit trung bình của hệ thống với điều chế MPSK.
 - c. Phân tích dung lượng Shannon của hệ thống.
 - d. Viết chương trình mô phỏng trên Matlab kiểm chứng kết quả ở các câu trên.

6. Xem xét hệ thống MRC có N nhánh với N cho trước hoạt động ở kênh truyền fading Rayleigh, giả sử rằng $\bar{\gamma}_1 \neq \dots \neq \bar{\gamma}_N$. Hãy:
- Phân tích xác suất dừng của hệ thống.
 - Phân tích tỷ lệ lỗi bit trung bình của hệ thống với điều chế MPSK.
 - Phân tích dung lượng Shannon của hệ thống.
 - Viết chương trình mô phỏng trên Matlab kiểm chứng kết quả ở các câu trên.
7. Xem xét hệ thống lựa chọn anten phát (Transmit Antenna Selection) với số lượng anten phía máy phát là N và phía máy thu là 1 hoạt động ở kênh truyền fading Rayleigh. Giả sử là kênh truyền hồi tiếp là không trễ và không lỗi và các kênh truyền là độc lập và đồng nhất. Hãy:
- Phân tích xác suất dừng của hệ thống.
 - Phân tích tỷ lệ lỗi bit trung bình của hệ thống với điều chế MPSK.
 - Phân tích dung lượng Shannon của hệ thống.
 - Viết chương trình mô phỏng trên Matlab kiểm chứng kết quả ở các câu trên.
8. Xem xét hệ thống phân tập phát sử dụng mã không gian thời gian Alamouti với số lượng anten phía máy phát là 2 và phía máy thu là 1, hoạt động ở kênh truyền fading Nakagami-m. Giả sử các kênh truyền là độc lập và đồng nhất. Hãy:
- Phân tích xác suất dừng của hệ thống.
 - Phân tích tỷ lệ lỗi bit trung bình của hệ thống với điều chế MPSK.
 - Phân tích dung lượng Shannon của hệ thống.
 - Viết chương trình mô phỏng trên Matlab kiểm chứng kết quả ở các câu trên.
9. Xem xét hệ thống phân tập phát sử dụng mã không gian thời gian Alamouti với số lượng anten phía máy phát là 2 và phía máy thu là 2, hoạt động ở kênh truyền fading Rayleigh. Giả sử các kênh truyền là độc lập và đồng nhất. Hãy:
- Phân tích xác suất dừng của hệ thống.
 - Phân tích tỷ lệ lỗi bit trung bình của hệ thống với điều chế MPSK.
 - Phân tích dung lượng Shannon của hệ thống.
 - Viết chương trình mô phỏng trên Matlab kiểm chứng kết quả ở các câu trên.
10. Xem xét hệ thống điều chế thích ứng $K+1$ chế độ sử dụng điều chế MQAM ở kênh truyền fading Rayleigh. Hãy:
- Phân tích lý thuyết các tham số hiệu năng của hệ thống.
 - Viết chương trình mô phỏng trên Matlab kiểm chứng kết quả ở câu trên.
11. Xem xét hệ thống điều chế thích ứng $K+1$ chế độ truyền sử dụng điều chế MQAM ở kênh truyền fading Nakagami-m. Hãy:

