

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**GIÁO TRÌNH
TÀI LIỆU HƯỚNG DẪN THỰC HÀNH
TRUYỀN DỮ LIỆU**

Biên soạn: Thái Huy Tân

Nguyễn Huỳnh Quốc Việt

NHÀ XUẤT BẢN
ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

GIÁO TRÌNH
**TÀI LIỆU HƯỚNG DẪN THỰC HÀNH
TRUYỀN DỮ LIỆU**

Thái Huy Tân

Nguyễn Huỳnh Quốc Việt

NHÀ XUẤT BẢN ĐẠI HỌC QUỐC GIA
THÀNH PHỐ HỒ CHÍ MINH – NĂM 2022

PHẦN MỞ ĐẦU

Tài liệu hướng dẫn Thực hành Truyền dữ liệu cung cấp các nội dung cơ bản nhằm trực quan hóa các nội dung lý thuyết thông qua hai công cụ chính: Matlab và Simulink.

Matlab đưa ra các hàm, phương thức hỗ trợ để có thể lập trình nhằm biểu diễn các nội dung đã học dưới dạng hình vẽ, biểu đồ.

Simulink cung cấp các khối giả lập cho phép kết hợp, hiệu chỉnh nhằm thiết kế nên những mô hình chỉ thông qua việc kéo thả. Thông qua việc thiết kế các mô hình này, sinh viên có thể nắm rõ thành phần cấu tạo cũng như cách thức hoạt động của những kỹ thuật điều chế dữ liệu.

MỤC LỤC

| | |
|--|----|
| PHẦN MỞ ĐẦU | i |
| MỤC LỤC | ii |
| DANH MỤC HÌNH ẢNH..... | vi |
| BÀI 1. HƯỚNG DẪN SỬ DỤNG MATLAB VÀ SIMULINK 1 | |
| 1.1 Hướng dẫn sử dụng Matlab | 1 |
| 1.1.1 Một số thao tác cơ bản trong MATLAB | 4 |
| 1.1.2 Khai báo biến, các phép toán và toán tử | 5 |
| 1.1.3 Thao tác với ma trận..... | 6 |
| 1.1.4 Xử lý ma trận..... | 7 |
| 1.1.5 Đồ họa | 8 |
| 1.1.6 Lập trình trong Matlab | 10 |
| 1.2 Hướng dẫn sử dụng Simulink | 12 |
| 1.3 Thực hành | 19 |
| 1.3.1 Phần thực hành tại lớp | 19 |
| 1.3.2 Phần thực hành về nhà..... | 19 |
| BÀI 2. TRUYỀN TẢI THÔNG TIN | 21 |
| 2.1 Tín hiệu tuần tự, tín hiệu số, tín hiệu tuần hoàn | 21 |
| 2.1.1 Tín hiệu tuần tự (Analog signal) | 21 |

| | | |
|--------|--|----|
| 2.1.2 | Tín hiệu số (Digital signal)..... | 22 |
| 2.1.3 | Tín hiệu tuần hoàn (Periodic signal) | 24 |
| 2.2 | Tín hiệu tuần hoàn | 25 |
| 2.2.1 | Sóng Sin (Sine signal) | 25 |
| 2.2.2 | Sóng vuông (Sine signal) | 28 |
| 2.3 | Biểu diễn sóng trong miền thời gian và miền tần số | 30 |
| 2.4 | Tổng hợp tín hiệu số từ tín hiệu tuần tự | 33 |
| 2.5 | Sự suy hao trong không gian | 39 |
| 2.6 | Nhiễu trong truyền tải thông tin | 41 |
| 2.7 | Thực hành | 44 |
| 2.7.1 | Phần thực hành tại lớp..... | 44 |
| 2.7.2 | Phần thực hành về nhà..... | 44 |
| BÀI 3. | CÁC KỸ THUẬT MÃ HÓA DỮ LIỆU SỐ | |
| | 46 | |
| 3.1 | Các kỹ thuật mã hóa dữ liệu số sang tín hiệu số..... | 46 |
| 3.1.1 | Kỹ thuật mã hóa NRZ – L..... | 46 |
| 3.1.2 | Kỹ thuật mã hóa NRZI | 49 |
| 3.1.3 | Kỹ thuật mã hóa Bipolar – AMI..... | 52 |
| 3.1.4 | Kỹ thuật mã hóa Pseudoternary | 54 |
| 3.1.5 | Kỹ thuật mã hóa Manchester..... | 57 |
| 3.1.6 | Kỹ thuật mã hóa Differential Manchester | 59 |

| | | |
|---------|--|-----|
| 3.1.7 | Kỹ thuật mã hóa B8ZS | 62 |
| 3.1.8 | Kỹ thuật mã hóa HDB3 | 65 |
| 3.2 | Các kỹ thuật mã hóa dữ liệu số sang tín hiệu tuần tự | |
| | 69 | |
| 3.2.1 | Amplitude-shift Keying (ASK) | 70 |
| 3.2.2 | Frequency-Shift Keying (FSK) | 72 |
| 3.2.3 | Phase-Shift Keying (PSK)..... | 76 |
| 3.2.4 | Quadrature Phase-Shift Keying (QPSK)..... | 79 |
| 3.2.5 | QAM Quadrature amplitude modulation | 85 |
| 3.3 | Thực hành | 90 |
| 3.3.1 | Phần thực hành tại lớp | 90 |
| 3.3.2 | Phần thực hành về nhà..... | 90 |
| BÀI 4. | CÁC KỸ THUẬT MÃ HÓA DỮ LIỆU | |
| TUẦN TỰ | 91 | |
| 4.1 | Các kỹ thuật mã hóa dữ liệu tuần tự sang tín hiệu số | |
| | 91 | |
| 4.1.1 | Điều chế xung mã PCM (Pulse Code Modulation) | |
| | 91 | |
| 4.1.2 | Điều chế Delta DM (Delta Modulation) | 98 |
| 4.2 | Các kỹ thuật mã hóa dữ liệu tuần tự sang tín hiệu tuần | |
| tự | 101 | |
| 4.2.1 | Điều chế biên độ AM (Amplitude Modulation). | 101 |

| | | |
|-------------------------|---|-----|
| 4.2.2 | Điều chế tần số FM (Frequency Modulation) | 105 |
| 4.2.3 | Điều chế pha PM (Phase Frequency Modulation) | |
| | 107 | |
| 4.3 | Thực hành | 111 |
| 4.3.1 | Phần thực hành tại lớp..... | 111 |
| 4.3.2 | Phần thực hành về nhà..... | 111 |
| BÀI 5. | CÁC KỸ THUẬT GHÉP KÊNH | 112 |
| 5.1 | Ghép kênh phân chia theo tần số FDM (Frequency Division Multiplexing) | 112 |
| 5.1.1 | Mô hình bộ ghép kênh FDM sử dụng kỹ thuật điều chế AM | 115 |
| 5.1.2 | Mô hình bộ ghép kênh FDM sử dụng kỹ thuật điều chế FM | 121 |
| 5.2 | Ghép kênh đồng bộ phân chia theo thời gian TDM | |
| | 128 | |
| 5.3 | Thực hành | 137 |
| 5.3.1 | Phần thực hành tại lớp..... | 137 |
| 5.3.2 | Phần thực hành về nhà..... | 137 |
| TÀI LIỆU THAM KHẢO..... | | 138 |

DANH MỤC HÌNH ẢNH

| | |
|--|----|
| Hình 1-1: Màn hình khởi động của phần mềm Matlab .. | 2 |
| Hình 1-2: Thanh công cụ chứa biểu tượng Simulink library | 13 |
| Hình 1-3: Cửa sổ chính thư viện Simulink..... | 13 |
| Hình 1-4: Vị trí của icon New Model..... | 13 |
| Hình 1-5: Màn hình kéo thả các khối giả lập được hỗ trợ bởi Simulink | 14 |
| Hình 1-6: Minh họa việc lưu trữ mô hình đã tạo | 14 |
| Hình 1-7: Minh họa việc kết nối giữa các khối giả lập | 15 |
| Hình 1-8: Minh họa việc chạy mô hình thiết kế | 15 |
| Hình 1-9: Minh họa kết quả chạy mô hình thiết kế | 16 |
| Hình 1-10: Hiệu chỉnh thông số của khối hằng số constant | 17 |
| Hình 1-11: Các thông số mô phỏng của Simulink | 18 |
| Hình 2-1: Tín hiệu tuần tự | 22 |
| Hình 2-2: Tín hiệu số | 24 |
| Hình 2-3: Tín hiệu tuần hoàn | 25 |
| Hình 2-4: Tín hiệu Sin với pha là $0, \pi/2, \pi$ | 27 |
| Hình 2-5: Tín hiệu Sin với pha là $\pi/4, 3\pi/4, 5\pi/4$ | 28 |

| | |
|---|----|
| Hình 2-6: Tín hiệu sóng vuông có Duty cycle 50% và 30% | 29 |
| Hình 2-7: Tín hiệu Sin trong miền thời gian có tần số 15Hz và 20Hz | 32 |
| Hình 2-8: Tín hiệu Sin trong miền tần số | 33 |
| Hình 2-9: Mô hình kết hợp hai sóng sin khi $k = 3$ | 34 |
| Hình 2-10: Tín hiệu $s(t)$ đầu ra khi $k = 3$ | 34 |
| Hình 2-11: Mô hình kết hợp ba sóng sin khi $k = 5$ | 35 |
| Hình 2-12: Tín hiệu $s(t)$ đầu ra khi $k = 5$ | 36 |
| Hình 2-13: Mô hình kết hợp 6 sóng khi $k = 11$ | 37 |
| Hình 2-14: Tín hiệu $s(t)$ đầu ra khi $k = 11$ | 38 |
| Hình 2-15: Biểu đồ thể hiện sự suy hao trong khoảng cách 1000m ứng với tần số tín hiệu là 2400 Hz | 41 |
| Hình 2-16: Sự ảnh hưởng của nhiễu Gaussian với tín hiệu tuần tự | 43 |
| Hình 3-1: Tín hiệu số tương ứng với dữ liệu đầu vào theo kỹ thuật NRZ – L | 49 |
| Hình 3-2: Tín hiệu số tương ứng với dữ liệu đầu vào theo kỹ thuật NRZI..... | 52 |
| Hình 3-3: Tín hiệu số tương ứng với dữ liệu đầu vào theo kỹ thuật Bipolar – AMI | 54 |

| | |
|--|----|
| Hình 3-4: Tín hiệu số tương ứng với dữ liệu đầu vào theo thuật toán Pseudoternary | 57 |
| Hình 3-5: Tín hiệu số tương ứng với dữ liệu đầu vào theo thuật toán Manchester | 59 |
| Hình 3-6: Tín hiệu số tương ứng với dữ liệu đầu vào theo thuật toán Differential Manchester..... | 62 |
| Hình 3-7: Tín hiệu số tương ứng với dữ liệu đầu vào theo thuật toán B8ZS | 65 |
| Hình 3-8: Tín hiệu số tương ứng với dữ liệu đầu vào theo thuật toán HDB3..... | 69 |
| Hình 3-9: Quá trình điều chế ASK | 70 |
| Hình 3-10: Sơ đồ mô phỏng hệ thống điều chế ASK... | 71 |
| Hình 3-11: Biểu diễn tín hiệu ASK | 72 |
| Hình 3-12: Quá trình điều chế FSK..... | 73 |
| Hình 3-13: Sơ đồ mô phỏng hệ thống điều chế FSK ... | 74 |
| Hình 3-14: Biểu diễn tín hiệu FSK..... | 75 |
| Hình 3-15: Quá trình điều chế PSK..... | 76 |
| Hình 3-16: Sơ đồ mô phỏng hệ thống điều chế PSK ... | 77 |
| Hình 3-17: Biểu diễn tín hiệu PSK..... | 79 |
| Hình 3-18: Biểu diễn dạng sóng và lược đồ chòm sao của tín hiệu QPSK..... | 79 |
| Hình 3-19: Sơ đồ nguyên lý hệ thống điều chế QPSK. | 81 |

| | |
|--|-----|
| Hình 3-20: Sơ đồ mô phỏng hệ thống điều chế QPSK. | 83 |
| Hình 3-21: Tín hiệu điều chế QPSK với các pha $\pi/4$, $3\pi/4$, $5\pi/4$ ($-3\pi/4$), $7\pi/4$ ($-\pi/4$). | 84 |
| Hình 3-22: Biểu diễn dạng sóng và lược đồ chòm sao của tín hiệu QAM | 85 |
| Hình 3-23: Sơ đồ nguyên lý hệ thống điều chế QPSK. | 86 |
| Hình 3-24: Sơ đồ mô phỏng hệ thống điều chế QAM . | 87 |
| Hình 3-25: Tín hiệu điều chế QAM | 89 |
| Hình 4-1: Mô hình điều chế xung mã PCM | 92 |
| Hình 4-2: Quá trình điều chế và giải điều chế tín hiệu tuần tự sử dụng PCM | 97 |
| Hình 4-3: Quá trình điều chế sử dụng Delta Modulation | 98 |
| Hình 4-4: Quá trình điều chế sử dụng Delta Modulation | 100 |
| Hình 4-5: Dạng tín hiệu điều chế AM | 101 |
| Hình 4-6: Sơ đồ mô phỏng hệ thống điều chế AM | 102 |
| Hình 4-7: Dạng tín hiệu điều chế AM với hệ số điều chế $n_a = 0.8$ | 104 |
| Hình 4-8: Dạng tín hiệu điều chế AM với hệ số điều chế $n_a = 1.2$ | 105 |
| Hình 4-9: Sơ đồ mô phỏng hệ thống điều chế FM | 106 |

| | |
|--|-----|
| Hình 4-10: Biểu diễn dạng tín hiệu điều chế FM | 107 |
| Hình 4-11: Bộ điều chế PM sử dụng simulink | 108 |
| Hình 4-12: Biểu diễn dạng tín hiệu điều chế PM với độ lệch pha $K_C = \pi/2$ | 110 |
| Hình 4-13: Biểu diễn dạng tín hiệu điều chế PM với độ lệch pha $K_C = \pi$ | 110 |
| Hình 5-1: Sơ đồ nguyên lý bộ ghép kênh FDM | 113 |
| Hình 5-2: Phổ tín hiệu điều chế dải nền tổng hợp | 114 |
| Hình 5-3: Sơ đồ nguyên lý bộ thu và tách kênh FDM | 115 |
| Hình 5-4: Mô hình Simulink bộ ghép kênh FDM sử dụng điều biên AM | 116 |
| Hình 5-5: Biểu diễn 3 tín hiệu đầu vào..... | 118 |
| Hình 5-6: Ba tín hiệu điều chế AM tương ứng với 3 tín hiệu đầu vào | 118 |
| Hình 5-7: Biểu diễn tín hiệu tổng hợp sau ghép kênh ở đầu ra | 119 |
| Hình 5-8: Mô hình Simulink bộ tách kênh FDM sử dụng điều biên AM..... | 119 |
| Hình 5-9: Biểu diễn 3 tín hiệu (mang dữ liệu) sau khi tách kênh | 121 |
| Hình 5-10: Mô hình Simulink bộ ghép kênh FDM sử dụng điều chế FM..... | 122 |

| | |
|--|-----|
| Hình 5-11: Biểu diễn 3 tín hiệu đầu vào..... | 123 |
| Hình 5-12: Ba tín hiệu điều chế FM tương ứng với 3 tín hiệu đầu vào | 123 |
| Hình 5-13: Biểu diễn tín hiệu tổng hợp sau ghép kênh ở đầu ra (FM)..... | 124 |
| Hình 5-14: Mô hình Simulink bộ tách kênh FDM sử dụng điều tần FM | 125 |
| Hình 5-15: Biểu diễn 3 tín hiệu đầu vào..... | 127 |
| Hình 5-16: Biểu diễn tín hiệu tổng hợp sau ghép kênh FM ở đầu ra | 127 |
| Hình 5-17: Biểu diễn 3 tín hiệu mang dữ liệu sau khi tách kênh FM | 128 |
| Hình 5-18: Mô hình hoạt động của hệ thống ghép kênh đồng bộ phân chia theo thời gian | 129 |
| Hình 5-19: Tín hiệu đầu vào sử dụng để ghép kênh đồng bộ theo TDM | 134 |
| Hình 5-20: Tín hiệu tổng hợp sử dụng ghép kênh đồng bộ theo TDM | 135 |
| Hình 5-21: Dữ liệu được chia ra về từng nguồn nhận tương ứng | 136 |

BÀI 1. HƯỚNG DẪN SỬ DỤNG MATLAB VÀ SIMULINK

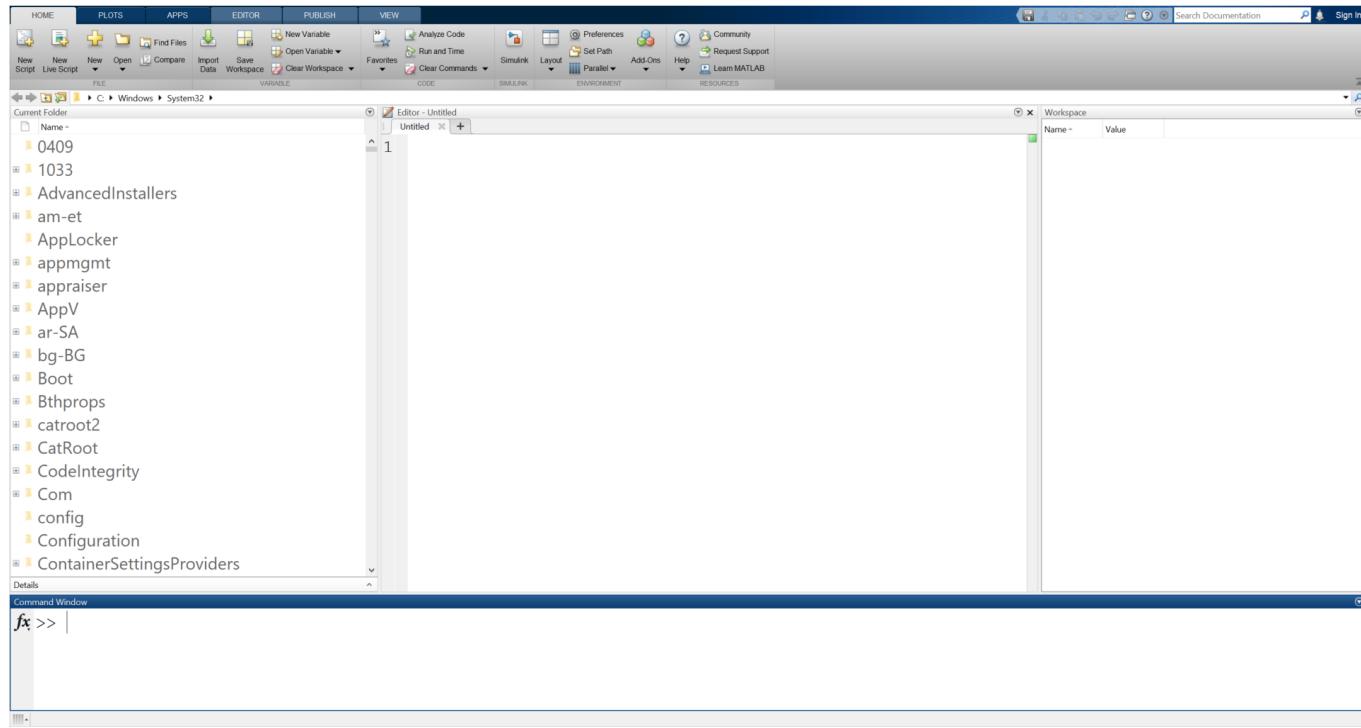
Sau khi học xong bài này, sinh viên có thể:

- Sử dụng phần mềm Matlab.
- Thực hiện tạo các script file hay function và lưu trữ trên Matlab.
- Biết tạo mô hình cơ bản bằng Simulink trên Matlab.

1.1 Hướng dẫn sử dụng Matlab

MATLAB (**M**atrix **L**aboratory) [2] là một phần mềm khoa học được thiết kế để cung cấp việc tính toán số và hiển thị đồ họa bằng ngôn ngữ lập trình cấp cao. Matlab cung cấp các tính năng tương tác tuyệt vời cho phép người sử dụng thao tác dữ liệu linh hoạt dưới dạng ma trận để tính toán và quan sát. Các dữ liệu đầu vào của Matlab có thể được nhập từ "Command line" hoặc từ "m files", trong đó tập lệnh được cho trước bởi Matlab.

Matlab cung cấp cho người dùng các toolbox tiêu chuẩn tùy chọn. Người dùng cũng có thể tạo ra các hộp công cụ riêng của mình gồm các "mfiles" được viết cho các ứng dụng cụ thể.