- a. Phân tích lý thuyết các tham số hiệu năng của hệ thống.
 - b. Chứng minh rằng ở vùng tỷ số tín hiệu trên nhiễu cao, hiệu suất phô tàn của hệ thống sẽ hội tụ đến một giá trị nào đó và tìm giá trị đó.
 - c. Viết chương trình mô phỏng trên Matlab kiểm chứng kết quả ở các câu trên.
12. Xem xét hệ thống điều chế thích ứng có $K+1$ chế độ truyền hoạt động ở kênh truyền fading Rayleigh. Hãy
- a. Tìm giá trị ngưỡng tối ưu để tối đa hiệu suất phô tàn của hệ thống.
 - b. Viết chương trình mô phỏng trên Matlab kiểm chứng kết quả ở các câu trên.
13. Xem xét hệ thống truyền 2 chặng với chế độ khuếch đại và chuyển tiếp, biết rằng tỷ số tín hiệu trên nhiễu tương đương có công thức $\gamma_{\Sigma} = \gamma_1 \gamma_2 / (\gamma_1 + \gamma_2 + 1)$. Hãy:
- a. Phân tích xác suất dùng của hệ thống, biết rằng hệ thống hoạt động ở kênh truyền fading Rayleigh.
 - b. Viết chương trình mô phỏng trên Matlab kiểm chứng kết quả ở câu trên.
14. Xem xét hệ thống truyền 2 chặng và áp dụng mã không gian thời gian Alamouti trên mỗi chặng với 2 anten phát và 1 anten thu, hoạt động ở kênh truyền fading Rayleigh. Hãy
- a. Mô tả hoạt động của hệ thống biết rằng nút chuyển tiếp sử dụng kỹ thuật truyền chuyển tiếp cố định.
 - b. Phân tích hiệu năng của hệ thống dưới dạng xác suất dùng và tỷ lệ lỗi bit cho điều chế MQAM.
 - c. Mô phỏng hệ thống và kiểm chứng kết quả với câu b.
15. Hệ thống chuyển tiếp hai chặng sử dụng kỹ thuật khuếch đại và chuyển tiếp có tỷ số tín hiệu trên nhiễu tương đương có công thức $\gamma_{\Sigma} = \gamma_1 \gamma_2 / (\gamma_1 + \gamma_2 + 1)$.
- a. Phân tích xác suất dùng của hệ thống, biết rằng hệ thống hoạt động ở kênh truyền Nakagami-m
 - b. Viết chương trình mô phỏng trên Matlab kiểm chứng kết quả ở câu trên.
16. Xem xét hệ thống truyền đa chặng với số chặng tổng quát cho trước và sử dụng kỹ thuật truyền chuyển tiếp cố định.
- a. Phân tích xác suất dùng của hệ thống, biết rằng hệ thống hoạt động ở kênh truyền fading Rayleigh.
 - b. Phân tích tỷ lệ lỗi bit của hệ thống
 - c. Viết chương trình mô phỏng trên Matlab kiểm chứng kết quả ở các câu trên.

TÀI LIỆU THAM KHẢO

- [1] M. Hata, "Empirical formula for propagation loss in land mobile radio services," *IEEE Transactions on Vehicular Technology*, vol. 29, no. 3, pp. 317-325, 1980.
- [2] I. P802.16j, "IEEE Standard for Local and Metropolitan Area Networks, Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems: Multihop Relay Specification," in *802.16 Relay Task Group* vol. P802.16j/D3, ed. http://ieee802.org/16/relay/docs/80216j-06_026r4.zip, 2008.
- [3] S. M. Ross, *Simulation*, 4th ed. San Diego: Elsevier Academic Press, 2006, pp. xiii, 274 p.
- [4] A. Goldsmith, *Wireless communications*. Cambridge ; New York: Cambridge University Press, 2005, pp. xxviii, 644 p.
- [5] M. Abramowitz, I. A. Stegun, and Knovel (Firm). (1972). *Handbook of mathematical functions with formulas, graphs, and mathematical tables* (10th printing, with corrections. ed.). Available: <http://www.convertit.com/Go/GovCon/Reference/AMS55.ASP?Res=200 &Page=-1>
- [6] R. W. Hamming, *Coding and information theory*. Englewood Cliffs, N.J.: Prentice-Hall, 1980, pp. xii, 239 p.
- [7] M. K. Simon and M.-S. Alouini, *Digital communication over fading channels*, 2nd ed. (Wiley series in telecommunications and signal processing). Hoboken, N.J.: John Wiley & Sons, 2005, pp. xxxiv, 900 p.
- [8] M. S. Alouini and M. K. Simon, "Dual diversity over correlated log-normal fading channels," *IEEE Transactions on Communications*, vol. 50, no. 12, pp. 1946-1959, 2002.
- [9] M. Nakagami, "The m-distribution - A general formula of intensity distribution of rapid fading," 1967.
- [10] S. Haghani and N. C. Beaulieu, "Optimization of Predetection Switched Combining in Correlated Rician Fading," in *Vehicular Technology Conference, 2006. VTC-2006 Fall. 2006 IEEE 64th*, 2006, pp. 1-5.
- [11] M. K. Simon, *Probability distributions involving Gaussian random variables : a handbook for engineers and scientists*. Boston: Kluwer Academic Publishers, 2002, pp. xix, 200 p.
- [12] M. K. Simon and M.-S. Alouini, *Digital communication over fading channels : a unified approach to performance analysis* (Wiley series in telecommunications and signal processing). New York ; Chichester [England]: John Wiley & Sons, 2000, pp. xix, 544 p.
- [13] I. S. Gradshteyn, I. M. Ryzhik, A. Jeffrey, and D. Zwillinger, *Table of integrals, series and products*, 7th ed. Amsterdam ; Boston: Elsevier, 2007, pp. xlv, 1171 p.
- [14] J. G. Proakis, *Digital communications*, 4th ed. (McGraw-Hill series in electrical and computer engineering). Boston: McGraw-Hill, 2001, pp. xxi, 1002 p.
- [15] S. M. Alamouti, "A Simple Transmit Diversity Technique for Wireless Communications," *IEEE Journal on Sel. Areas in Comm.*, vol. 16, no. 8, pp. 1451-1458, 1998.
- [16] J. Hayes, "Adaptive Feedback Communications," *IEEE Transactions on Communication Technology*, vol. 16, no. 1, pp. 29-34, 1968.
- [17] A. J. Goldsmith and C. Soon-Ghee, "Variable-rate variable-power MQAM for fading channels," *IEEE Transactions on Communications*, vol. 45, no. 10, pp. 1218-1230, 1997.

- [18] M. S. Alouini, "Adaptive and Diversity Techniques for Wireless Digital Communications over Fading Channels," PhD., Electrical Engineering, California Institute of Technology, Pasadena, California, 1998.
- [19] M. S. Alouini and A. J. Goldsmith, "Capacity of Rayleigh fading channels under different adaptive transmission and diversity-combining techniques," *IEEE Transactions on Vehicular Technology* vol. 48, no. 4, pp. 1165-1181, 1999.
- [20] V. K. N. Lau and S. V. Maric, "Variable rate adaptive modulation for DS-CDMA," *IEEE Transactions on Communications*, vol. 47, no. 4, pp. 577-589, 1999.

CHỈ MỤC

abs 25, 60, 62, 63, 132, 133, 140, 149, 157, 196, 197, 213, 215, 216, 218, 220, 227, 228, 229, 230, 232, 235, 237, 240, 241, 245, 250, 256	comment 19, 76
Alamouti 242, 243, 244, 245, 259, 260	Communications Toolbox 104, 112, 113, 202
A-law 169	compand 169, 170
all 31, 32, 33, 34, 61, 71, 127, 128, 141, 192	contains 24, 68
AM 114, 185, 186, 187, 191	convenc 182
amdemod 185	corroef 130
ammod 185, 186	cov 25, 54, 130, 131
any 57, 61, 71	cyclgen 178, 179
App Designer 97, 98	cyclpoly 178, 179
apskdemod 191	char 60, 62, 63, 67
apskmod 191	Chebyshev 131, 132
arithdeco 175	chi số logic 38, 40
arithenco 175, 176	Chi số logic 38
axis80 , 83, 91, 92, 93, 134, 135, 136, 137, 139, 142, 143, 171, 173, 227	Chi số tuyển tính 44
bảng giải mã syndrome 178, 179	chi2rnd 160, 215, 216
bar 68, 69, 77, 96	chirp 162
bar3 77	Chi-Square 89, 90, 166, 167, 215
bar3h 78	chol 25, 54
bát đắng thức Markov 131	chú giải 76
bchdec 180	Chuỗi 60, 61
bchenc 180	Chuyển vị 30
bchgenpoly 180	dbclear 75
bchnumerr 180	dbquit 75
berawgn 193, 194, 195	dbstep 75
bien chay 21, 58, 74	dbstop 75
biểu đồ tần suất	deblank 60, 61, 62
18, 78, 84, 85, 86, 87, 117, 122, 124, 133, 134, 135, 151, 156, 157, 159, 160	debug 17, 74, 75
bin2dec 60, 63	dec2bin 60, 63
binocdf 138	dec2hex 60, 63
binofit 138	decode 178, 179
binoinv 138	det25 , 54
binopdf 138	diag 25, 27, 36
binornd 138, 140	diric 162
binostat 138	distspec 182
blkdiag 25, 27	DPCM 173, 174
Bộ dao động điều khiển theo điện áp 162	dpcmdeco 171, 174
Bộ giải mã khối 178	dpcmenco 171, 174
Bộ mã hóa khối 178	dpcmopt 171
breakpoint 75	dpskdemod 191
can nhiễu 205, 253	dpskmod 191
Cáu trúc 72	dung lượng 14, 81, 175, 200, 212, 221, 223, 224, 230, 231, 232, 258, 259
cáu trúc phô Doppler 202	Dung lượng dừng 224
cáu trúc trellis 182	Dung lượng Shannon 224
cdf 88, 89, 138, 139, 142, 159	dvbapskdemod 191
cell 61, 65, 66, 67, 68, 69, 70, 71, 72, 73, 105	dvbapskmod 191
cell2mat 61, 66	dvbs2ldpc 182
cellstr 61, 67	đa thức sinh 178, 181
clear all 21, 22, 135, 136, 138, 160, 208, 210, 213, 215, 218, 220, 236, 249, 256	điều chế BPSK113, 114, 192, 201, 208, 209, 222, 224, 225, 236, 238, 239, 240, 242, 244, 245, 254, 255
colormap 92, 93	điều chế tương tự 113, 114, 183, 184, 185, 187, 189
Command Window 19, 20	Định luật số lớn 132
	Định lý giới hạn trung tâm 133
	độ lợi kênh truyền 212, 222, 223, 226, 229, 232