Hình 1-1: Màn hình khởi động của phần mềm Matlab

Vị trí của các cửa sổ con như workspace, current folder, command window được hiển thị từ khi chạy Matlab và có thể di chuyển thay đổi vị trí cũng như đóng mở theo ý muốn người sử dụng.

Command Window: Đây là cửa sổ làm việc chính của Matlab. Tại đây ta thực hiện toàn bộ việc nhập dữ liệu và xuất kết quả tính toán. Dấu nháy >> báo hiệu chương trình sẵn sàng cho việc nhập dữ liệu. Ta kết thúc việc nhập dữ liệu bằng cách nhấn phím Enter. Matlab sẽ thực thi dòng lệnh mà ta nhập vào *Command Window* và trả kết quả trong *Command Window*.

Workspace browser: Trong Matlab các dữ liệu được lưu trong biến. *Workspace browser* liệt kê tất cả các biến mà ta đang sử dụng trong Matlab. Nó cung cấp thông tin về kích thước, loại dữ liệu. Ta có thể truy cập trực tiếp vào dữ liệu bằng cách nhấn kép vào biến để hiển thị *Array editor*.

Launch pad: Cho phép người dùng truy cập nhanh vào các bộ *Toolbox*, phần *Help*.

Editor window: dùng để soạn thảo và debug các M-file của Matlab.

Current Folder: xem các file trong thư mục hiện hành. Lưu ý, khi thực thi (run) file script hay gọi hàm mà không đưa đường dẫn chi tiết thì các file script hay hàm phải được để ở trong thư mục hiện hành.

■ Một số thao tác cơ bản trong MATLAB

Trong Matlab, thanh trình đơn thay đổi tùy theo cửa sổ mà ta lựa chọn. Tuy vậy các trình đơn *File*, *Desktop*, *Window*, *Help* có mặt hầu hết trong các thanh trình đơn.

Trình đơn File

- *New*: tạo một đối tượng mới (biến, m-file, figure, model, GUI).
- *Open*: mở một file theo định dạng của MATLAB (*.m, *.mat, *.mdl)
- *Import data...*: nhập dữ liệu từ các file khác vào MATLAB.
- *Save workspace...*: lưu các biến trong MATLAB vào file *.mat.
- *Set path*: khai báo các đường dẫn của các thư mục chứa các m-file.
- *Preferences*: thay đổi các định dạng về font, font size, color cũng như các tùy chọn cho Editor, Command Window v.v.
- *Page Setup*: định dạng trang in.
- *Print*: in.

Trình đơn Desktop

- *Desktop layout*: sắp xếp các cửa sổ trong giao diện.
- *Save layout*: lưu cách sắp xếp cửa sổ.

■ Khai báo biến, các phép toán và toán tử

Khai báo biến

- Phân biệt chữ hoa và chữ thường.
- Không cần phải khai báo kiểu biến.
- Tên biến phải bắt đầu bằng ký tự và không được có khoảng trắng.
- Không đặt tên trùng với các tên đặc biệt của Matlab.
- Để khai báo biến toàn cục (sử dụng được trong tất cả chương trình con), phải dùng thêm từ khoá **global** phía trước

Các phép toán và toán tử

- Các phép toán: + , - , * , / , \ (chia trái) , ^ (mũ) , ' (chuyển vị hay số phức liên hiệp).
- Các toán tử quan hệ : < , <= , > , >= , == , ~=
- Các toán tử logic : & , | (or) , ~ (not)
- Các hằng:
 - + pi 3.14159265
 - + i, j: số ảo.
 - + inf: vô cùng.
 - + NaN: không phải là số

Chú ý:

Các lệnh kết thúc bằng dấu chấm phẩy, Matlab sẽ không thể hiện kết quả trên màn hình.

Các chú thích được đặt phía sau dấu %.

Trong quá trình nhập nếu các phần tử trên một hàng dài quá ta có thể xuống dòng bằng toán tử ba chấm (...).

■ Thao tác với ma trận

Ma trận là một mảng có m hàng và n cột. Trường hợp ma trận chỉ có một phần tử (ma trận 1x1) ta có một số. Ma trận chỉ có một cột hay một hàng được gọi là một vector.

- Khi nhập ma trận ta phải tuân theo các quy định sau:
 - + Ngăn cách các phần tử trong một hàng của ma trận bằng dấu “,” hay khoảng trắng.
 - + Dùng dấu “;” để kết thúc một hàng và bắt đầu hàng kế tiếp.
 - + Các phần tử của ma trận phải để trong cặp dấu ngoặc vuông [].
- Phần tử ở hàng i cột j của ma trận có ký hiệu là A(i,j). Lưu ý rằng các chỉ số của ma trận thường bắt đầu từ 1.
- Toán tử ‘:’ là một toán tử quan trọng của Matlab. Nó xuất hiện ở nhiều dạng khác nhau.

>>1:10 %là một vector hàng chứa 10 số nguyên từ 1 đến 10

>>100:-7:50 %tạo dãy số từ 100 đến 51, cách đều nhau 7

>>0: pi/4: pi %tạo một dãy số từ 0 đến π , cách đều nhau $\pi/4$

- Các biểu thức chỉ số có thể tham chiếu tới một phần của ma trận. $A(1:k, j)$ xác định k phần tử đầu tiên của cột j. Ngoài ra toán tử “:” tham chiếu tới tất cả các phần tử của một hàng hay một cột.

`>>A(:,3)`

`>>A(3, :)`

`>>B = A(:, [1 3 2 4])` %tạo ma trận B từ ma trận A bằng cách đổi thứ tự các cột từ [1 2 3 4] thành [1 3 2 4]

`>>b(:, 2) = [] ;` %xoá cột thứ 2

- Matlab cung cấp một số hàm để tạo các ma trận cơ bản:
 - + ***zeros*** tạo ra ma trận mà các phần tử đều là 0.
 - + ***ones*** tạo ra ma trận mà các phần tử đều là 1.
 - + ***rand*** tạo ra ma trận mà các phần tử ngẫu nhiên phân bố đều.
 - + ***randn*** tạo ra ma trận mà các phần tử ngẫu nhiên phân bố chuẩn.

■ Xử lý ma trận

- Cộng: $X = A + B$
- Trừ: $X = A - B$

- Nhân: $X = A * B$
 - + $X = A.*B$ nhân các phần tử tương ứng với nhau, yêu cầu 2 ma trận A và B phải có cùng kích thước.
- Chia : $X = A/B$ lúc đó $X = A * \text{inv}(B)$
 - + $X = A\backslash B$ lúc đó $X = \text{inv}(A) * B$
 - + $X=A./B$ chia các phần tử tương ứng với nhau, 2 ma trận A và B có cùng kích thước.
- Luỹ thừa : $X = A.^2$

$$X = A.^2$$

■ Đồ họa

Matlab cung cấp một loạt hàm để vẽ biểu diễn các vector cũng như giải thích và in các đường cong này.

- **plot**: đồ họa 2-D với số liệu 2 trực vô hướng và tuyến tính.
- **plot3**: đồ họa 3-D với số liệu 2 trực vô hướng và tuyến tính.
- **loglog**: đồ họa với các trục x, y ở dạng logarit.
- **semilogx**: đồ họa với trục x logarit và trục y tuyến tính.
- **semilogy**: đồ họa với trục y logarit và trục x tuyến tính.

Tạo hình vẽ

Hàm plot có các dạng khác nhau phụ thuộc vào các đối số đưa vào. Ví dụ nếu y là một vector thì plot(y) tạo ra một đường quan hệ giữa các giá trị của y và chỉ số của nó. Nếu ta có 2 vector x và y thì plot(x, y) tạo ra đồ thị quan hệ giữa x và y.

```
>>t = [0:pi/100:2*pi];  
>>y = sin(t);  
>>plot(t,y); % Vẽ hình sin từ 0->2π
```

Ta có thể dùng các kiểu đường vẽ khác nhau khi vẽ hình.

```
>>plot(t,y, '. ') % vẽ bằng đường chấm chấm
```

Các dạng đường thẳng xác định bằng:

‘-’ đường liền; ‘--’ đường đứt nét; ‘:’ đường chấm chấm;
‘-.’ đường chấm gạch

Thao tác với các trực tọa độ

Khi ta tạo một hình vẽ, MATLAB tự động chọn các giới hạn trên trực tọa độ và khoảng cách đánh dấu dựa trên dữ liệu dùng để vẽ. MATLAB cung cấp các lệnh ghi nhãn lên đồ họa gồm:

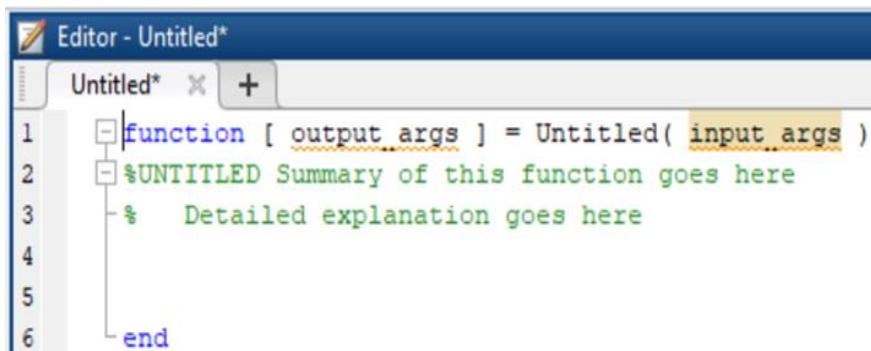
- ***title*** thêm nhãn vào đồ họa.
- ***xlabel*** thêm nhãn vào trực x.
- ***ylabel*** thêm nhãn vào trực y.

- *zlabel* thêm nhãn vào trục z.
- *legend* thêm chú giải vào đồ thị.
- *text* hiển thị chuỗi văn bản ở vị trí nhất định.

■ Lập trình trong Matlab

- Tạo hàm trong Matlab

Sử dụng trình đơn Home → New → Function (hoặc Script) mở ra Editor window nhằm soạn thảo hàm (function) hoặc tập hợp các lệnh sẽ được thực thi hàng loạt (Script file) sẽ được lưu lại ở dạng M file.



```

Editor - Untitled*
Untitled* X +
1 function [ output args ] = Untitled( input args )
2 %UNTITLED Summary of this function goes here
3 % Detailed explanation goes here
4
5
6 end

```

Hình 1-2: Cửa sổ soạn thảo hàm trong Matlab

Trong đó nơi chúng ta đặt tên hàm (Untitled) là tên được lưu lại của M file ở ổ đĩa cứng.

- Câu lệnh if, else, elseif

Cú pháp của if như sau:

if <biểu thức điều kiện>
<phát biểu>

End

Nếu *<biểu thức điều kiện>* cho kết quả đúng thì phần lệnh trong thân của *if* được thực hiện. Các phát biểu *else* và *elseif* cũng tương tự.

- Câu lệnh **switch**

Cú pháp của switch như sau:

switch <biểu thức>

case n1

<lệnh 1>

case n2

<lệnh 2>

.....

case nn

<lệnh n>

Otherwise

<lệnh n+1>

end

- Câu lệnh **while**

Vòng lặp while dùng khi không biết trước số lần lặp.

Cú pháp của nó như sau:

while <biểu thức>

<phát biểu>

end

- Câu lệnh **For**

Vòng lặp for dùng khi biết trước số lần lặp. Cú pháp như sau:

```
for <chi số>=<giá trị đầu>:<mức tăng>:<giá trị cuối>  
<phát biểu>  
end
```

- Câu lệnh **break**

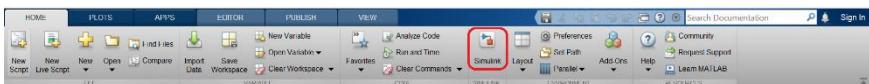
Phát biểu break để kết thúc vòng lặp for hay while mà không quan tâm đến điều kiện kết thúc vòng lặp đã thoả mãn hay chưa.

1.2 Hướng dẫn sử dụng Simulink

Simulink [3] là một công cụ trong Matlab dùng để tạo mô hình, mô phỏng và phân tích các hệ thống với môi trường giao diện sử dụng bằng đồ họa. Việc xây dựng mô hình được đơn giản hóa bằng các hoạt động nhấp chuột và kéo thả. Simulink bao gồm một bộ thư viện khối với các hộp công cụ toàn diện cho cả việc phân tích tuyến tính và phi tuyến. Simulink là một phần quan trọng của Matlab và có thể dễ dàng chuyển đổi qua lại trong quá trình phân tích, và vì vậy người dùng có thể tận dụng được ưu thế của cả hai môi trường.

Có thể mở Simulink bằng 2 cách:

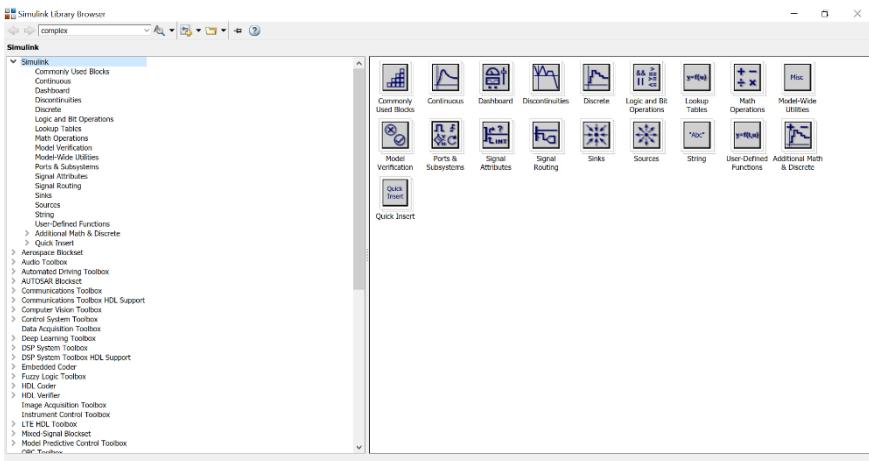
- Click vào biểu tượng Simulink library như hình dưới (Simulink icon).



Hình 1-3: Thanh công cụ chứa biểu tượng Simulink library

- Từ cửa sổ lệnh, đánh lệnh **Simulink** và enter.

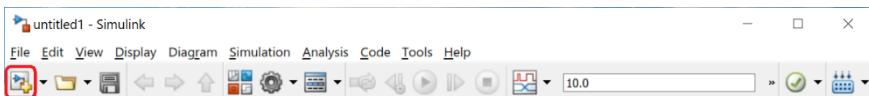
Cửa sổ thư viện Simulink sẽ hiển thị như sau:



Hình 1-4: Cửa sổ chính thư viện Simulink

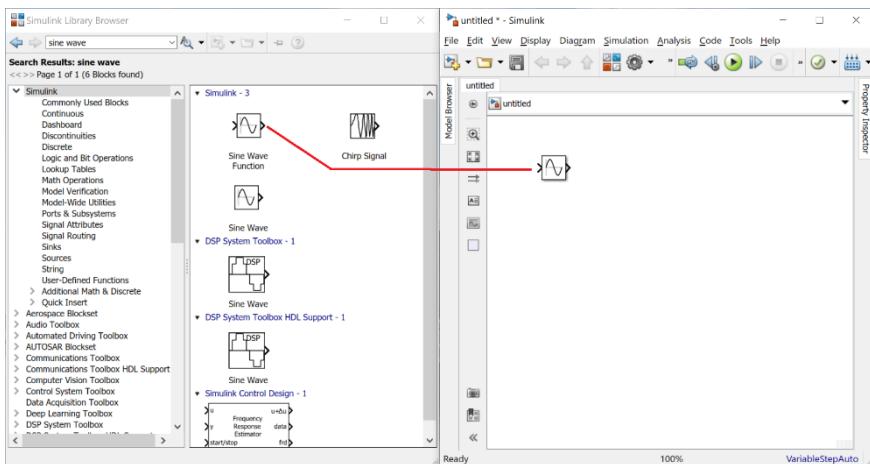
Tạo một mô hình mới bằng cách:

- Click vào icon **New model** hoặc gõ **Ctrl-N** hoặc click vào Menu **File → New → Model**



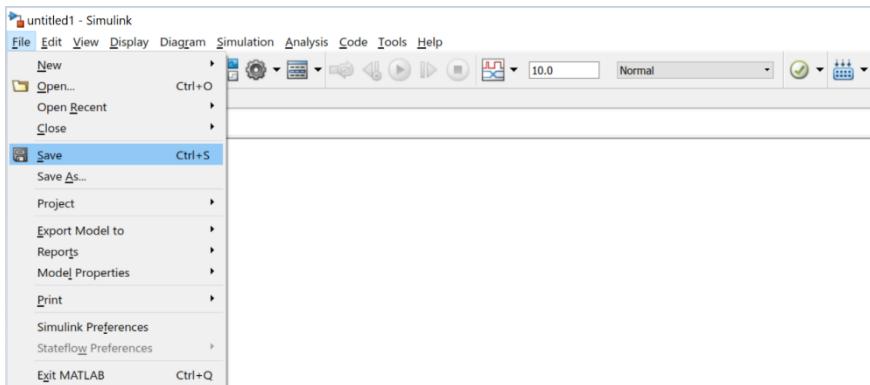
Hình 1-5: Vị trí của icon New Model

Lúc này, một màn hình mới được tạo bao gồm các khối mà Simulink hỗ trợ sẵn. Người dùng tạo các khối bằng cách kéo thả từ thư viện vào cửa sổ thao tác.



Hình 1-6: Màn hình kéo thả các khối giả lập được hỗ trợ bởi Simulink

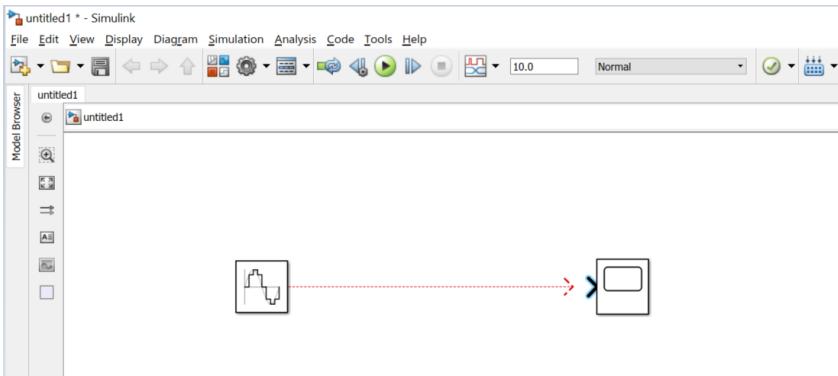
Người dùng có thể lưu trữ mô hình bằng lệnh Save (**File → Save**) hoặc nhấp vào icon Save.



Hình 1-7: Minh họa việc lưu trữ mô hình đã tạo

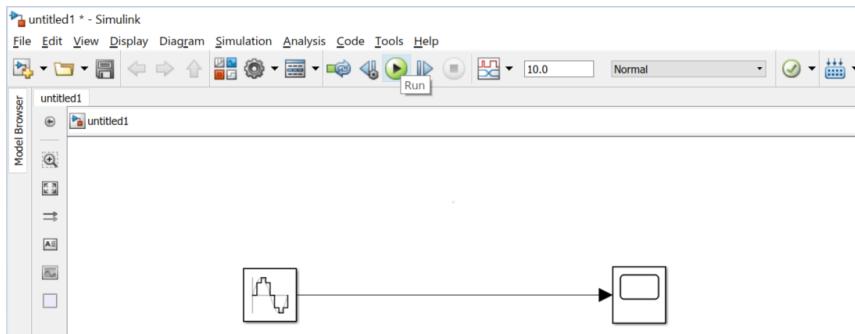
Để kết nối giữa các khối, người dùng đưa con chuột tới ngõ ra của khối (dấu “>”), khi đó con chuột sẽ có dạng “+”.

Kéo rê chuột tới ngõ vào của một khối khác và thả ra để kết nối tín hiệu.



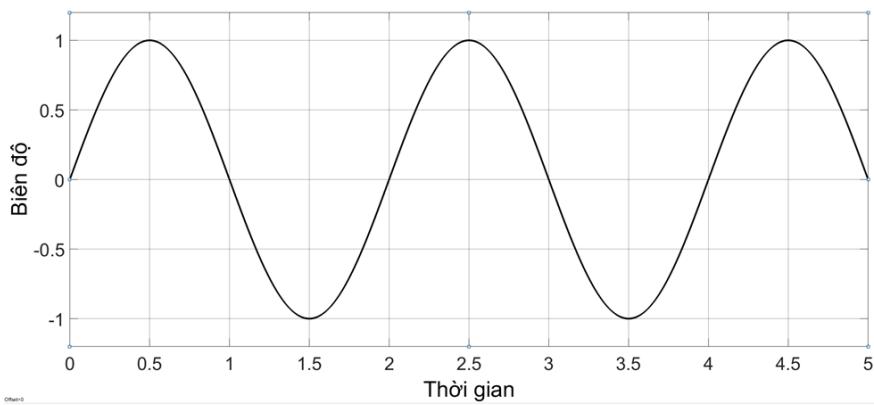
Hình 1-8: Minh họa việc kết nối giữa các khối giả lập

Lệnh Start (Menu **Simulation → Run**) được sử dụng để chạy mô phỏng mô hình. Ngoài ra, người dùng còn có thể nhấp chuột vào icon **Run** để bắt đầu việc mô phỏng.