đò thi	76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 87, 91, 92, 93, 133, 137, 153, 160, 173, 194, 235	
đường bao phức	190	
đường nhín thẳng	217	
eigs	54	
encode	178, 179	
errorbar	83	
expcdf	91, 92, 152	
expfit	153	
expinv	152	
explike	153	
Exponential	89, 166	
exppdf	91, 152	
exprnd	134, 152, 153, 213	
exptat	152	
Extreme Value	89, 167	
eye	25, 27	
false	20, 25, 27, 186, 206	
fcontour	78	
fieldnames	61, 72	
filter	198, 199	
fimplicit	78	
fimplicit3	78	
FM	114, 184, 185	
fmdemod	185	
fmmod	185	
for	21, 23, 24, 49, 51, 58, 59, 84, 86, 89, 102, 103, 116, 119, 120, 125, 133, 138, 139, 140, 141, 142, 145, 153, 156, 159, 161, 166, 192, 193, 195, 208, 235, 237, 240, 241, 244, 249, 250	
fplot	78, 79, 80	
fplot3	78	
fprint	65	
fskdemod	191	
fskmod	191	
fsurf	78	
Gamma	89, 95, 158, 167, 168, 215	
gauspuls	162	
gen2par	178	
genqamdemod	191	
genqammmod	191	
geocdf	141	
geofit	141	
geoinv	141	
Geometric	89, 140, 167	
geopdf	141	
geornd	141	
geostat	141	
get	85	
gfweight	178	
gmonopuls	162	
grid on/off	83	
Giải mã mã xoắn	182, 183	
Giải mã số học	175	
Giải mã vi sai	171	
giải thuật đê quy	116	
giải thuật Lloyds	171, 172, 173	
Giải thuật loại bỏ	123, 124	
hàm area	91	
hàm Bessel điều chỉnh bậc một	217	
hàm Diriclet	162	
hàm Gamma	158, 215	
hàm Gamma chưa hoàn chỉnh chặn trên	215	
hàm phân bố xác suất tích lũy	88, 138, 141, 142, 144, 149, 152, 155, 159, 223	
Hàm phân phối tích lũy	138	
Hàm sinh đa thức	180	
hàm stem	91, 153	
hàm step	184	
hàm truyền hệ thống	12	
hammgen	178	
handle	78, 85	
hex2dec	60, 63	
hệ số kênh truyền phức	158, 211, 214, 217	
hệ thống đa anten	219	
hệ thống thông tin	13, 14, 15, 16, 18, 112, 163, 176, 182, 204, 212, 219, 223, 224	
hệ thống thực tế	11, 12	
Hiệp phương sai	129	
hiệu suất phô tàn	246, 247, 248, 260	
histfist	168	
histogram	78, 87, 117, 119, 121, 122, 134, 135, 136, 151, 165, 168	
Histograms	85	
hold on	82, 83, 91, 92, 117, 151, 153, 159, 160, 161, 173, 186	
Huffman	169, 175, 176	
Hypergeometric	89, 167	
icdf	159	
if	56, 58, 59, 60, 103, 117	
Image Processing	104	
imagesc	92, 93	
imread	92	
imshow	92	
ind2sub	25, 45	
intersect	61, 69	
inv	25, 52	
Inverse Gaussian	89, 167	
iscatastrophic	182	
iscell	61, 66	
ischar	60, 61, 62	
isequal	25, 196, 197	
isfinite	38	
isletter	60, 61, 62	
ismember	61, 69, 70	
isprime	25, 38, 39, 71	
isspace	61, 62	
isstrprop	60, 62	
isstruct	61, 72	
istrellis	182	
isvarname	61, 72	
keyboard	75	
Kênh nhiễu trắng	204, 230	
kênh truyền fading	13, 81, 136, 154, 155, 157, 203, 211, 212, 214, 216, 217, 221, 224, 225, 226, 227, 228, 229, 230, 231, 232, 234, 236, 238, 239, 240, 242,	