Hình 1-9: Minh họa việc chạy mô hình thiết kế

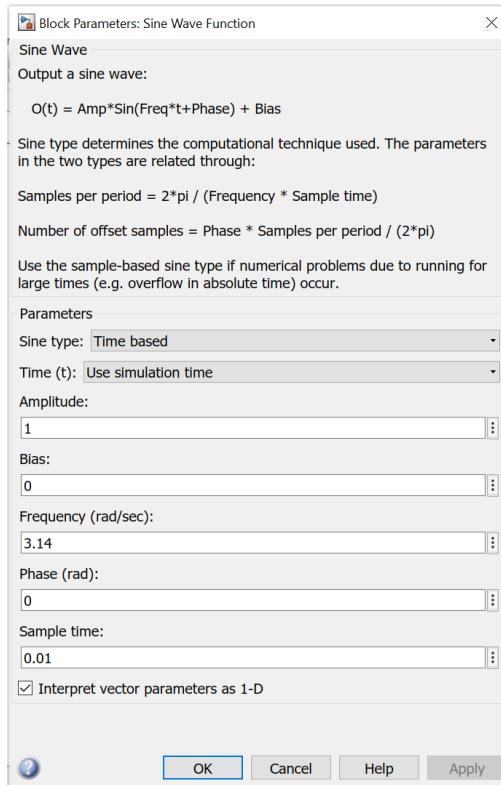
Sau đó, người dùng có thể xem tín hiệu bằng cách nhấp đúp vào khối **Scope**.



Hình 1-10: Minh họa kết quả chạy mô hình thiết kế

Người dùng có thể hiệu chỉnh thông số của một khối bằng cách nhấp đúp vào khối cần chỉnh.

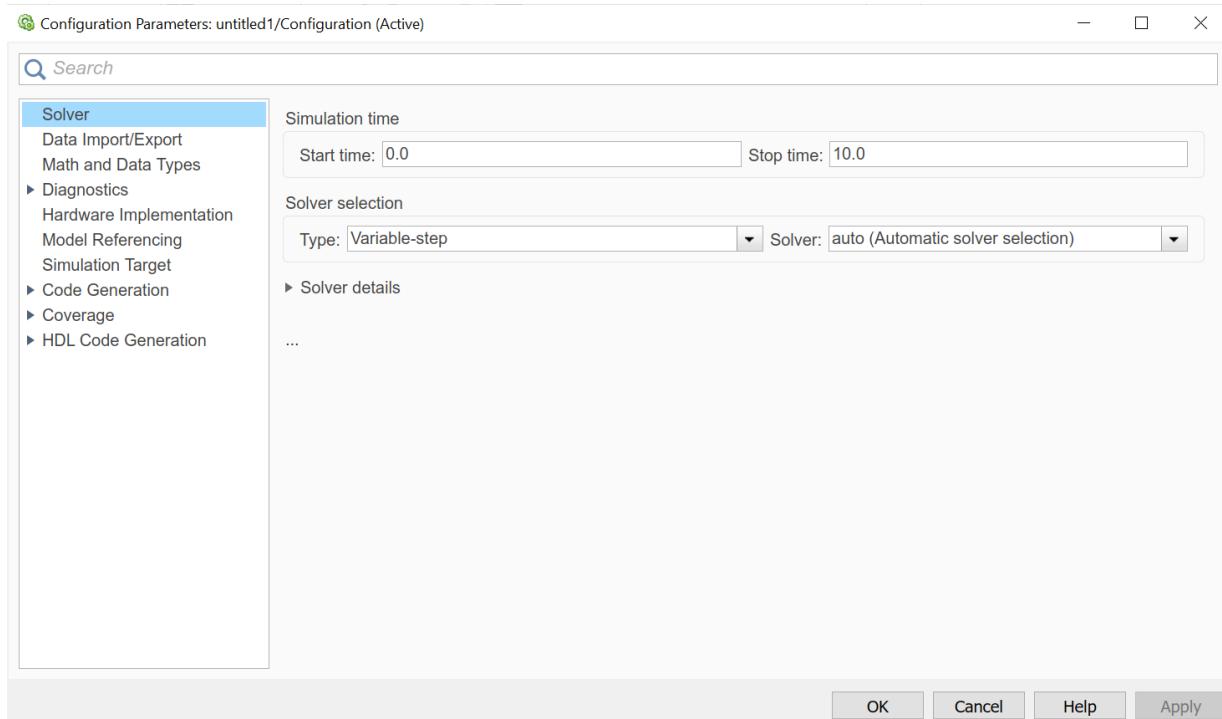
Ví dụ: hiệu chỉnh thông số của khối Sine wave.



Hình 1-11: Hiệu chỉnh thông số của khối Sine wave

Trước khi mô phỏng mô hình Simulink, chúng ta cần đặt các thông số mô phỏng bằng cách chọn menu **Simulation → Configuration Parameters**.

Ở cửa sổ **Configuration Parameters**, chúng ta có thể đặt một số thông số như **Start time**, **Stop time** (second – giây), các bước thời gian mô phỏng,... sau đó nhấn nút **OK**.



Hình 1-12: Các thông số mô phỏng của Simulink

1.3 Thực hành

■ Phần thực hành tại lớp

- 1) Sử dụng lệnh **help** hoặc **lookfor** để tìm kiếm thông tin cho các câu hỏi sau:
 - Hãy tìm lệnh thể hiện phép toán hàm cosin
 - Tìm thông tin về hàm logarithms
- 2) Cho hai số phức bất kỳ, ví dụ $-3 + 2i$ và $5 - 7i$. Hãy thực hiện các phép toán để cộng, trừ, nhân và chia hai số phức này với nhau sao cho đơn giản, nhanh nhất. (gợi ý khai báo biến).
- 3) Hãy thực hiện các bài tập sau đây:
 - Tạo một vector bao gồm những số lẻ trong khoảng từ 21 đến 47.
 - Cho $x = [4 \ 5 \ 9 \ 6]$
 - o Trừ đi 3 ở mỗi thành phần của vector
 - o Cộng 11 vào các thành phần có vị trí lẻ
- 4) Vẽ đồ thị của hàm $y = \sin(x)$ trong khoảng $0 < x < 30$ thêm tiêu đề và mô tả của các trục vào đồ thị.

■ Phần thực hành về nhà

- 1) Cho ma trận $A = [2 \ 4 \ 1; 6 \ 7 \ 2; 3 \ 5 \ 9]$. Thực thi các phép toán sau đối với A:
 - Gán hàng thứ 1 của A cho vector x
 - Gán 2 hàng cuối của A cho y
- 2) Hãy xóa tất cả các biến (lệnh **clear**). Định nghĩa ma trận $A = [1:4; 5:8; 1 \ 1 \ 1 \ 1]$. Hãy thực thi và kiểm tra kết quả của các phép tính sau:
 - $x = A(:, 3)$
 - $y = A(3 : 3, 1 : 4)$
 - $B = A(1 : 3, 2 : 2)$
 - $A = [A; 2 \ 1 \ 7 \ 7; 7 \ 7 \ 4 \ 5]$
 - $A(1, 1) = 9 + A(2, 3)$
 - $C = A([1, 3], 2)$

- $A(2 : 3, 1 : 3) = [0 \ 0 \ 0; 0 \ 0 \ 0]$
- $D = A([2, 3, 5], [1, 3, 4])$
- $D(2, :) = []$

3) Vẽ một đường bằng nét gạch ngắn nối các điểm sau lại với nhau: (2, 6), (2.5, 18), (5, 17.5), (4.2, 12.5) và (2, 12).

4) Dùng Simulink vẽ tín hiệu sin như sau: (nhớ bỏ giới hạn bộ đếm data của scope)

- Vẽ 1 tín hiệu sin và biểu diễn bằng scope trong 5s, thay đổi số mẫu (sample) và nhận xét.
- Vẽ 3 tín hiệu sin khác nhau và biểu diễn trên cùng scope trong 5s. (gợi ý: thêm số lượng input trong setup scope).

BÀI 2. TRUYỀN TẢI THÔNG TIN

Sau khi học xong bài này, sinh viên có thể:

- Hiểu rõ sự khác nhau cơ bản giữa tín hiệu tuần tự và tín hiệu số.
- Biết được khái niệm cơ bản của phô, băng thông.
- Biểu diễn tín hiệu dưới miền tần số và miền thời gian.
- Có khả năng thiết kế tạo ra sóng vuông dựa trên việc tổng hợp nhiều sóng sin.
- Hiểu công thức suy hao của tín hiệu và minh họa được sự suy hao theo quãng đường.
- Phân biệt được các loại nhiễu của tín hiệu và ảnh hưởng của các loại nhiễu đến tín hiệu.
- Vận dụng Matlab để minh họa những tín hiệu tuần tự và tín hiệu số khác nhau.

2.1 Tín hiệu tuần tự, tín hiệu số, tín hiệu tuần hoàn

■ Tín hiệu tuần tự (Analog signal)

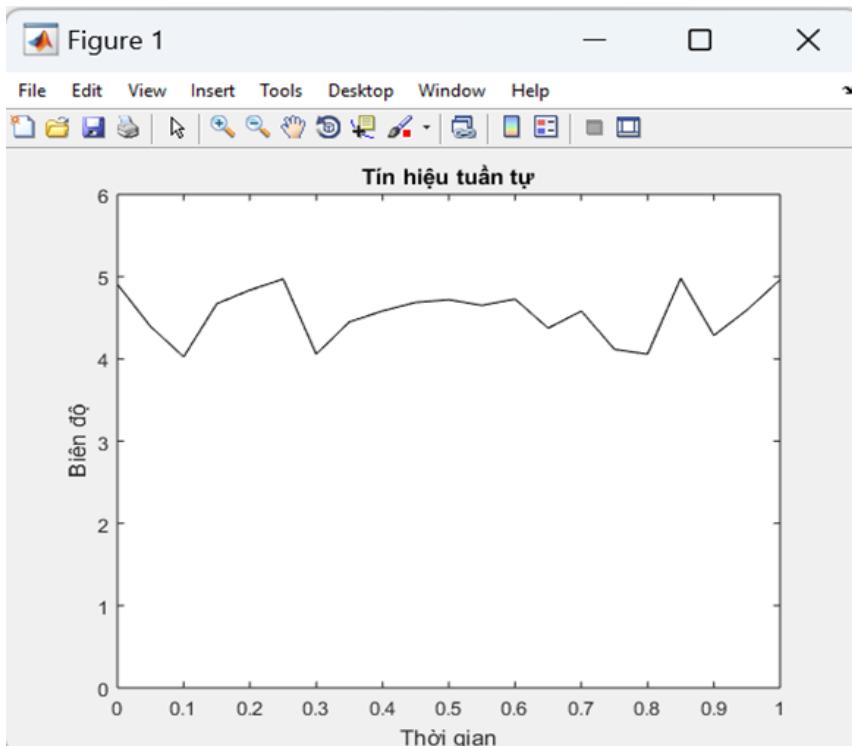
Không có thay đổi đột ngột, có thể biểu diễn một tín hiệu tuần tự bằng matlab như sau:

```
t=0:0.05:10;  
x=4+rand(1,201);  
plot(t,x);  
title('Tín hiệu tuần tự');
```

```
xlabel('Thời gian');
```

```
ylabel('Biên độ');
```

```
axis([0 1 0 6]);
```



Hình 2-1: Tín hiệu tuần tự

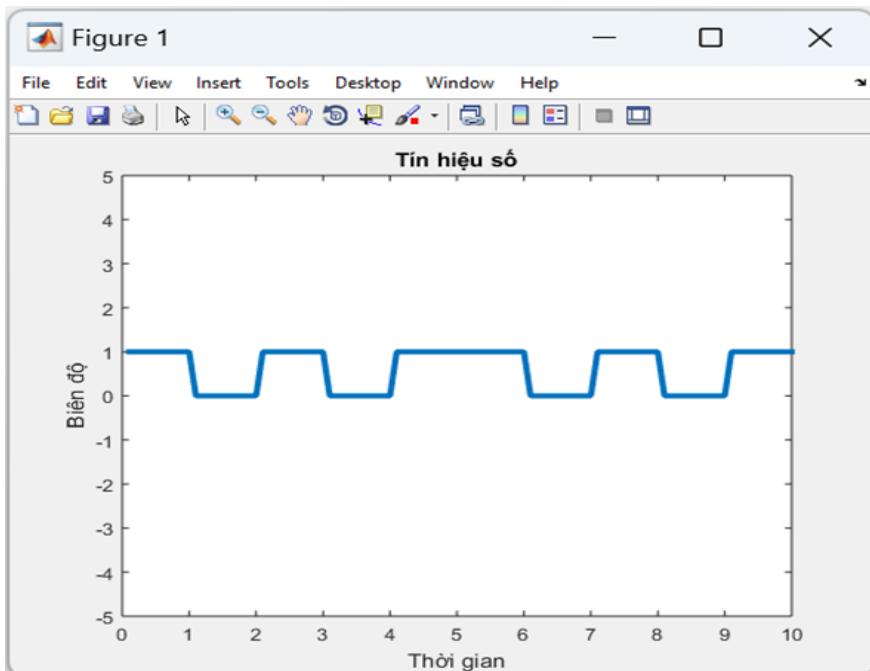
■ Tín hiệu số (Digital signal)

Là tín hiệu có thể thay đổi đột ngột từ một mức này sang một mức khác, có thể biểu diễn bằng matlab như sau:

```
t=1:10;
```

```
x=[1 0 1 0 1 1 0 1 0 1]
```

```
t=length(x);  
T=0.1:0.1:t;  
X=[x' x' x' x' x' x' x' x' x' x']  
X=X';  
X=X(:,:);  
X=X';  
plot(T,X);  
title('Tín hiệu số');  
xlabel('Thời gian');  
ylabel('Biên độ');  
axis([0 10 -5 5]);
```



Hình 2-2: Tín hiệu số

■ Tín hiệu tuần hoàn (Periodic signal)

Là tín hiệu có giá trị được lặp lại theo một khoảng thời gian cố định được gọi là chu kỳ của tín hiệu:

$$s(t + T) = s(t) \quad (T \text{ là chu kỳ của tín hiệu})$$

Một dạng sóng có chu kỳ như sau:

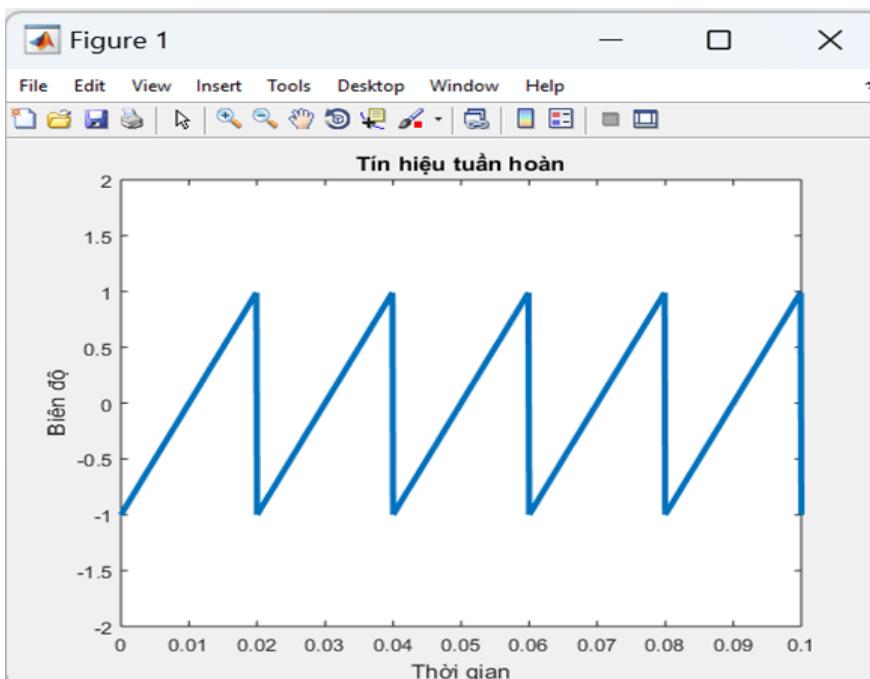
$$f=10000;$$

$$t=0:1/f:1.5;$$

$$x=sawtooth(2*pi*50*t);$$

$$plot(t,x);$$

```
title('Tín hiệu tuần hoàn');  
 xlabel('Thời gian');  
 ylabel('Biên độ');  
 axis([0 0.1 -2 2]);
```



Hình 2-3: Tín hiệu tuần hoàn

2.2 Tín hiệu tuần hoàn

Sóng Sin (Sine signal)

Sóng Sin là dạng tín hiệu tuần hoàn cơ bản nhất được thể hiện bằng các tham số sau:

- Biên độ (Peak Amplitude - A)
 - + Cao độ lớn mạnh nhất của tín hiệu
 - + Tính bằng volts
- Tần số (Frequency - f)
 - + Số lần lặp lại của tín hiệu trong một giây
 - + Tính bằng Hertz (Hz)
 - + Chu kỳ - T: $T=1/f$, là thời gian để tín hiệu lặp lại
- Pha (Phase - Φ): Vị trí tương đối về thời gian

Sóng sin có thể được biểu diễn bằng Matlab như sau:

% tín hiệu Sin có pha 0, $\pi/2$, π

$f=10000;$

$t=0:1/f:1.5;$

$\%x=sin(2*pi*50*t+0);$ % tín hiệu Sin có pha 0

$x=sin(2*pi*50*t+1*pi/4);$ % tín hiệu Sin có pha $\pi/4$

$\%y=sin(2*pi*50*t+2*pi/4);$ % tín hiệu Sin có pha

$\pi/2$

$y=sin(2*pi*50*t+3*pi/4);$ % tín hiệu Sin có pha $3\pi/4$

$\%z=sin(2*pi*50*t+4*pi/4);$ % tín hiệu Sin có pha π

$z=sin(2*pi*50*t+5*pi/4);$ % tín hiệu Sin có pha $5\pi/4$

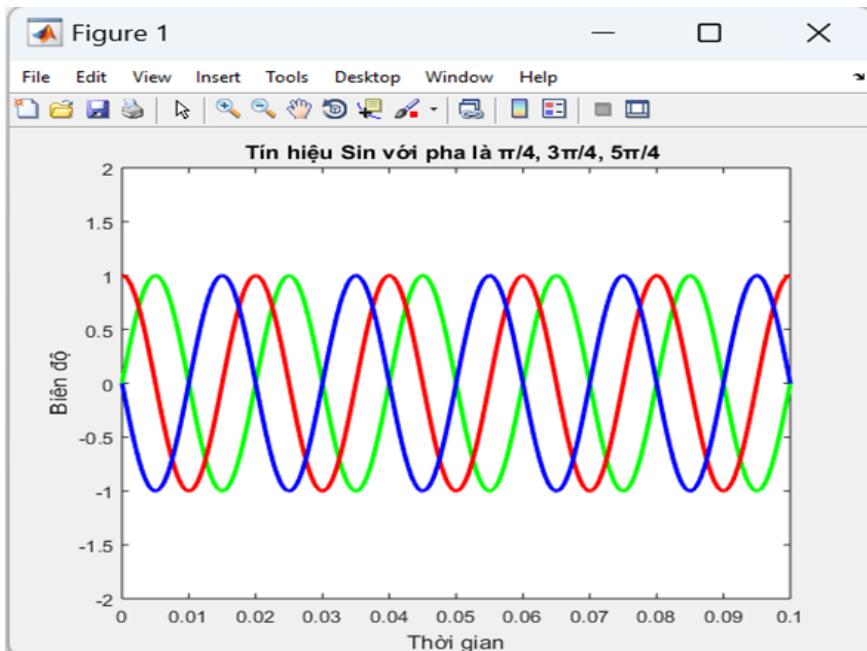
$plot(t,x,'g', t,y,'r', t,z,'b');$

$title('Tín hiệu Sin với pha là \pi/4, 3\pi/4, 5\pi/4');$

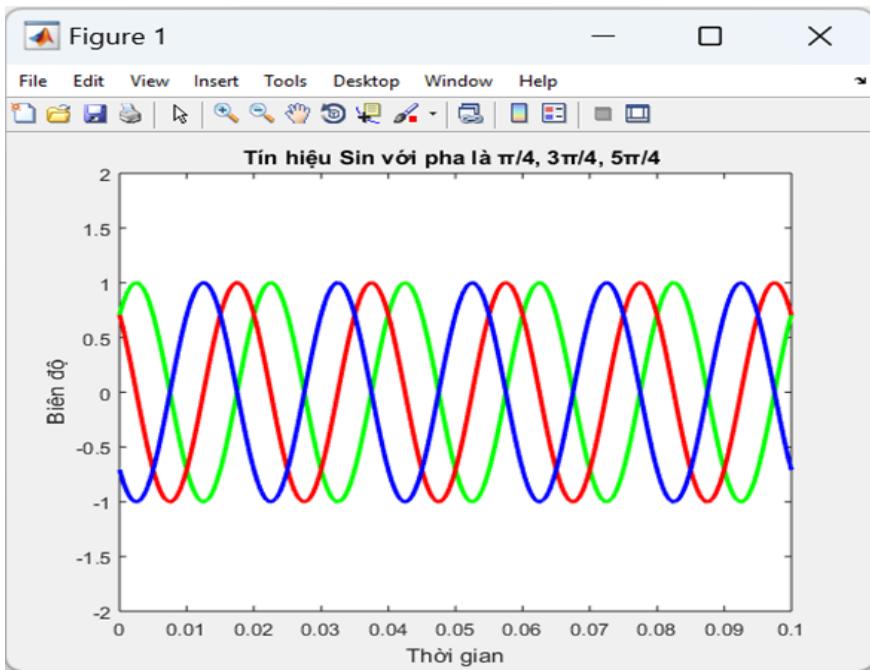
```
xlabel('Thời gian');
```

```
ylabel('Biên độ');
```

```
axis([0 0.1 -2 2]);
```



Hình 2-4: Tín hiệu Sin với pha là $0, \pi/2, \pi$



Hình 2-5: Tín hiệu Sin với pha là $\pi/4, 3\pi/4, 5\pi/4$

■ Sóng vuông (Sine signal)

Sóng vuông cũng là một dạng sóng đặc trưng của tín hiệu tuần hoàn được thể hiện bằng các tham số:

- Biên độ (Peak Amplitude - A)
- Tần số (hay chu kỳ)
- Duty cycle: thể hiện tỉ lệ % tín hiệu ở mức cao.

Sóng vuông có thể biểu diễn bằng Matlab như sau:

$$f=10000;$$

$$t=0:1/f:1.5;$$

$$x=1.5*square(2*pi*50*t,50);$$

```

y=square(2*pi*80*t,30);

plot(t,x,t,y);

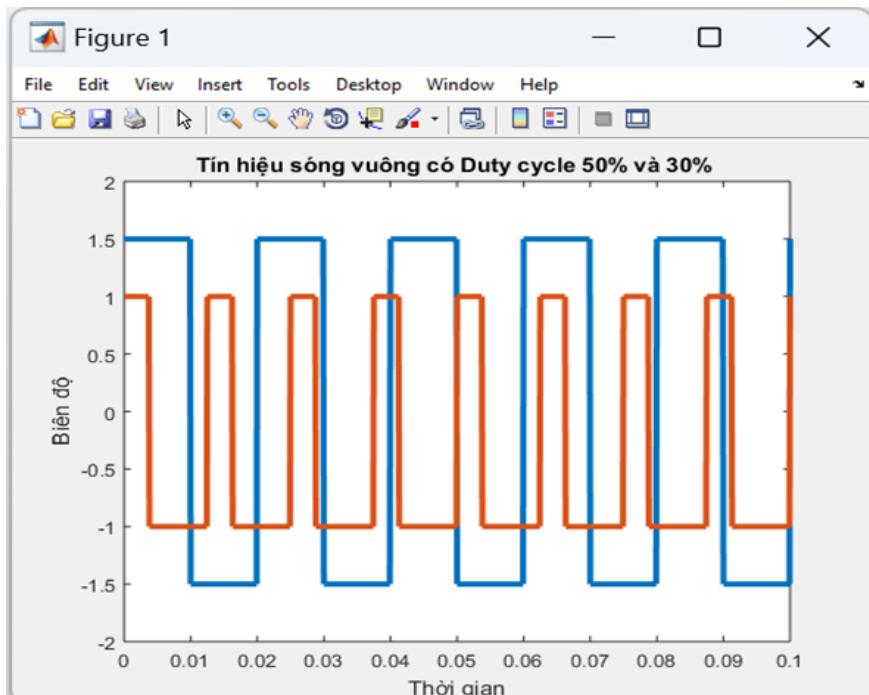
title('Tín hiệu sóng vuông có Duty cycle 50% và 30%');

xlabel('Thời gian');

ylabel('Biên độ');

axis([0 0.1 -2 2]);

```



Hình 2-6: Tín hiệu sóng vuông có Duty cycle 50% và 30%

2.3 Biểu diễn sóng trong miền thời gian và miền tần số

Tín hiệu có thể được biểu diễn trong miền thời gian như chúng ta thường thấy như các sóng Sin được biểu diễn trong phần 1.2. Trong một số hoàn cảnh cụ thể, chúng ta cần biểu diễn tín hiệu trong miền tần số để thuận tiện phân tích chúng.

Phương pháp phân tích: Cho một tín hiệu đi qua một hệ thống (tuyến tính bát biến), phô tần số của tín hiệu đầu ra sẽ bằng tích của phô tần số của tín hiệu đầu vào và *đáp ứng tần số* của hệ thống. điều này cũng giống như cho ánh sáng trắng đi qua lăng kính sẽ thu được các vạch phô tương ứng với các thành phần tần số của ánh sáng: đỏ, da cam, vàng...v.v.

Ta có thể biểu diễn tín hiệu trong miền tần số bằng hàm biến đổi Fast Fourier fft() trong Matlab như sau :

```
% Biểu diễn trong miền thời gian.
```

```
t = 0:1/50:5-1/50;
```

```
x = sin(2*pi*15*t) + sin(2*pi*20*t);
```

```
plot(t,x);
```

```
title('Tín hiệu Sin trong miền thời gian có tần số 15Hz  
và 20Hz');
```

```
xlabel('Thời gian');
```

```
ylabel('Biên độ');
```

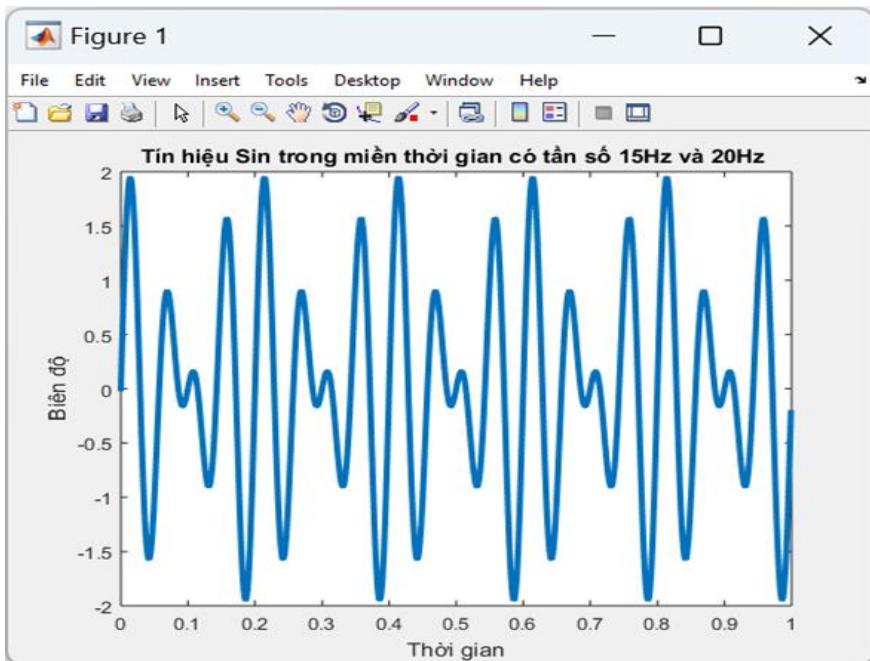
```
% Biểu diễn trong miền tần số
```

```

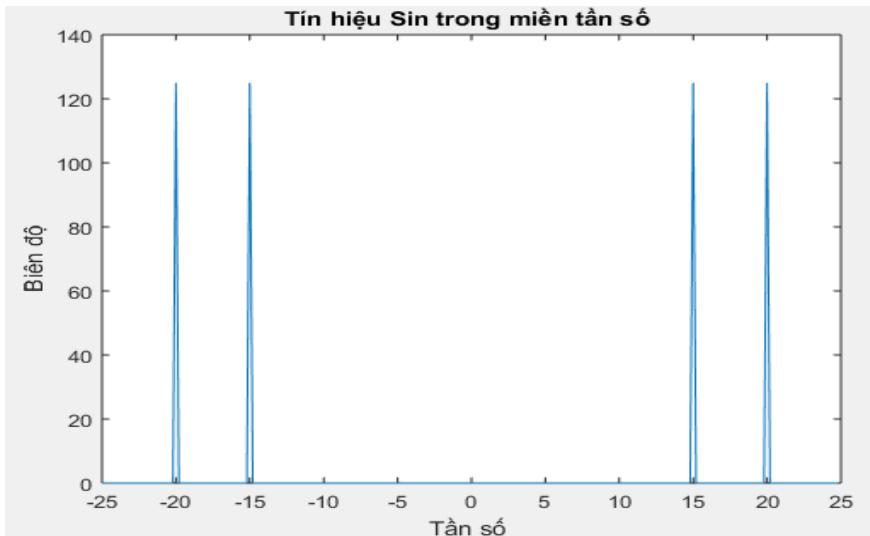
y = fft(x);
n = length(x);
fshift = (-n/2:n/2-1)*(50/n);
yshift = fftshift(y);
plot(fshift,abs(yshift));
title('Tín hiệu Sin trong miền tần số ');
xlabel('Tần số ');
ylabel('Biên độ');

```

Biến đổi Fast fourier cũng tạo ra một bản sao đối xứng của các tần số (tương ứng với tần số âm của tín hiệu). Để hình dung rõ hơn tính tuần hoàn này, chúng ta có thể sử dụng hàm fftshift, thực hiện dịch chuyển vòng tròn, tập trung vào số 0 trên đồ thị.



Hình 2-7: Tín hiệu Sin trong miền thời gian có tần số 15Hz
và 20Hz



Hình 2-8: Tín hiệu Sin trong miền tần số

2.4 Tổng hợp tín hiệu số từ tín hiệu tuần tự

Trong thực tế, các sóng vuông (square waves) có thể được tổng hợp từ các sóng sin theo công thức dưới đây:

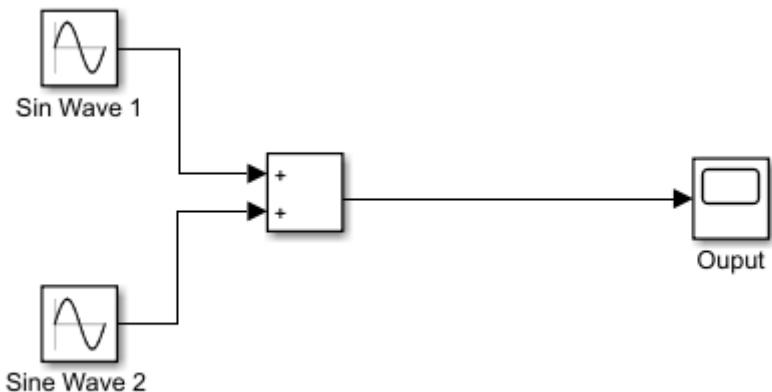
$$s(t) = A \times \frac{4}{\pi} \times \sum_{k \text{ odd}, k=1}^{\infty} \frac{\sin(2\pi ft)}{k} \quad (1)$$

Ví dụ:

Khi k bằng 3

$$s(t) = A \times \frac{4}{\pi} \times (\sin 2\pi ft + \frac{\sin 2\pi 3ft}{3})$$

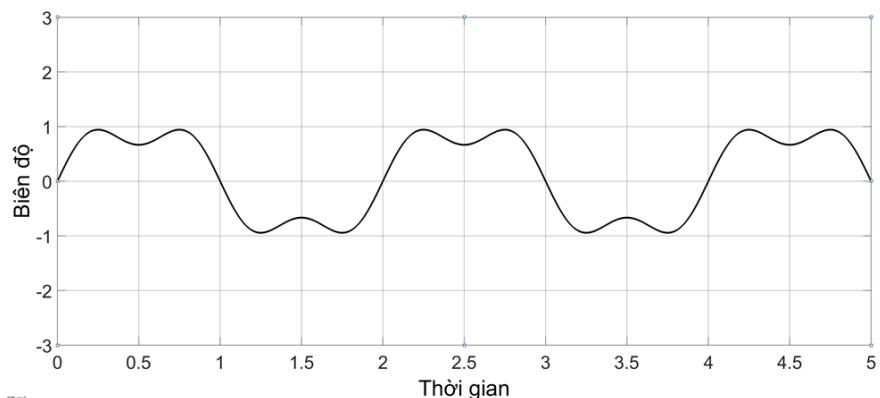
Thiết kế sóng trên sử dụng simulink của Matlab theo mô hình dưới:



Hình 2-9: Mô hình kết hợp hai sóng sin khi $k = 3$

Trong đó, Sin Wave 1 có dạng $\sin 2\pi ft$ và Sin Wave 2 có dạng $\frac{1}{3} \times \sin 2\pi 3ft$.

Lúc này sóng $s(t)$ sẽ có dạng như sau:

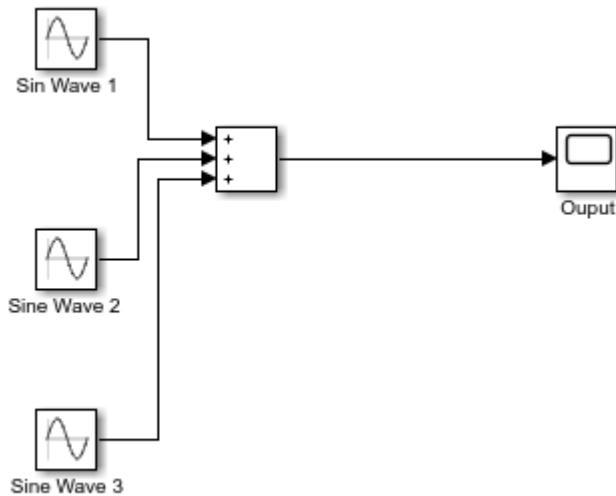


Hình 2-10: Tín hiệu $s(t)$ đầu ra khi $k = 3$

Khi k bằng 5

$$s(t) = A \times \frac{4}{\pi} \times (\sin 2\pi ft + \frac{\sin 2\pi 3ft}{3} + \frac{\sin 2\pi 5ft}{5})$$

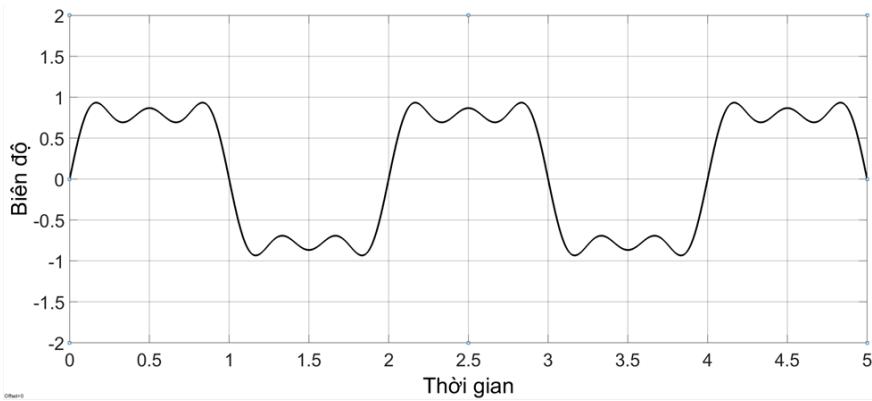
Thiết kế sóng trên sử dụng simulink theo mô hình bên dưới:



Hình 2-11: Mô hình kết hợp ba sóng sin khi k = 5

Trong đó, Sin Wave 1 có dạng $\sin 2\pi ft$, Sin Wave 2 có dạng $\frac{1}{3} \times \sin 2\pi 3ft$ và Sin Wave 3 có dạng $\frac{1}{5} \times \sin 2\pi 5ft$.

Lúc này sóng $s(t)$ sẽ có dạng như sau:

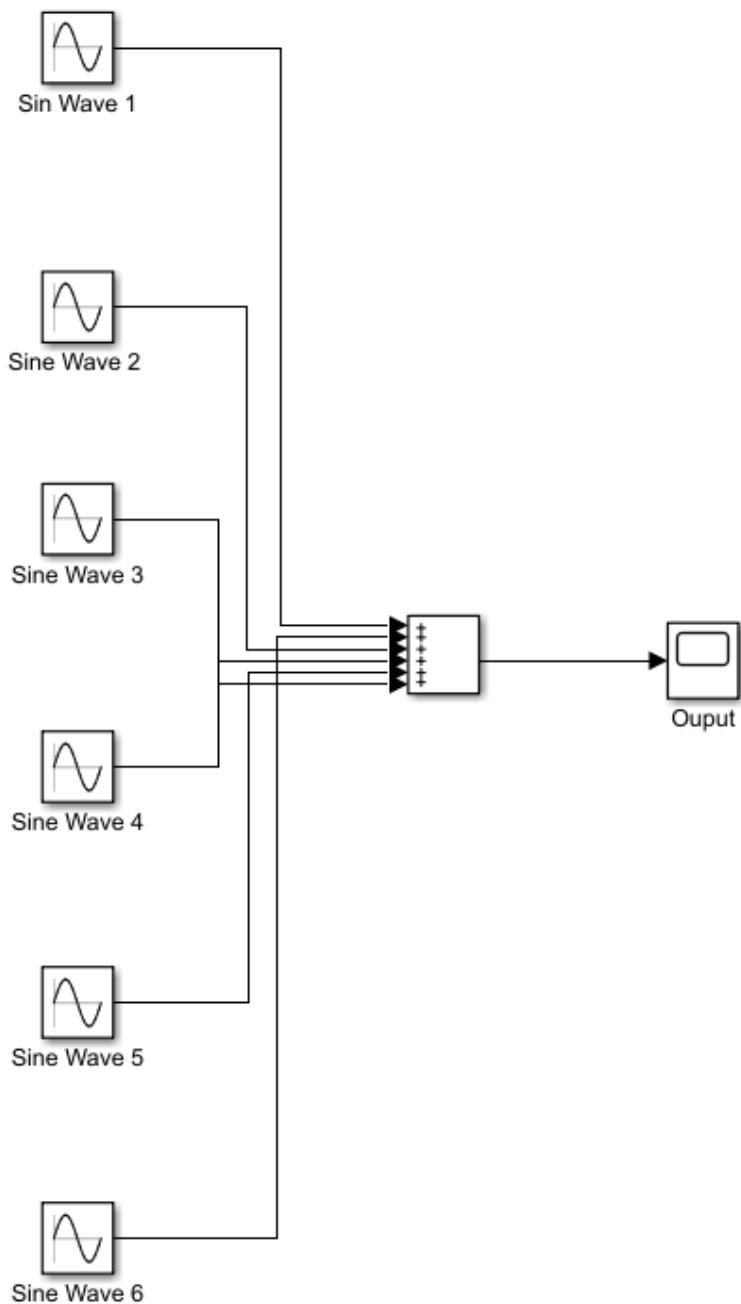


Hình 2-12: Tín hiệu $s(t)$ đầu ra khi $k = 5$

Khi k bằng 11

$$s(t) = A \times \frac{4}{\pi} \times \left(\sin 2\pi ft + \frac{\sin 2\pi 3ft}{3} + \frac{\sin 2\pi 5ft}{5} + \frac{\sin 2\pi 7ft}{7} + \frac{\sin 2\pi 9ft}{9} + \frac{\sin 2\pi 11ft}{11} \right)$$

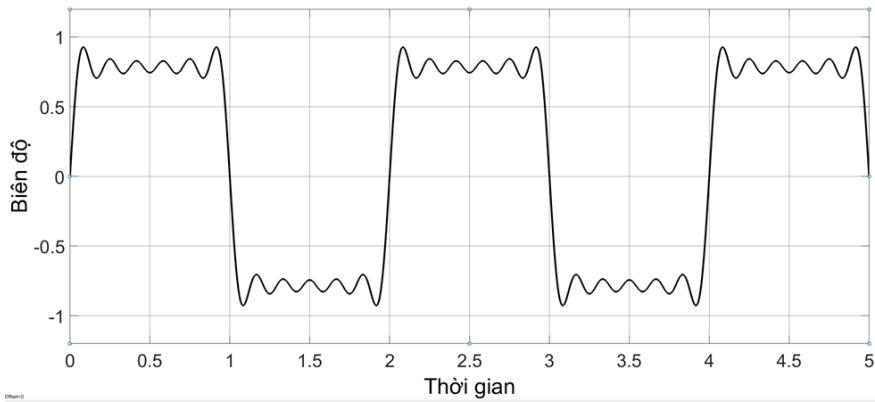
Lúc này, mô hình Simulink mô phỏng $s(t)$ sẽ có dạng như hình bên dưới:



Hình 2-13: Mô hình kết hợp 6 sóng khi $k = 11$

Trong đó, Sin Wave 1 có dạng $\sin 2\pi ft$, Sin Wave 2 có dạng $\frac{1}{3} \times \sin 2\pi 3ft$, Sin Wave 3 có dạng $\frac{1}{5} \times \sin 2\pi 5ft$, Sin Wave 4 có dạng $\frac{1}{7} \times \sin 2\pi 7ft$, Sin Wave 5 có dạng $\frac{1}{9} \times \sin 2\pi 9ft$ và Sin Wave 3 có dạng $\frac{1}{11} \times \sin 2\pi 11ft$.