244, 245, 247, 248, 251, 252, 253, 254, 255, 257,
 258, 260
Kênh truyền fading Rician 115
Kênh truyền log-normal 209
Kênh truyền nhị phân 114, 115, 202, 203
 kênh truyền tương quan 219
Kênh truyền tương quan 219
 kết hợp lựa chọn 227, 233, 234, 258
 kết hợp theo tỷ lệ tối ưu 228, 238
Kỹ thuật chuyển tiếp 253
Kỳ vọng 126, 128
-law 169
 lập trình giao diện 97, 98, 102
legend 82, 83, 86, 92, 96, 125, 134, 135, 137, 139, 143,
 145, 153, 155, 157, 160, 161, 162, 170, 171, 173,
 174, 186, 188, 189, 192, 193, 195, 199, 205, 209,
 211, 213, 216, 219, 220, 227, 228, 229, 230, 232,
 235, 237, 240, 241, 245, 251, 256, 257
Lempel-Ziv-Welch 169
length 24, 25, 29, 30, 86, 91, 103, 111, 119, 120, 125,
 133, 138, 139, 145, 153, 156, 159, 161, 170, 174,
 192, 193, 195, 208, 227, 228, 229, 230, 232, 235,
 236, 237, 240, 241, 244, 249, 250, 256
linespec 79
lloyds 171, 172
load 21, 22, 92
 lọc tín hiệu số 198
loglog 77, 81
Loglogistic 89, 167
Lognormal 90, 167
logspace 25, 27
Logistic 89, 167
lookfor 23, 76
lower 60, 61, 62, 90, 168
 lỗi cú pháp 74, 75
 lỗi giải thuật 74
 lỗi lập trình 74
LTE 104
 mã BCH 114, 180, 201
 mã CRC 176, 177
 Mã hóa BCH 180
Mã hóa Huffman 175
 Mã hóa mã xoắn 182, 183
 Mã hóa nguồn 113, 169
Mã hóa số học 175, 176
 Mã hóa vi sai 171
 Mã khôi 178
 mã không gian thời gian 242, 244, 245, 259, 260
 mã LDPC 182
 mã Turbo 182, 183
 ma trận 15, 25, 26, 27, 29, 30, 31, 33, 35, 38, 40, 41, 42,
 46, 48, 49, 51, 52, 55, 56, 58, 61, 65, 66, 67, 71, 72,
 92, 93, 101, 102, 105, 106, 109, 128, 130, 133, 138,
 141, 142, 144, 150, 153, 155, 164, 165, 178, 179,
 181, 208
 MAC 14
magic 54, 55
 makedist 159, 160, 161