Lúc này sóng $s(t)$ sẽ có dạng như sau:



Hình 2-14: Tín hiệu $s(t)$ đầu ra khi $k = 11$

Qua 3 mô hình trên chúng ta có thể rút ra một số kết luận như sau:

- Sóng vuông tạo ra sẽ có tần số bằng với tần số của Sine Wave 1.

- Khi k càng tăng thì năng lượng của sóng thứ k càng giảm do $A = \frac{1}{k}$. Do vậy, năng lượng của sóng $s(t)$ được tạo ra phụ thuộc vào các sóng có k nhỏ (ứng với A lớn).

2.5 Sự suy hao trong không gian

Cường độ của tín hiệu thường bị suy giảm với các mức độ khác nhau tùy thuộc vào khoảng cách của trạm phát và trạm thu. Sự suy hao này có thể được tính với công thức sau:

$$FSL = \left(\frac{4\pi d}{\lambda}\right)^2 \quad (2)$$

Trong đó,

FSL (Free Space Loss): suy hao truyền dẫn trong không gian tự do.

λ : bước sóng của tín hiệu (m).

d : khoảng cách giữa trạm phát và trạm thu (m).

Ngoài ra ta có $\lambda = \frac{c}{f}$. Do vậy,

$$FSL = \left(\frac{4\pi df}{c}\right)^2 \quad (3)$$

Bên cạnh đó để thể hiện sự suy hao người ta thường dùng đơn vị dB. Công thức chuyển đổi độ suy hao sang đơn vị này như sau:

$$FSL_{dB} = 10 \log_{10} FSL \quad (4)$$

Chúng ta sẽ sử dụng mã nguồn Matlab dưới đây để minh họa và tính toán sự suy hao.

```
% tần số (Đơn vị MHz)
```

```
f = 2400;
```

```
% vận tốc ánh sáng (đơn vị  $10^6$  m/s)
```

```
c = 299.792458;
```

```
% Ma trận khoảng cách giữa hai trạm (Đơn vị: mét)
```

```
d = 1:1:1000;
```

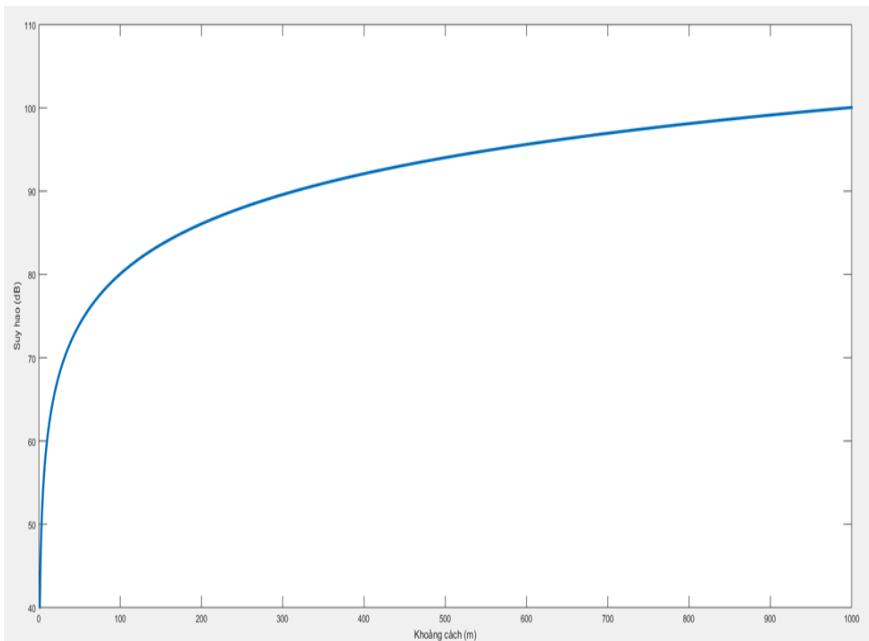
```
fspl =((4*pi*d*f)/c).^2;
```

```
plot(d,10*log10(fspl));
```

```
xlabel('Khoảng cách (m)');
```

```
ylabel('Suy hao (dB)');
```

Lúc này, biểu đồ thể hiện sự suy hao sẽ được thể hiện theo hình dưới:



Hình 2-15: Biểu đồ thể hiện sự suy hao trong khoảng cách 1000m ứng với tần số tín hiệu là 2400 Hz

Sự suy giảm của tín hiệu do suy hao truyền dẫn trong không gian tự do là hàm phi tuyến và tỉ lệ thuận với thời gian.

2.6 Nhiễu trong truyền tải thông tin

Trong quá trình truyền thông, ngoài việc bị suy giảm do quãng đường giữa trạm phát và trạm thu, tín hiệu truyền còn bị thay đổi bởi nhiễu. Có bốn loại nhiễu phổ biến:

- Nhiễu nhiệt: do dao động nhiệt của các điện tử trong chất dẫn và không thể bị loại bỏ.
- Nhiễu điều chế: phát sinh do việc nhiều tín hiệu chia sẻ một đường truyền chung. Tín hiệu nhiễu có tần số sẽ là tổng hoặc hiệu của các tín hiệu dùng chung môi trường truyền dẫn.
- Nhiễu xuyên kẽm: tín hiệu từ đường truyền này ảnh hưởng sang tín hiệu ở đường truyền khác. Ví dụ như một người nghe điện thoại bỗng nhiên nghe được đoạn đàm thoại của một đường truyền khác.
- Nhiễu xung: là một loại nhiễu bất thường, không liên tục, có cường độ cao xảy ra trong thời gian ngắn có thể phát sinh bởi nhiều điện tử. Nhiễu xung không ảnh hưởng nhiều đối với tín hiệu tuần tự nhưng thường gây ra những sai sót nghiêm trọng trong truyền tải tín hiệu số.

Dưới đây là một đoạn mã Matlab minh họa một loại của nhiễu nhiệt (nhiễu Gaussian) đối với tín hiệu tuần tự.

```
t = 0:9999;
```

```
x = 5*sin(2*pi*100*t/10000); % Tín hiệu tuần hoàn ban đầu
```

```
subplot(2,1,1);
```

```
plot(t(1:500),x(1:500));
```

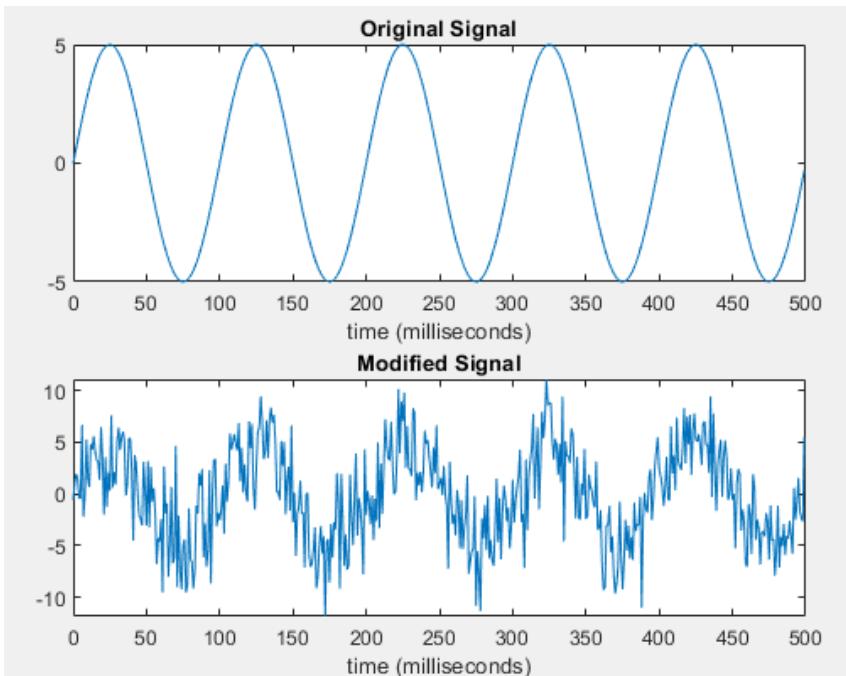
```
title('Original Signal');
```

```

xlabel('time (milliseconds)');
subplot(2,1,2);
y = awgn(x,1,'measured'); % tín hiệu sau khi thêm nhiễu
plot(t(1:500),y(1:500));
title('Modified Signal');
xlabel('time (milliseconds)');

```

Lúc này, biểu đồ minh họa tín hiệu ban đầu và tín hiệu sau khi nhiễu sẽ được thể hiện như hình dưới:



Hình 2-16: Sự ảnh hưởng của nhiễu Gaussian với tín hiệu tuần tự

2.7 Thực hành

■ Phân thực hành tại lớp

1) Thiết kế mô hình Simulink tạo nguồn tín hiệu Sin có tần số 50 Hz, pha ban đầu 90° biểu diễn hình ảnh bằng đồ thị (gợi ý dùng các khối Sine, scope trong thư viện Simulink).

2) Thiết kế mô hình Simulink tạo 3 nguồn tín hiệu Sin có tần số 50 Hz, pha ban đầu $0^\circ, 90^\circ, 180^\circ$ và biểu diễn hình ảnh trên cùng một đồ thị.

3) Lần lượt thay đổi giá trị của các tham số f, d để thấy được mối quan hệ của độ suy hao với các đại lượng này.

4) Tìm hiểu độ suy hao FSL khi có thêm đại lượng độ lợi của anten phát và anten thu. Sử dụng matlab để minh họa công thức như ở mục 2.5.

■ Phân thực hành về nhà

1) Thiết kế mô hình Simulink phát chuỗi bit nhị phân, biểu diễn chuỗi bit bằng đồ thị (gợi ý dùng các khối constant, switch, scope trong thư viện Simulink).

2) Thiết kế mô hình Simulink như 3 ví dụ trong phần 2.4.

3) Giải thích nguyên nhân vì sao nhiều xung thường gây ảnh hưởng lớn đối với tín hiệu số tuy nhiên lại không ảnh hưởng nhiều đến các tín hiệu tuần tự.

4) Cho sóng vuông có dạng như sau:

```
t=0:0.001:2;
```

```
x = square(5*pi*t);
```

Hãy minh họa nhiều Gauss đối với sóng này và so sánh với trường hợp trong ví dụ ở 2.6.

BÀI 3. CÁC KỸ THUẬT MÃ HÓA DỮ LIỆU SỐ

Sau khi học xong bài này, sinh viên có thể:

- Năm được các kỹ thuật mã hóa dữ liệu số sang tín hiệu số như: NRZ – L, NRZI, Bipolar – AMI, Pseudoternary, Manchester, Differential Manchester, B8ZS, HDB3.
- Năm được các kỹ thuật mã hóa dữ liệu số sang tín hiệu tuần tự như: ASK, FSK, PSK, QAM,...

3.1 Các kỹ thuật mã hóa dữ liệu số sang tín hiệu số

Có rất nhiều kỹ thuật mã hóa dữ liệu số sang tín hiệu số, trong đó, có một số kỹ thuật phổ biến [1] bao gồm:

- NRZ – L (Nonreturn to Zero-Level)
- NRZI (Nonreturn to Zero Inverted)
- Bipolar – AMI
- Pseudoternary
- Manchester
- Differential Manchester
- B8ZS
- HDB3

■ Kỹ thuật mã hóa NRZ – L

Ở kỹ thuật mã hóa này có hai mức điện áp khác nhau cho bit 1 và bit 0. Ví dụ điện áp mức cao được sử dụng cho bit 0 và

mức thấp được sử dụng cho bit 1. Khi không có điện áp của bit 0, sẽ chỉ có một mức điện áp không đổi cho các bit 1, điện áp sẽ thay đổi khi có sự thay đổi tín hiệu (từ 0 → 1 hoặc ngược lại).

Đoạn mã nguồn dưới đây minh họa cho kỹ thuật mã hóa NRZ – L (sử dụng bit 0 ở điện áp mức cao và bit 1 ở mức thấp).

Tập tin *nrz.m* chứa định nghĩa hàm Noreturn to Zero – Level.

```
function [t,x] = nrz(bits, bitrate)
```

```
T = length(bits)/bitrate;
```

```
n = 200;
```

```
N = n*length(bits);
```

```
dt = T/N;
```

```
t = 0:dt:T;
```

```
x = zeros(1,length(t));
```

```
for i = 0:length(bits)-1
```

```
    if bits(i+1) == 1
```

```
        x(i*n+1:(i+1)*n) = 0;
```

```
    else
```

```
        x(i*n+1:(i+1)*n) = 1;
```

```
    end
```

```
end
```

Tập tin *sample.m* sử dụng hàm được định nghĩa ở *nrz.m*.

```
bits = [1 0 1 0 0 1 0 1 1 0];
```

```
bitrate = 1;
```

```
figure;
```

```
[t,s] = nrz(bits,bitrate);
```

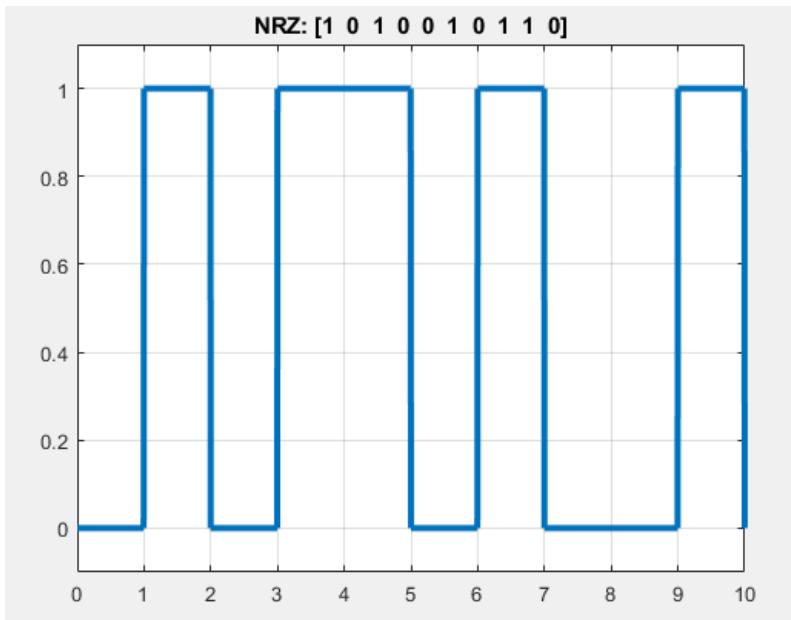
```
plot(t,s,'LineWidth',3);
```

```
axis([0 t(end) -0.1 1.1])
```

```
grid on;
```

```
title(['NRZ: [' num2str(bits) ']']);
```

Tín hiệu số đầu ra được mã hóa theo kỹ thuật NRZ – L
được thể hiện như ở hình dưới.



Hình 3-1: Tín hiệu số tương ứng với dữ liệu đầu vào theo kỹ thuật NRZ – L

■ Kỹ thuật mã hóa NRZI

Trong kỹ thuật mã hóa NRZI, bit 0 sẽ có mức điện áp giống với mức điện áp trước đó, bit 1 sẽ có mức điện áp ngược với mức điện áp trước đó.

Đoạn mã nguồn dưới đây minh họa cho kỹ thuật mã hóa NRZI.

Tập tin *nrzi.m* chứa định nghĩa hàm Noreturn to Zero Inverted.

```
function [t,x] = nrzi(bits, bitrate)
```

```
T = length(bits)/bitrate;
```

```

n = 200;

N = n*length(bits);

dt = T/N;

t = 0:dt:T;

x = zeros(1,length(t)); % tín hiệu đầu ra

last = 0;

for i = 0:length(bits)-1

    if i == 0

        if bits(i+1) == 0

            x(i*n+1:(i+1)*n) = 0;

            last = 0;

        else

            x(i*n+1:(i+1)*n) = 1;

            last = 1;

        end

    else

        if bits(i+1) == 0

            x(i*n+1:(i+1)*n) = last;

        else

            temp = mod(last+1,2);

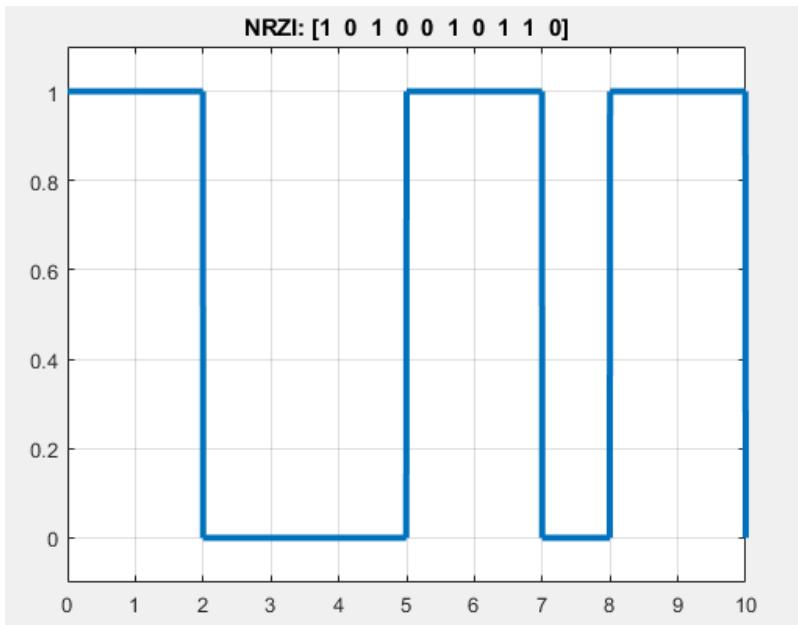

```

```
x(i*n+1:(i+1)*n) = temp;  
last = temp;  
end  
end
```

Tập tin *sample.m* sử dụng hàm được định nghĩa ở *nrz.m*.

```
bits = [1 0 1 0 0 1 0 1 1 0];  
bitrate = 1;  
figure;  
[t,s] = nrzi(bits,bitrate);  
plot(t,s,'LineWidth',3);  
axis([0 t(end) -0.1 1.1])  
grid on;  
title(['NRZI: [' num2str(bits) ']']);
```

Tín hiệu số đầu ra được mã hóa theo kỹ thuật NRZI được thể hiện như ở hình dưới.



Hình 3-2: Tín hiệu số tương ứng với dữ liệu đầu vào theo kỹ thuật NRZI

■ Kỹ thuật mã hóa Bipolar – AMI

Kỹ thuật mã hóa Bipolar – AMI sử dụng 3 mức điện áp trong đó bit 1 được biểu diễn bằng mức âm hoặc dương, bit 0 được biểu diễn bằng không có tín hiệu (điện áp 0). Trong đó, các bit 1 sẽ có điện áp âm dương xen kẽ lẫn nhau.

Đoạn mã nguồn dưới đây minh họa cho kỹ thuật mã hóa Bipolar – AMI.

Tập tin *AMI.m* chứa định nghĩa hàm AMI.

```
function [t,x] = AMI(bits, bitrate)
```

```
T = length(bits)/bitrate;
```

```

n = 200;
N = n*length(bits);
dt = T/N;
t = 0:dt:T;
x = zeros(1,length(t));
last = 1;
for i = 0:length(bits)-1
    if bits(i+1) == 1
        x(i*n+1:(i+1)*n) = last;
        last = last*-1;
    else
        x(i*n+1:(i+1)*n) = 0;
    end
end

```

Tập tin *sample.m* sử dụng hàm được định nghĩa ở *AMI.m*.

```

bits = [1 0 1 0 0 1 0 1 1 0];
bitrate = 1;
figure;
[t,s] = AMI(bits,bitrate);
plot(t,s,'LineWidth',3);

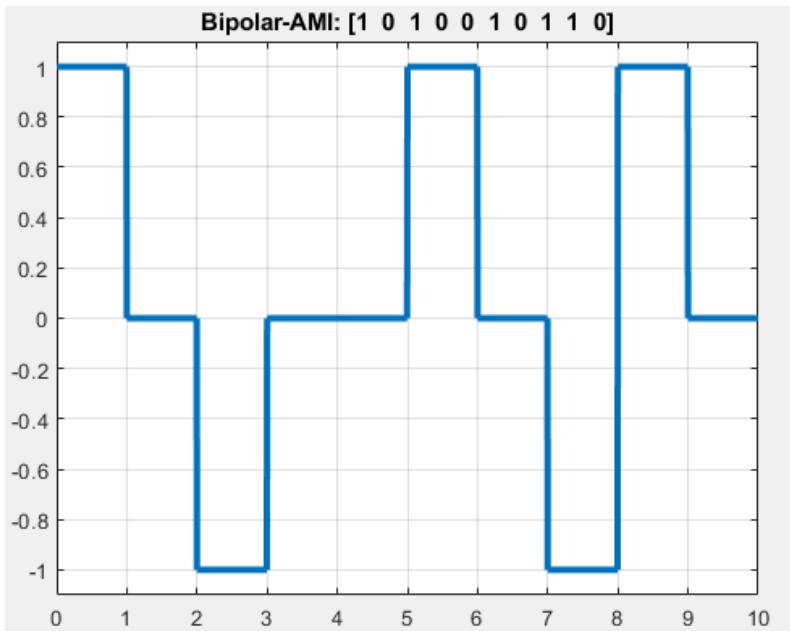
```

```

axis([0 t(end) -1.1 1.1])
grid on;
title(['Bipolar-AMI: [' num2str(bits) ']']);

```

Tín hiệu số đầu ra được mã hóa theo kỹ thuật Bipolar - AMI được thể hiện như ở hình dưới.



Hình 3-3: Tín hiệu số tương ứng với dữ liệu đầu vào theo kỹ thuật Bipolar – AMI

■ Kỹ thuật mã hóa Pseudoternary

Tương tự với kỹ thuật mã hóa Bipolar – AMI, kỹ thuật này cũng sử dụng 3 mức tín hiệu. Tuy nhiên, ở kỹ thuật này bit 0 sẽ được biểu diễn bằng mức điện áp âm hoặc dương và bit 1

sẽ được biểu diễn bằng không có tín hiệu (điện áp 0). Trong đó, các bit 0 sẽ có mức điện áp âm dương xen kẽ lẫn nhau.

Đoạn mã nguồn dưới đây minh họa cho kỹ thuật mã hóa Pseudoternary.

Tập tin *Pseudo.m* chứa định nghĩa hàm Pseudo.

```
function [t,x] = Pseudo(bits, bitrate)
```

```
n = 200;
```

```
N = n*length(bits);
```

```
dt = T/N;
```

```
t = 0:dt:T;
```

```
x = zeros(1,length(t));
```

```
last = 1;
```

```
for i = 0:length(bits)-1
```

```
    if bits(i+1) == 0
```

```
        x(i*n+1:(i+1)*n) = last;
```

```
        last = last*-1;
```

```
    else
```

```
        x(i*n+1:(i+1)*n) = 0;
```

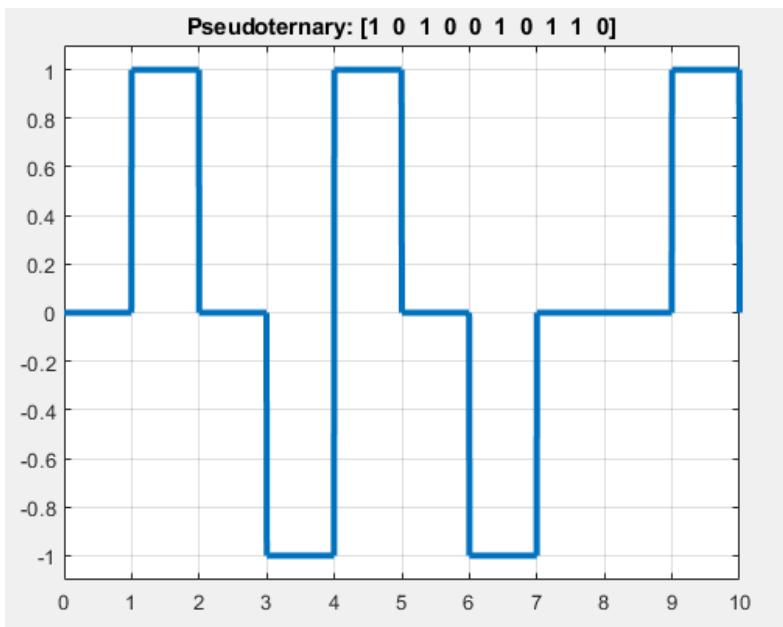
```
    end
```

```
end
```

Tập tin *sample.m* sử dụng hàm được định nghĩa ở *Pseudo.m*.

```
bits = [1 0 1 0 0 1 0 1 1 0];
bitrate = 1;
figure;
[t,s] = Pseudo(bits,bitrate);
plot(t,s,'LineWidth',3);
axis([0 t(end) -1.1 1.1])
grid on;
title(['Pseudoternary: [' num2str(bits) ']']);
```

Tín hiệu số đầu ra được mã hóa theo kỹ thuật Pseudoternary được thể hiện như ở hình dưới.



Hình 3-4: Tín hiệu số tương ứng với dữ liệu đầu vào theo thuật toán Pseudoternary

■ Kỹ thuật mã hóa Manchester

Trong kỹ thuật mã hóa Manchester, điện áp được thay đổi ở khoảng giữa của một bit. Trong đó, điện áp được thay đổi từ mức thấp lên cao đại diện cho bit 1 và điện áp thay đổi từ mức cao xuống thấp đại diện cho bit 0.

Đoạn mã nguồn dưới đây minh họa cho kỹ thuật mã hóa Manchester.

Tập tin *manchester.m* chứa định nghĩa hàm manchester.

```
function [t,x] = manchester(bits, bitrate)
```

```
T = length(bits)/bitrate;
```

```

n = 200;
N = n*length(bits);
dt = T/N;
t = 0:dt:T;
x = zeros(1,length(t));
for i = 0:length(bits)-1
    if bits(i+1) == 0
        x(i*n+1:(i+0.5)*n) = 1;
        x((i+0.5)*n+1:(i+1)*n) = -1;
    else
        x(i*n+1:(i+0.5)*n) = -1;
        x((i+0.5)*n+1:(i+1)*n) = 1;
    end
end

```

Tập tin *sample.m* sử dụng hàm được định nghĩa ở *manchester.m*.

```

bits = [1 0 1 0 0 1 0 1 1 0];
bitrate = 1;
figure;
[t,s] = manchester(bits,bitrate);
plot(t,s,'LineWidth',3);

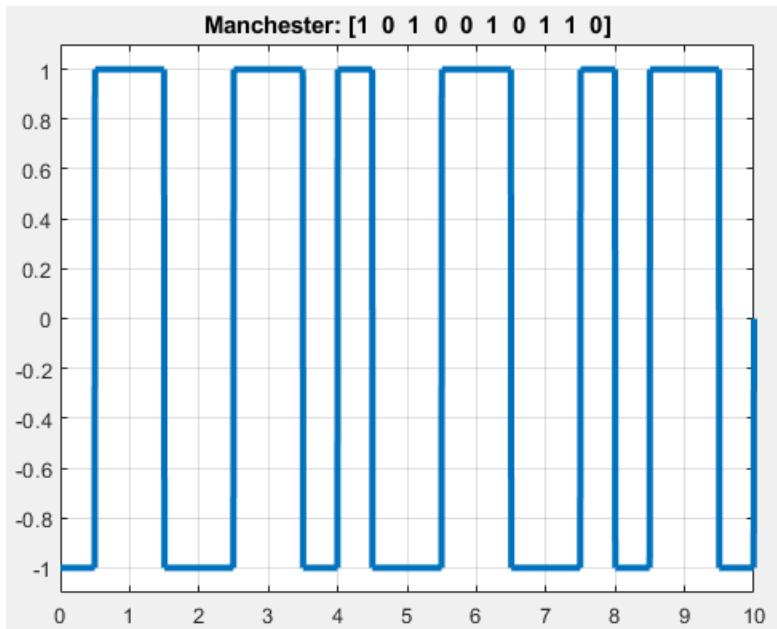
```

```

axis([0 t(end) -1.1 1.1])
grid on;
title(['Manchester: [' num2str(bits) ']']);

```

Tín hiệu số đầu ra được mã hóa theo kỹ thuật Manchester được thể hiện như ở hình dưới.



Hình 3-5: Tín hiệu số tương ứng với dữ liệu đầu vào theo
thuật toán Manchester

■ Kỹ thuật mã hóa Differential Manchester

Trong mã hóa Differential Manchester, bit 0 sẽ có mức thay đổi điện áp giống với mức thay đổi điện áp của bit trước đó và bit 1 sẽ có mức thay đổi điện áp ngược lại với mức thay đổi tín hiệu của bit trước đó.

Tập tin *DMan.m* chứa định nghĩa hàm DMan.

```
function [t,x] = DMan(bits, bitrate)
```

```
T = length(bits)/bitrate;
```

```
n = 200;
```

```
N = n*length(bits);
```

```
dt = T/N;
```

```
t = 0:dt:T;
```

```
x = zeros(1,length(t));
```

```
last = [0 0];
```

```
for i = 0:length(bits)-1
```

```
    if i==0
```

```
        if bits(i+1) == 0
```

```
            x(i*n+1:(i+0.5)*n) = 1;
```

```
            x((i+0.5)*n+1:(i+1)*n) = -1;
```

```
            last(1) = 1;
```

```
            last(2) = -1;
```

```
    else
```

```
        x(i*n+1:(i+0.5)*n) = -1;
```

```
        x((i+0.5)*n+1:(i+1)*n) = 1;
```

```
        last(1) = -1;
```

```

last(2) = 1;

end

else

if bits(i+1) == 0

x(i*n+1:(i+0.5)*n) = last(1);

x((i+0.5)*n+1:(i+1)*n) = last(2);

else

last(1) = last(1)^-1;

last(2) = last(2)^-1;

x(i*n+1:(i+0.5)*n) = last(1);

x((i+0.5)*n+1:(i+1)*n) = last(2);

end

end

end

```

Tập tin *sample.m* sử dụng hàm được định nghĩa ở *DMan.m*.

```

bits = [1 0 1 0 0 1 0 1 1 0];

bitrate = 1;

figure;

[t,s] = DMan(bits,bitrate);

plot(t,s,'LineWidth',3);

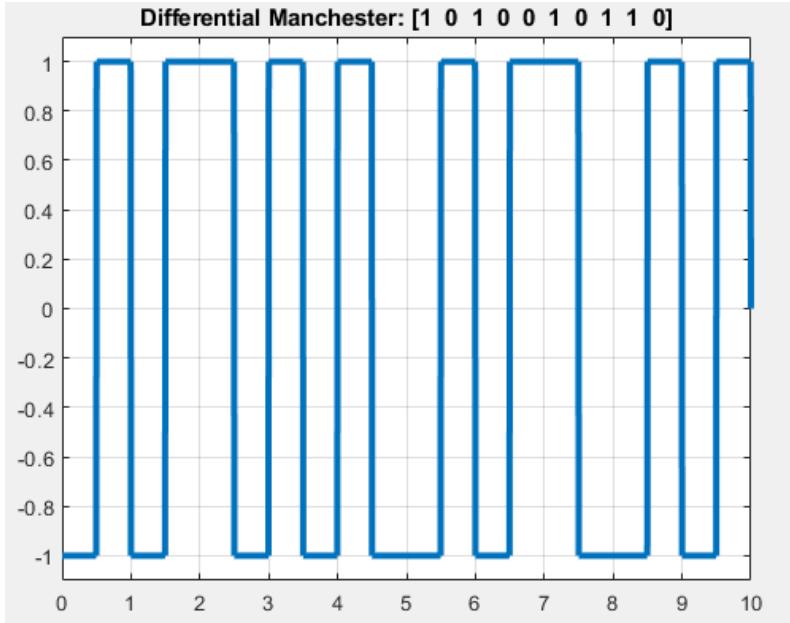
```

```

axis([0 t(end) -1.1 1.1])
grid on;
title(['Differential Manchester: [' num2str(bits) ']']);

```

Tín hiệu số đầu ra được mã hóa theo kỹ thuật Differential Manchester được thể hiện như ở hình dưới.



Hình 3-6: Tín hiệu số tương ứng với dữ liệu đầu vào theo
thuật toán Differential Manchester

■ Kỹ thuật mã hóa B8ZS

Kỹ thuật mã hóa B8ZS (Bipolar with 8-zeros substitution) là kỹ thuật dựa trên Bipolar – AMI. Kỹ thuật này sẽ tiến hành thay thế dãy 8 bit 0 liên tiếp nhau theo luật dưới đây:

- Nếu có 8 số 0 liên tiếp và xung điện áp cuối cùng trước đó là dương, 8 số 0 sẽ được mã thành 000+-0+-.
- Nếu có 8 số 0 liên tiếp và xung điện áp cuối cùng trước đó là âm, 8 số 0 sẽ được mã thành 000-+0+-.

Đoạn mã nguồn dưới đây minh họa cho kỹ thuật mã hóa B8ZS.

Nội dung tập tin *B8ZS.m*

```
bits = [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0  
1 0 1];
```

```
bitrate = 1;
```

```
n = 1000;
```

```
T = length(bits)/bitrate;
```

```
N = n*length(bits);
```

```
dt = T/N;
```

```
t = 0:dt:T;
```

```
x = zeros(1,length(t));
```

```
counter = 0;
```

```
lastbit = -1;
```

```
for i=1:length(bits)
```

```
    if bits(i)==0
```

```
        counter = counter + 1;
```

```
        if counter==8
```

```

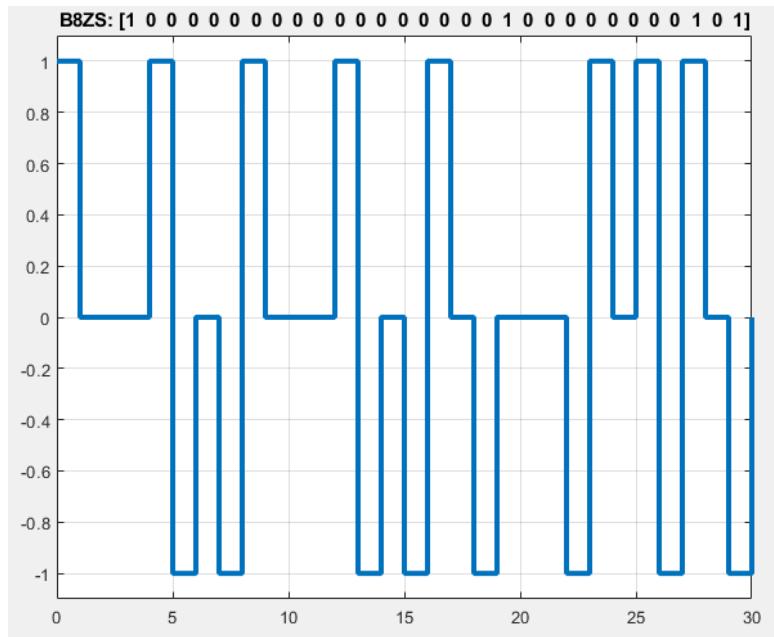
x((i-1-7)*n+1:(i-7)*n) = 0;
x((i-1-6)*n+1:(i-6)*n) = 0;
x((i-1-5)*n+1:(i-5)*n) = 0;
x((i-1-4)*n+1:(i-4)*n) = lastbit;
x((i-1-3)*n+1:(i-3)*n) = -lastbit;
lastbit = -lastbit;
x((i-1-2)*n+1:(i-2)*n) = 0;
x((i-1-1)*n+1:(i-1)*n) = lastbit;
x((i-1)*n+1:i*n) = -lastbit;
lastbit = -lastbit;
counter = 0;
end
else
    counter = 0;
    x((i-1)*n+1:i*n) = -lastbit;
    lastbit = -lastbit;
end
end
plot(t, x, 'Linewidth', 3);
axis([0 t(end) -1.1 1.1])

```

```
grid on;
```

```
title(['B8ZS: [' num2str(bits) ']']);
```

Tín hiệu số đầu ra được mã hóa theo kỹ thuật B8ZS được thể hiện như ở hình dưới.



Hình 3-7: Tín hiệu số tương ứng với dữ liệu đầu vào theo
thuật toán B8ZS

■ Kỹ thuật mã hóa HDB3

Kỹ thuật mã hóa HDB3 (high-density bipolar-3 zeros) là kỹ thuật dựa trên Bipolar – AMI. Kỹ thuật này sẽ tiến hành thay thế dãy 4 bit 0 liên tiếp nhau theo luật dưới đây:

- Nếu điện áp trước dây là dương
 - + Nếu có lẻ số lượng số 1 tính từ lần thay thế trước đó thì 4 số 0 sẽ được mã thành 000+.
 - + Nếu có chẵn số lượng số 1 tính từ lần thay thế trước đó thì 4 số 0 sẽ được mã thành -00-.
- Nếu điện áp trước dây là âm
 - + Nếu có lẻ số lượng số 1 tính từ lần thay thế trước đó thì 4 số 0 sẽ được mã thành 000-.
 - + Nếu có chẵn số lượng số 1 tính từ lần thay thế trước đó thì 4 số 0 sẽ được mã thành +00+.

Đoạn mã nguồn dưới đây minh họa cho kỹ thuật mã hóa HDB3.

Nội dung tập tin *HDB3.m*

```
bits = [1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 1];
```

```
bitrate = 1;
```

```
n = 1000;
```

```
T = length(bits)/bitrate;
```

```
N = n*length(bits);
```

```
dt = T/N;
```

```
t = 0:dt:T;
```

```
x = zeros(1,length(t));
```

```
counter = 0;
```

```

counter1 = 0;

lastbit = 1;

for i=1:length(bits)

    if bits(i)==0

        counter = counter + 1;

        if counter==4

            if mod(counter1,2) == 0

                x((i-1-3)*n+1:(i-3)*n) = lastbit;

                x((i-1-2)*n+1:(i-2)*n) = 0;

                x((i-1-1)*n+1:(i-1)*n) = 0;

                x((i-1)*n+1:i*n) = lastbit;

                lastbit = -lastbit;

            counter = 0;

            counter1 = 0;

        else

            x((i-1-3)*n+1:(i-3)*n) = 0;

            x((i-1-2)*n+1:(i-2)*n) = 0;

            x((i-1-1)*n+1:(i-1)*n) = 0;

            lastbit = -lastbit;

            x((i-1)*n+1:i*n) = lastbit;

```

```

lastbit = -lastbit;

counter = 0;

counter1 = 0;

end

end

else

counter = 0;

counter1 = counter1 + 1;

x((i-1)*n+1:i*n) = lastbit;

lastbit = -lastbit;

end

end

plot(t, x, 'Linewidth', 3);

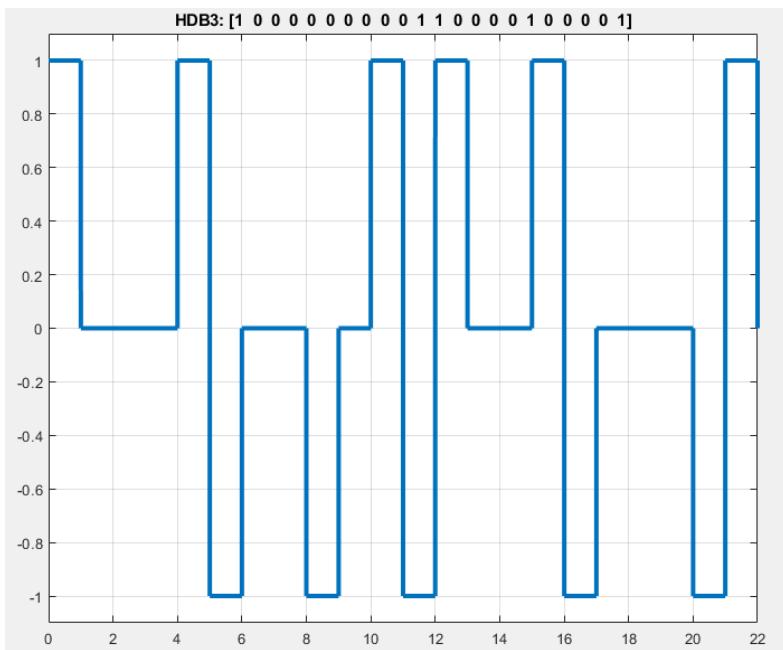
axis([0 t(end) -1.1 1.1])

grid on;

title(['HDB3: [' num2str(bits) ']']);


```

Tín hiệu số đầu ra được mã hóa theo kỹ thuật HDB3 được thể hiện như ở hình dưới.