mảng cell 60, 65, 66, 72
Mảng cell 65
Mảng đa chiều 55
 mạng vô tuyến thế hệ mới 253
 Marcum Q 217, 218
marcumq 218, 219
 Markov 131, 132
mat2cell 61, 66
mat2str 60, 63, 64
MATLAB 3, 15, 16, 19, 24, 101
Mathematica 16, 231
max25, 33, 34, 56, 86, 120, 142, 153, 159, 161, 170, 171,
 173, 220, 228, 235, 237
mean25, 31, 32, 33, 39, 56, 89, 90, 127, 129, 130, 131,
 133, 134, 135, 136, 137, 146, 153, 159, 166, 167,
 196, 197, 220
meshgrid 25, 27
mil188demod 191
mil188qamod 191
min 25, 33, 34, 86
modnorm 191, 196, 197, 201
modulate 187, 188, 189
Monte Carlo 147, 225, 226
 mô hình kênh truyền 11
 mô hình mô phỏng 13, 14, 109
 mô hình toán học 12, 13, 15
 mô phỏng băng dài 183
 mô phỏng băng góc 183
 Mô phỏng máy tính 13
MRC 229, 238, 239, 240, 241, 242, 244, 258, 259
mskdemod 191
mskmod 191
 mục đích mô phỏng 14
 mức độ phức tạp 15, 226
mysetdiff 69
 Nakagami13, 90, 157, 158, 159, 160, 161, 167, 200, 214,
 215, 222, 258, 259, 260
ndims 24, 25, 55, 56
 Noncentral t 90, 167
normcdf 91, 150
normfit 150
norminv 150
normpdf 91, 150, 151
normrnd 150, 151
normtmat 150
ns-3 17, 18
null 42, 54
num2cell 61, 66
num2str 58, 60, 63, 102, 103, 104
numel 24, 25, 29, 30, 48, 49
oct2dec 182
ofdm demod 191
ofdm mod 191
ones24, 25, 27, 29, 35, 42, 49, 55, 56, 66, 83, 177, 199,
 230
orderfields 61, 72
orth 25, 54
pamdemod 191, 197, 201

pammod.....	191, 196, 197, 201
Pause on Errors	75
Pause on NaN or Inf	75
Pause on Warnings.....	75
pdf88, 89, 134, 135, 136, 137, 138, 139, 142, 151, 159, 160, 161	25, 61, 69
perm.....	25, 53
pinv	25, 53
plot76, 77, 78, 79, 80, 81, 82, 83, 86, 91, 92, 96, 101, 125, 134, 135, 137, 139, 142, 145, 151, 153, 155, 157, 159, 160, 161, 162, 163, 170, 171, 172, 173, 174, 185, 186, 199, 205, 220, 227, 230, 250, 251	77
plot3	77
plotmatrix	77
plotyy	77
PM	113, 114, 185
pmdemod.....	185
pmmod	185
poly2trellis.....	182
print	94
profile.....	75, 106
pskdemod191, 192, 194, 195, 201, 237, 241, 245, 250, 256	185
pskmod191, 192, 194, 195, 201, 237, 241, 244, 250, 256	185
pulstran.....	162
pwelch	188, 189
Phát hiện lỗi trong dữ liệu vào dùng CRC	177
Phân bố chuẩn.....	149
Phân bố đều	143
phân bố Gamma	158, 160, 215
phân bố Gauss.....	86, 90, 133, 134, 135, 136, 137, 212
Phân bố hình học.....	140
phân bố mũ.....	91, 123, 133, 134, 152, 153
Phân bố mũ.....	152
Phân bố Nakagami-m	157
Phân bố nhị thức	137
Phân bố Poison.....	141
Phân bố Poisson	141
Phân bố Rayleigh	154
Phân phối.....	89, 90, 137, 166, 167, 168
Phân phối Beta.....	89, 166
Phân phối Binomial.....	89, 166
Phân phối Birnbaum-Saunders.....	89, 166
phân phối Rician	218
phân tập thu	233, 242, 244
Phuong pháp mô phỏng.....	12
Phuong pháp phân tích đại số	12, 13
Phuong pháp số.....	12
Phuong pháp tổng hợp.....	118, 121
Phuong sai	127, 128, 129, 140, 152
qr	54
quá trình ngược	11
quantiz.....	170, 171, 172, 173
rand24, 25, 27, 55, 78, 84, 93, 116, 117, 119, 120, 122, 123, 125, 127, 128, 131, 132, 133, 134, 135, 136, 144, 146, 147, 148, 213, 215, 216, 218	163, 164, 180
randerr	163, 164, 180
randi59, 103, 129, 141, 144, 163, 164, 175, 178, 179, 180, 181, 182, 192, 195, 196, 197, 204, 206, 207, 208, 237, 241, 244, 250, 256	256
randn25, 27, 53, 86, 93, 94, 96, 101, 130, 150, 151, 157, 188, 189, 192, 195, 199, 207, 208, 213, 218, 220, 227, 228, 229, 230, 232, 235, 237, 240, 241, 244, 245, 250, 256	256
random	159, 161, 166, 168, 215, 218, 261
randperm	25
randsrc	163, 165, 166, 176, 204
rank	25, 54
raylcdf	155
Rayleigh13, 81, 90, 115, 136, 154, 155, 156, 157, 167, 203, 211, 212, 213, 214, 222, 224, 225, 226, 227, 228, 229, 230, 231, 232, 234, 236, 238, 239, 240, 242, 244, 245, 247, 248, 251, 252, 253, 254, 255, 257, 258, 259, 260	260
raylfit.....	155
raylinv	155
rayllylike	155
raylpdf	155
raylrnd	136, 155, 156, 157, 213, 222
rayltat	155
rectpuls	162
Reed-Solomon	181
repmat	25, 56, 60, 61, 62
reshape	25, 36, 46, 47, 48, 49
Rician....	13, 90, 115, 167, 203, 217, 218, 219, 222, 261
rmfield	61, 72
rref	25, 54
rsdec	181
rsenc	181
rsgenpoly	181
rsgenpolycoeffs	181
save	21, 22, 94
sawtooth	162, 163, 173, 174
scatter	77
scatter3	77
scatterplot	197, 198
Scilab	16, 18
selection combining.....	233, 235
semilogx	77, 81
semiology	77, 81
set 79, 85, 86, 91, 92, 96, 119, 121, 122, 125, 137, 139, 143, 145, 153, 155, 157, 160, 161, 162, 163, 168, 170, 171, 173, 174, 185, 186, 188, 189, 198, 199, 205, 209, 211, 214, 216, 219, 220, 227, 228, 229, 230, 232, 237, 241, 245, 250, 251, 256	256
setdiff	61, 69
sextror	61, 69
Shannon Fano	169
Signal Processing Toolbox	104
Simulink.....	16, 102, 107, 108, 109, 112, 113, 114, 115
sinc	162
size23, 24, 25, 26, 27, 29, 31, 43, 48, 55, 83, 86, 89, 96, 101, 125, 145, 153, 156, 159, 161, 167, 176, 181, 188, 189, 192, 195, 199, 207, 208, 237, 239, 241, 244, 249, 250	250

smart indent	74
So sánh chuỗi.....	67, 196, 197
sortrows.....	25, 60, 64
số giả ngẫu nhiên.....	116
sprint	65
square.....	51, 90, 162, 167
SSB	114, 185, 186, 187, 188
ssbdemod.....	185
ssbmod	185, 186
Stable	90, 167
std	83, 101, 128, 159
stem.....	91, 153, 160, 161, 162
Student's t	90, 167
str2num	60, 63, 64
strcat.....	58, 60, 64, 102, 103, 104
strcmp.....	61, 67, 68
strcmpi.....	61, 67, 68
strfind.....	61, 67, 68
strjust.....	60, 64
strmatch	61, 68
strncmp.....	61, 67, 68
strncmpi.....	61, 67, 68
strtok	61, 68
strtrim	60, 61, 62
struct	61, 72, 73, 74
strvcat.....	60, 64, 67
sub2ind	25, 45, 46
subplot.....	77, 84
sum25 , 31, 32, 86, 102, 103, 104, 125, 131, 132, 133, 134, 135, 136, 139, 140, 142, 145, 147, 148, 149, 151, 153, 156, 159, 161, 164, 170, 174, 177, 178, 179, 180, 181, 182, 192, 204, 209, 210, 213, 216, 218, 227, 228, 229, 230, 232, 235, 240, 241, 249, 250, 256	
suy hao đường truyền	212
suy hao đường truyền tự do.....	202
suy hao tín hiệu RF	202
suy hao tín hiệu RF do không khí.....	202
Sửa lỗi	74
svd	25, 54
switch.....	57, 58
symbolic	18, 97
syndtable	178, 179
TeX	94
tic	75, 103
tiêu chuẩn DVB-S.2	182
tín hiệu dạng chuỗi rời rạc	162
tín hiệu dạng hình răng cưa.....	162
tín hiệu điều chế Gaussian	162
tín hiệu ngẫu nhiên.....	113, 166, 182
tín hiệu xung vuông.....	111, 162
tính chính xác.....	15
title	80, 84, 237, 241, 245, 251, 256, 257
toc	75, 103, 104
toolbox	16, 92, 104
Tối ưu mã hóa vi sai	171
tỷ lệ lỗi bit	14, 81, 113, 114, 193, 194, 201, 208, 209, 212, 225, 226, 233, 236, 237, 239, 240, 241, 244, 246, 247, 248, 249, 250, 254, 255, 256, 258, 259, 260
Tỷ số tín hiệu trên nhiễu trung bình	213, 216, 219, 223, 226
tham số hiệu năng	14, 212, 223, 224, 225, 226, 259, 260
trace	25, 54
transpose	30
trạng thái hệ thống.....	12
tril	25, 27
tripuls	162
true	20, 24, 25, 27, 59, 106, 198, 206, 207
truyền giải mã và chuyển tiếp	253
truyền thích ứng	246, 247, 248
unifcdf	144, 145
unifit	144
unifinv	144
Uniform (Continuous)	90, 168
Uniform (Discrete)	90, 168
unipdf.....	144, 145
unirnd.....	144, 145
unitat	144
union	61, 69
unione	69
unique	61, 69, 71, 72
upper	60, 61, 62, 90, 168, 215, 216
var	127, 128, 129, 130, 131, 134, 135, 136, 137, 146, 150, 153, 206, 207, 208
vco	162
Vẽ đồ thị	76, 77, 78, 82, 86, 119, 121, 125, 137, 153, 159, 161, 165, 168, 194, 201, 208, 211, 227, 228, 230, 232, 245
<i>viễn thông</i>	3, 11, 15, 104, 226, 233
vitdec	182
Weibull	90, 168
while	21, 59, 60
Wifi	104
workspace	22, 75, 92
Workspace	19, 20, 21, 22, 114
Xác suất dừng	81, 223, 234, 240, 248, 252, 255, 257
Xác suất lỗi bit	224, 225
xác suất lỗi symbol.....	224
xlabel	78, 80, 81, 83, 86, 125, 134, 135, 137, 139, 142, 145, 153, 155, 157, 160, 161, 163, 171, 173, 192, 193, 195, 209, 211, 214, 216, 219, 220, 227, 228, 229, 230, 232, 235, 237, 240, 241, 245, 251, 256, 257
xuất chuỗi	65
xung cao tần.....	162
xung đơn Gauss	162
ylabel	78, 80, 81, 83, 86, 125, 134, 135, 137, 139, 142, 145, 153, 155, 157, 160, 161, 163, 192, 193, 195, 209, 211, 214, 216, 219, 220, 227, 228, 229, 232, 235, 237, 240, 241, 245, 251, 256, 257
zeros	25, 27, 31, 35, 42, 45, 46, 55, 86, 116, 119, 120, 125, 138, 139, 141, 142, 145, 153, 156, 159, 161, 181, 192, 195, 237, 239, 241, 244, 249, 250