Hình 3-8: Tín hiệu số tương ứng với dữ liệu đầu vào theo thuật toán HDB3

3.2 Các kỹ thuật mã hóa dữ liệu số sang tín hiệu tuần tự

Có rất nhiều kỹ thuật để mã hóa dữ liệu số sang tín hiệu tuần tự như điều chế khóa dịch biên độ ASK (Amplitude-Shift Keying), khóa dịch tần FSK (Frequency-Shift Keying), khóa dịch pha PSK (Phase-Shift Keying), điều chế kết hợp QAM (Quadrature Amplitude Modulation),... Các kỹ thuật này được phát triển theo sự tiến bộ của công nghệ chế tạo phần cứng.

■ Amplitude-shift Keying (ASK)

Là sóng mang có biên độ thay đổi theo dạng sóng tín hiệu điều chế, ở đây là biên độ thay đổi theo mức tín hiệu 0 và 1.

Biểu thức tín hiệu:

$$\text{Sóng mang: } S(t) = A_S \cdot \sin(\omega_0 t + j_0)$$

$$\text{Tín hiệu điều chế: } x = x(t)$$

$$\text{Tín hiệu ASK: } S_{ASK}(t) = m \cdot x(t) \cdot A_S \cdot \sin(\omega_0 t + j_0) \quad (1)$$

Với:

A_S : biên độ cực đại của tín hiệu ; j_0 : pha ban đầu của tín hiệu;

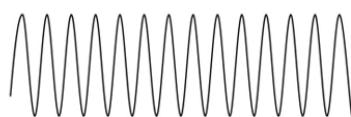
ω_0 :tần số góc của tín hiệu ($\omega_0 = 2\pi f_0$) ; m : hệ số điều chế.

Đơn giản hơn trong trường hợp một mức tín hiệu bằng 0, ta có:

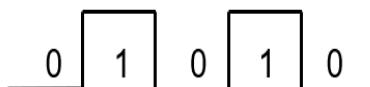
$$\text{Bit 1: } S_{ASK}(t) = A \cdot \sin(\omega_0 t)$$

$$\text{Bit 0: } S_{ASK}(t) = 0$$

Sóng mang



Tín hiệu điều chế



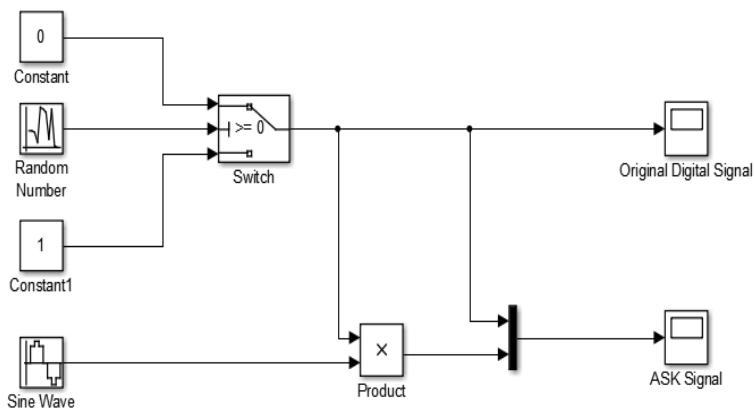
Tín hiệu ASK



Hình 3-9: Quá trình điều chế ASK

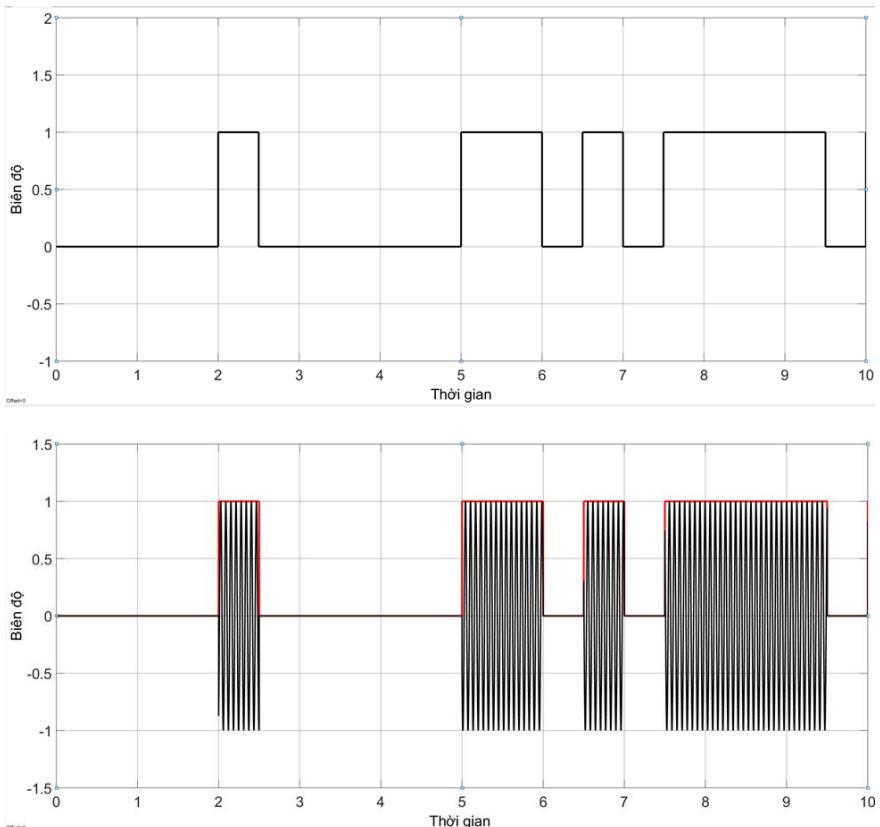
Mô hình điều chế ASK theo Simulink bao gồm:

- Các khối Constant 0, Constant 1, Random number, Switch dùng khởi tạo tín hiệu nguồn (chuỗi bit 0,1) cho quá trình điều chế.
- Khối Sine wave tạo sóng mang hình sin: $O(t) = A * \sin(F*t + Phase)$
 - + Biên độ sóng mang: $A = 1$
 - + Tân số góc sóng mang: $F = 100 \text{ rad/sec}$
 - + $Phase = 0$
- Original digital signal: Đồ thị (scope) hiển thị tín hiệu nguồn.
- ASK Signal: Đồ thị (scope) hiển thị tín hiệu sau điều chế ASK.
- Bộ Multiplex dùng để biểu diễn nhiều tín hiệu trên 1 scope.



Hình 3-10: Sơ đồ mô phỏng hệ thống điều chế ASK

Kết quả mô phỏng như sau:



Hình 3-11: Biểu diễn tín hiệu ASK

■ Frequency-Shift Keying (FSK)

Biểu thức tín hiệu:

$$\text{Sóng mang: } S(t) = A_S \cdot \sin(w_0 t + j_0)$$

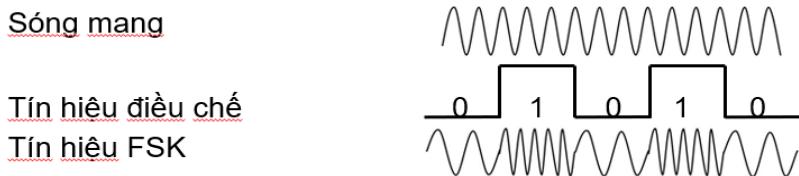
$$\text{Tín hiệu điều chế: } x = x(t)$$

$$\text{Tín hiệu FSK: } S_{FSK}(t) = A_S \cdot \sin(m \cdot x(t) \cdot w_0 t + j_0) \quad (2)$$

Với:

A_S : biên độ cực đại của tín hiệu ; j_0 : pha ban đầu của tín hiệu;

W_0 :tần số góc của tín hiệu ($W_0 = 2\pi f_0$) ; m : hệ số điều chế.

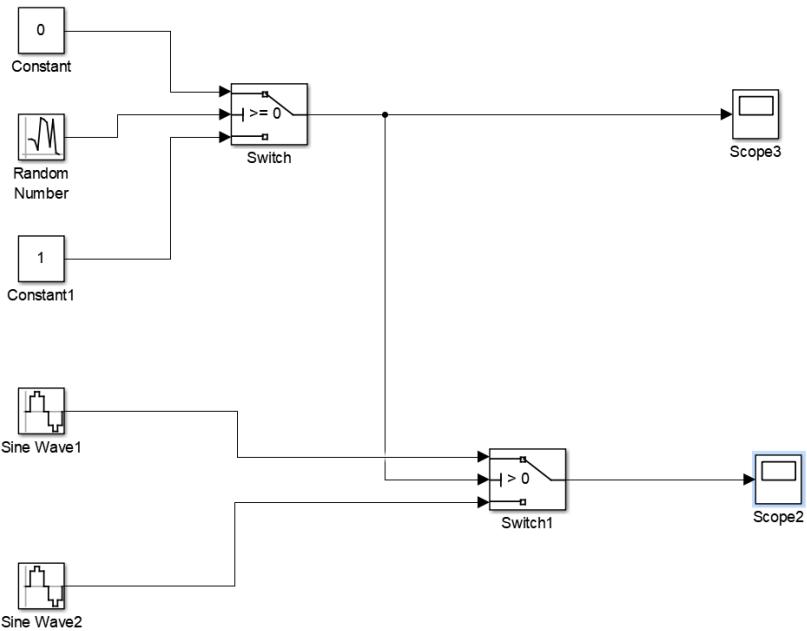


Hình 3-12: Quá trình điều chế FSK

Trong trường hợp FSK nhị phân BFSK (Binary FSK):

- Khi bit thông tin có giá trị logic 1 thì tần số là w_1 .
- Khi bit thông tin có giá trị logic 0 thì tần số là w_2 .
 - + Bit 1: $S_{FSK}(t) = A_S \cdot \sin(w_1 t)$
 - + Bit 0: $S_{FSK}(t) = A_S \cdot \sin(w_2 t)$

Thiết kế điều chế FSK sử dụng Simulink theo mô hình bên dưới:



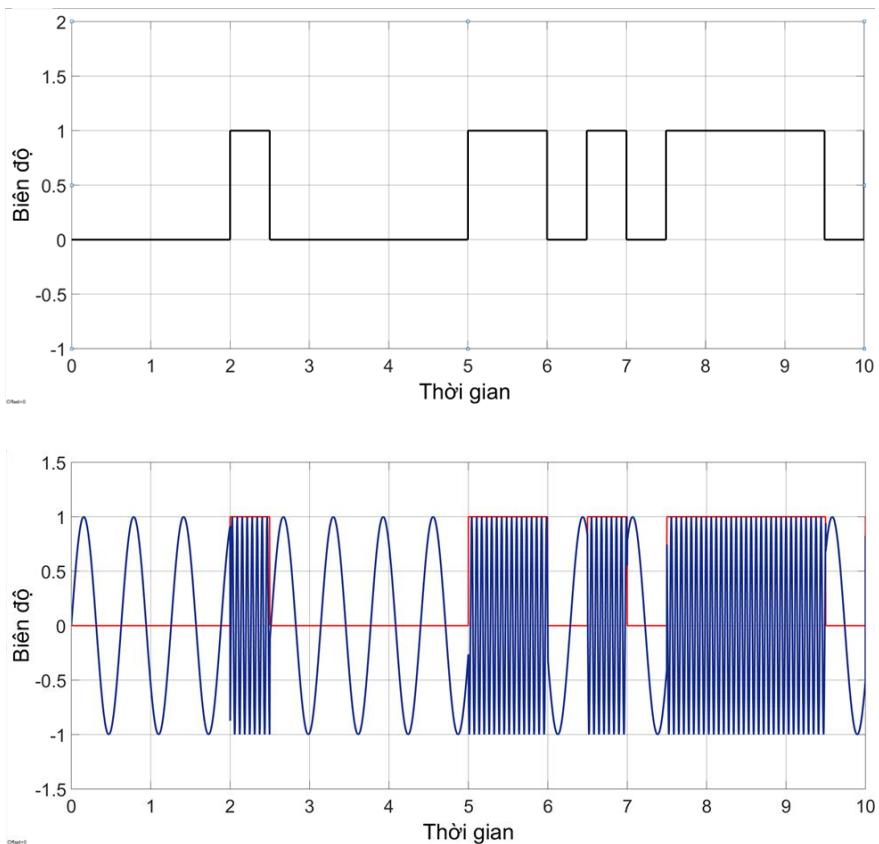
Hình 3-13: Sơ đồ mô phỏng hệ thống điều chế FSK

Trong đó:

- Các khối Constant 0, Constant 1, Random number, Switch dùng khởi tạo tín hiệu nguồn (chuỗi bit 0,1) cho quá trình điều chế.
- Khối Sine wave1 tạo sóng mang hình sin: $O(t) = A * \sin(F*t + Phase)$
 - + Biên độ sóng mang : $A = 1$
 - + Tần số góc sóng mang: $F = 100 \text{ rad/sec}$
 - + $Phase = 0$

- Khởi Sine wave2 tạo sóng mang hình sin: $O(t) = A * \sin(F*t + Phase)$
 - Biên độ sóng mang : $A = 1$
 - Tần số góc sóng mang: $F = 20 \text{ rad/sec}$
 - $Phase = 0$
- Scope2: hiển thị tín hiệu sau điều chế FSK.

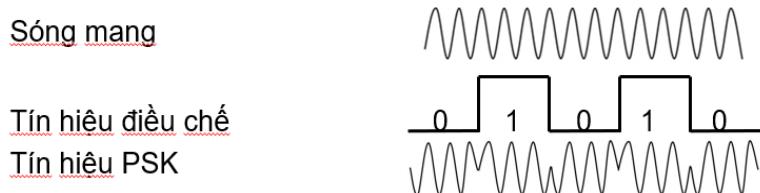
Kết quả mô phỏng như sau:



Hình 3-14: Biểu diễn tín hiệu FSK

■ Phase-Shift Keying (PSK)

Là sóng mang có pha biến đổi theo dạng của tín hiệu điều chế.



Hình 3-15: Quá trình điều chế PSK

Biểu thức tín hiệu:

$$\text{Sóng mang: } S(t) = A_s \cdot \sin(\omega_0 t + j_0)$$

$$\text{Tín hiệu điều chế: } x = x(t)$$

$$\text{Tín hiệu ASK: } S_{PSK}(t) = A_s \cdot \sin(\omega_0 t + m \cdot x(t) \cdot j_0) \quad (3)$$

Với:

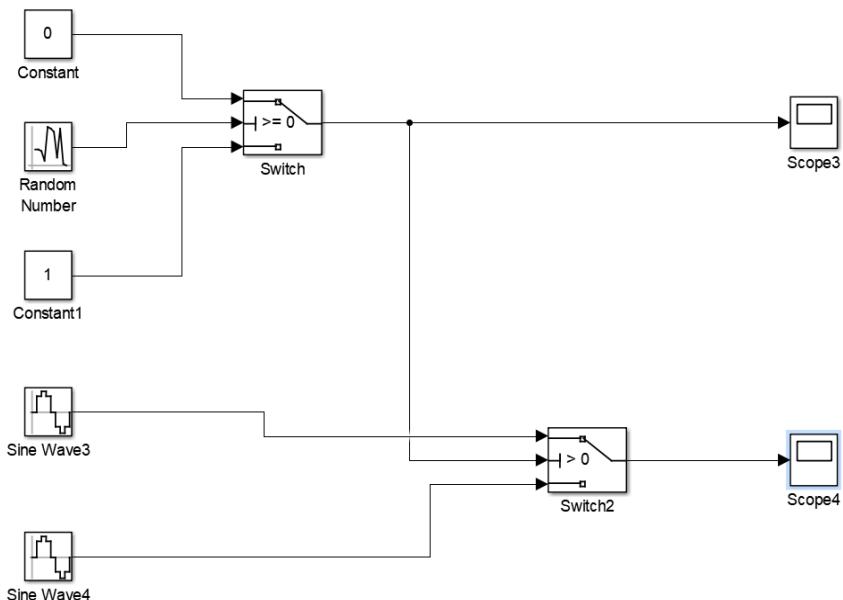
A_s : Biên độ cực đại của tín hiệu ; j_0 : Pha ban đầu của tín hiệu;

ω_0 :tần số góc của tín hiệu ($\omega_0 = 2\pi f_0$) ; m : hệ số điều chế.

Trong trường hợp PSK nhị phân - BFSK (Binary FSK):

- Khi bit thông tin có giá trị logic 1 thì tần số là j_1 .
- Khi bit thông tin có giá trị logic 0 thì tần số là j_2 .
 - + Bit 1: $S_{PSK}(t) = A_S \cdot \sin(\omega_0 t + j_1)$
 - + Bit 0: $S_{PSK}(t) = A_S \cdot \sin(\omega_0 t + j_2)$

Thiết kế điều chế PSK sử dụng Simulink theo mô hình bên dưới:

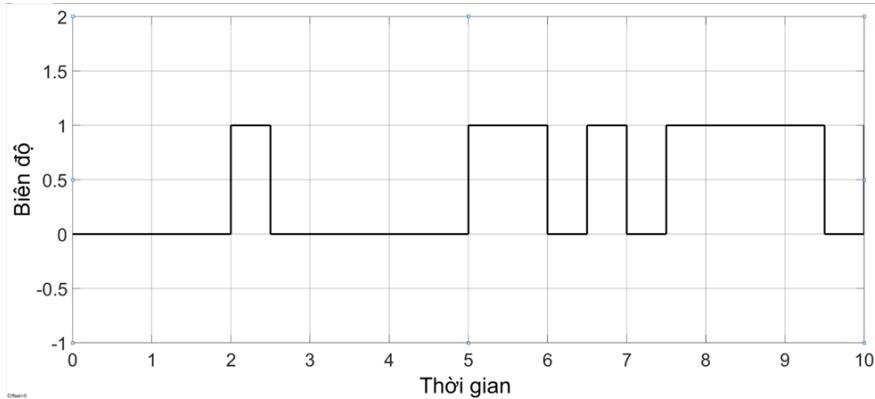


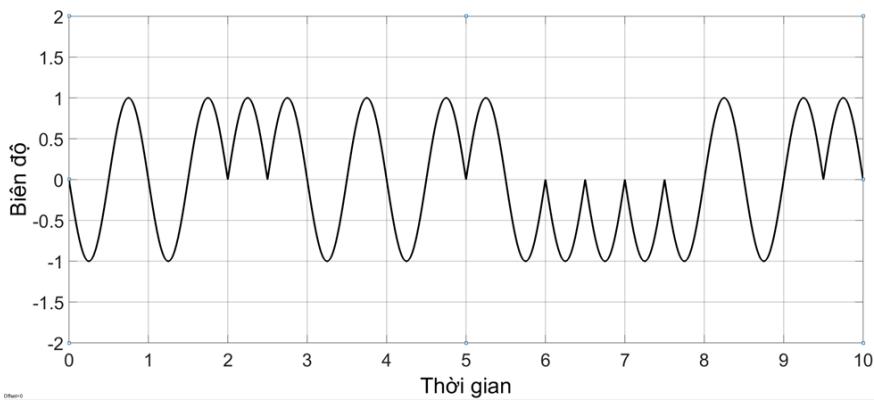
Hình 3-16: Sơ đồ mô phỏng hệ thống điều chế PSK

Trong đó:

- Các khối Constant 0, Constant 1, Random number, Switch dùng khởi tạo tín hiệu nguồn (chuỗi bit 0,1) cho quá trình điều chế.
- Khối Sine wave3 tạo sóng mang hình sin: $O(t) = A * \sin(F*t + Phase)$
 - + Biên độ sóng mang : $A = 1$
 - + Tần số góc sóng mang: $F = 2\pi \text{ rad/sec}$
 - + $Phase = 0$
- Khối Sine wave4 tạo sóng mang hình sin: $O(t) = A * \sin(F*t + Phase)$
 - + Biên độ sóng mang : $A = 1$
 - + Tần số góc sóng mang: $F = 2\pi \text{ rad/sec}$
 - + $Phase = \pi$
- Scope4: hiển thị tín hiệu sau điều chế PSK.