PGS. TS. Võ Nguyễn Quốc Bảo

MÔ PHỎNG HỆ THỐNG TRUYỀN THÔNG

Chịu trách nhiệm xuất bản
ThS. VÕ TUẤN HẢI

Biên tập: NGUYỄN QUỲNH ANH

Ché bǎn: NGUYỄN MINH CHÂU

Họa sỹ bìa: ĐẶNG NGUYÊN VŨ

NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT

70 Trần Hưng Đạo - Hoàn Kiếm - Hà Nội

ĐT: 024 3942 2443 Fax: 024 3822 0658

Email: nxbkhkt@hn.vnn.vn

Website: <http://www.nxbkhkt.com.vn>

CHI NHÁNH NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT

28 Đồng Khởi - Quận 1 - TP Hồ Chí Minh

ĐT: 028 3822 5062

In 150 bản, khổ 19 × 26,5 cm, tại Công ty TNHH Trần Công.

Địa chỉ: Số 12 ngách 155/176 đường Trường Chinh, thành phố Hà Nội.

Số ĐKXB: 1251-2020/CXBIPH/5-32/ KHKT.

Quyết định XB số: 29/QĐ-NXBKHKT ngày 16 tháng 4 năm 2020.

In xong và nộp lưu chiểu năm 2020.

Mã ISBN: 978-604-67-1545-0