Kết quả mô phỏng như sau:

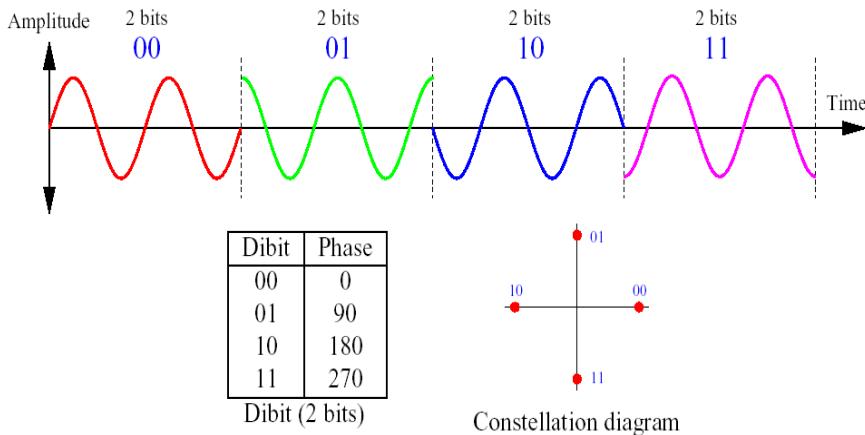




Hình 3-17: Biểu diễn tín hiệu PSK

■ Quadrature Phase-Shift Keying (QPSK)

Là sự phát triển trên cơ sở của PSK nhưng mỗi thành phần tín hiệu thể hiện 2 bit, còn được gọi là điều chế vuông pha. Tùy theo pha ban đầu của sóng mang mà ta có các pha của các tổ hợp Dibit khác nhau.



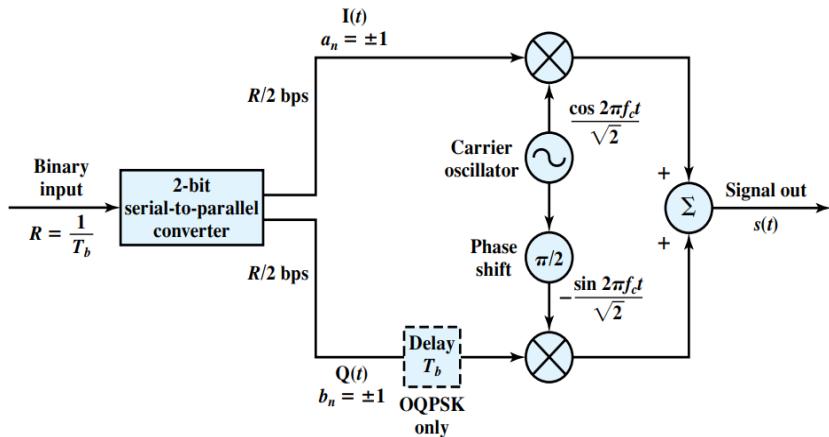
Hình 3-18: Biểu diễn dạng sóng và lược đồ chòm sao của tín hiệu QPSK

Các thành phần tín hiệu có thể biểu diễn như sau:

$$s(t) = \begin{cases} A \sin(2\pi f_c t) & 00 \\ A \sin(2\pi f_c t + \frac{\pi}{2}) & 01 \\ A \sin(2\pi f_c t + \pi) & 10 \\ A \sin(2\pi f_c t + \frac{3\pi}{2}) & 11 \end{cases}$$

Sơ đồ tạo tín hiệu QPSK dạng sin với bốn giá trị pha, xác định bởi tổ hợp (cặp) 2 bit liền nhau của chuỗi bit nhị phân. Tổ hợp 2 bit liền nhau này được gọi là Dibit có độ dài 2 khoảng bit. Chuỗi bit nhị phân trước khi đưa vào sơ đồ điều chế được tạo mã Dibit (nhờ trigger đếm đôi đơn giản) chuyển từ nối tiếp (serial) sang song song (Parallel). Mã Dibit được biểu thị bằng tín hiệu I và Q:

- Tín hiệu I (cùng pha – In Phase) ứng với giá trị bit đầu của cặp bit.
- Tín hiệu Q (Vuông pha – Quadrature) ứng với giá trị bit thứ hai của cặp bit.



Hình 3-19: Sơ đồ nguyên lý hệ thống điều chế QPSK

Bộ điều chế QPSK như vậy được xây dựng trên hai bộ BPSK, tạo ra hai tín hiệu BPSK_I và BPSK_Q cho bộ lấy tổng để hình thành tín hiệu 4 pha. Thiết kế điều chế PSK sử dụng sSmulink theo mô hình bên dưới. Trong đó:

- Các khối Switch-I dùng để khởi tạo tín hiệu nguồn đồng phase I cho quá trình điều chế.
- Các khối Switch-Q dùng để khởi tạo tín hiệu nguồn vuông phase Q cho quá trình điều chế.
- Khối Sine wave1 tạo sóng mang hình sin: $O(t) = A * \sin(F*t + Phase)$

Với biên độ sóng mang : $A = 1$, Tân số góc: $F = 2\pi rad/sec$, $Phase = 0$

- Khởi Sine wave2 tạo sóng mang hình sin: $O(t) = A * \sin(F*t + Phase)$

Với biên độ sóng mang : $A = 1$, Tần số góc: $F = 2\pi rad/sec$, $Phase = \pi$

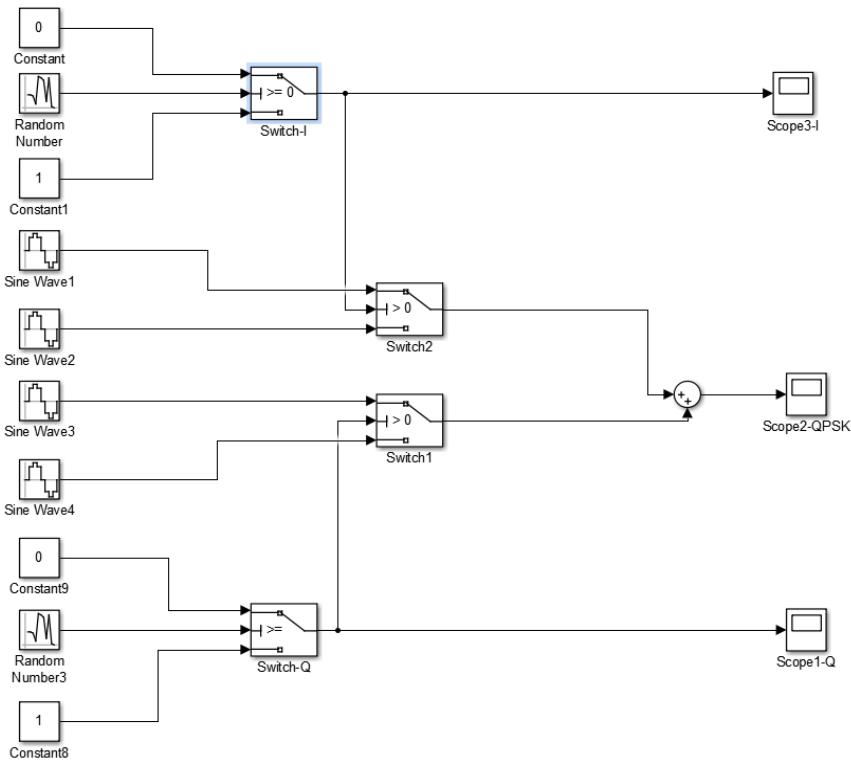
- Khởi Sine wave3 tạo sóng mang hình sin: $O(t) = A * \sin(F*t + Phase)$

Với biên độ sóng mang: $A = 1$, Tần số góc: $F = 2\pi rad/sec$, $Phase = \pi/2$

- Khởi Sine wave4 tạo sóng mang hình sin: $O(t) = A * \sin(F*t + Phase)$

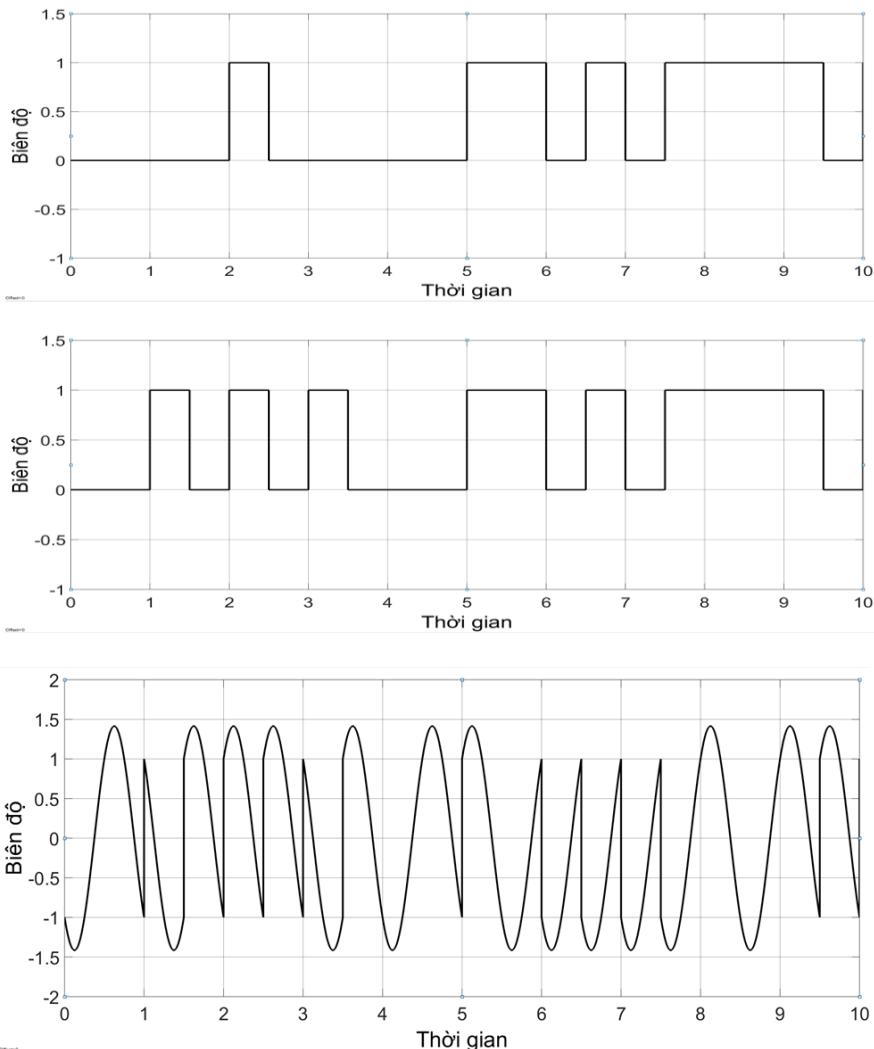
Với biên độ sóng mang : $A = 1$, Tần số góc: $F = 2\pi rad/sec$, $Phase = 3\pi/2$

- Scope3-I: hiển thị chuỗi bit I.
- Scope1-Q: hiển thị chuỗi bit Q.
- Scope2-QPSK: hiển thị tín hiệu điều chế QPSK.



Hình 3-20: Sơ đồ mô phỏng hệ thống điều chế QPSK

Kết quả mô phỏng như sau:

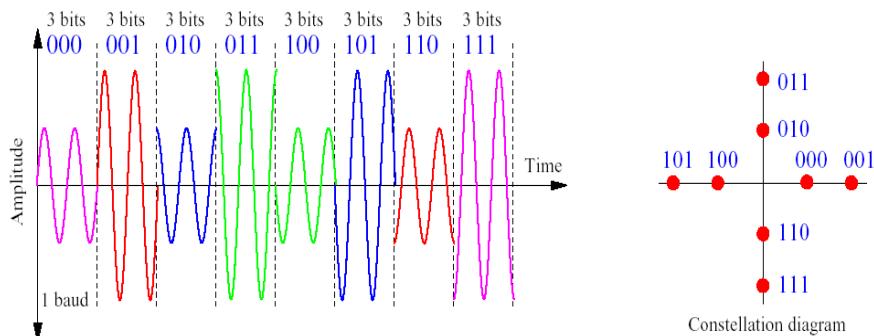


Hình 3-21: Tín hiệu điều chế QPSK với các pha $\pi/4$, $3\pi/4$,
 $5\pi/4$ ($-3\pi/4$), $7\pi/4$ ($-\pi/4$)

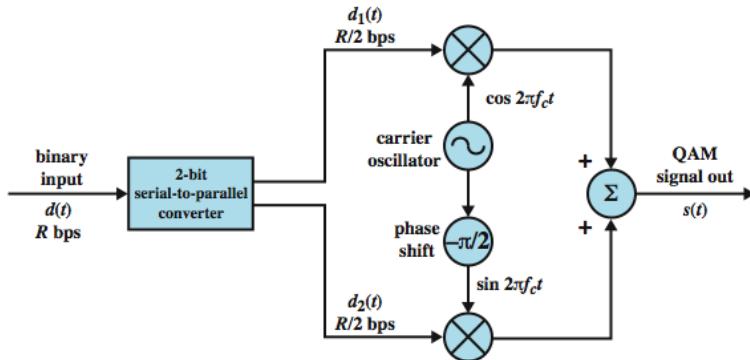
■ QAM Quadrature amplitude modulation

Điều chế biên độ cầu phương (QAM) là một kỹ thuật điều chế tín hiệu tương tự phổ biến được sử dụng trong đường dây thuê bao kỹ thuật số bất đối xứng (ADSL), và trong một số hệ thống không dây. Kỹ thuật điều chế này là sự kết hợp giữa ASK và PSK. QAM cũng có thể được coi là một phần mở rộng của QPSK. QAM sử dụng hai bản sao của tần số sóng mang, một tín hiệu thay đổi 90° so với tín hiệu kia.

Đối với QAM, mỗi sóng mang được điều chế ASK. Hai tín hiệu độc lập được truyền đồng thời trên cùng một đường truyền. Tại máy thu, hai tín hiệu được giải điều chế và kết quả được kết hợp để tạo ra đầu vào nhị phân ban đầu.



Hình 3-22: Biểu diễn dạng sóng và lược đồ chòm sao của tín hiệu QAM

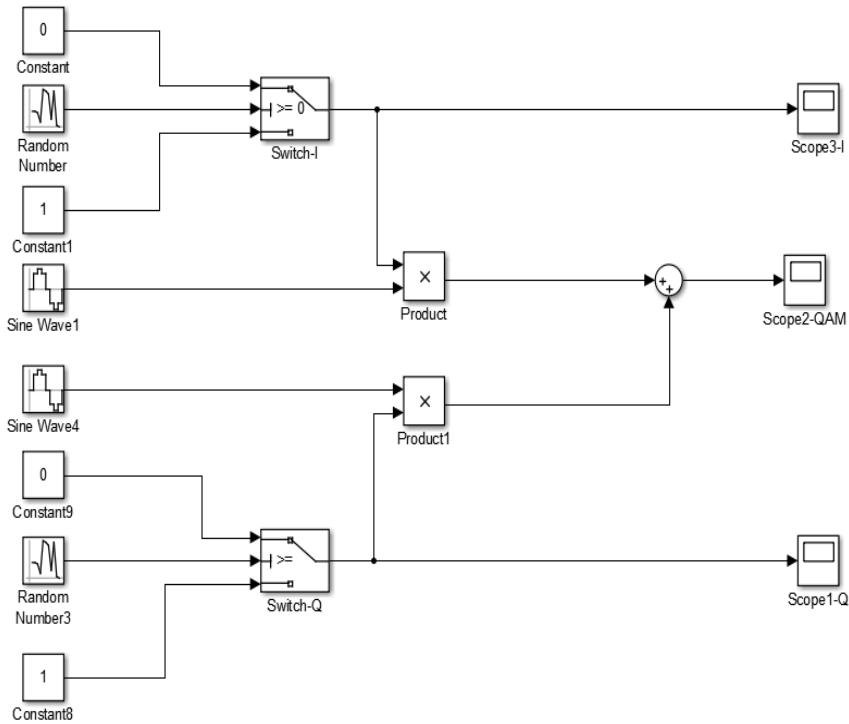


Hình 3-23: Sơ đồ nguyên lý hệ thống điều chế QPSK

Trong sơ đồ, luồng $d_1(t)$ được điều chế ASK bằng cách nhân luồng bit với sóng mang. Sóng mang này được dịch pha 90° và được sử dụng để điều chế ASK của luồng $d_2(t)$. Hai tín hiệu điều chế sau đó được cộng lại và truyền đi. Tín hiệu truyền có thể được biểu thị như sau:

$$s(t) = d_1(t) \cos 2\pi f_c t + d_2(t) \sin 2\pi f_c t \quad (4)$$

Thiết kế điều chế QAM sử dụng Simulink theo mô hình bên dưới:



Hình 3-24: Sơ đồ mô phỏng hệ thống điều chế QAM

Trong đó:

Luồng bit $d_1(t)$ được điều chế ASK với bit 0 được biểu thị bằng sự vắng mặt của sóng mang và bit 1 là sự hiện diện của sóng mang có biên độ không đổi. Luồng bit $d_2(t)$ được điều chế tương tự nhưng với sóng mang được dịch pha 90° . Hai tín hiệu điều chế sau đó được cộng lại và truyền đi.

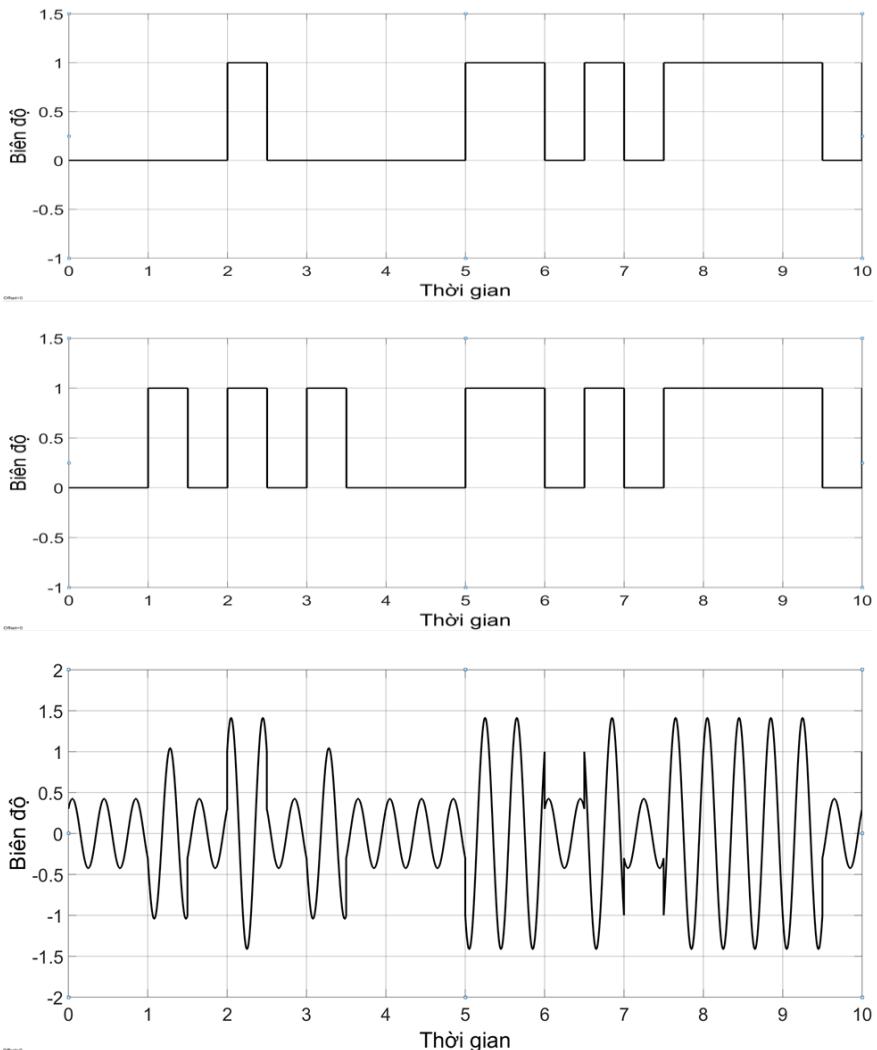
- Khởi Sine wave1 tạo sóng mang hình sin: $O(t) = A * \sin(F*t + Phase)$

Với biên độ sóng mang : $A = 1$, Tần số góc: $F = 2\pi$ rad/sec, Phase = $\pi /2$, tương đương với $O(t) = A * \cos(F*t)$

- Khởi Sine wave4 tạo sóng mang hình sin: $O(t) = A * \sin(F*t + Phase)$

Với biên độ sóng mang : $A = 1$, Tần số góc: $F = 2\pi$ rad/sec, Phase = 0

Kết quả mô phỏng như sau:



Hình 3-25: Tín hiệu điều chế QAM

3.3 Thực hành

■ Phản thực hành tại lớp

- 1) Phân biệt sự khác nhau giữa kỹ thuật mã hóa NRZ – L và NRZI?
- 2) Phân tích ưu nhược điểm của kỹ thuật mã hóa B8ZS và HDB3?
- 3) Thực hiện mô hình mô phỏng bộ điều chế ASK với mức tín hiệu của bit 0 = 0.5, bit 1= 2.

■ Phản thực hành về nhà

- 1) Thực hiện mô hình mô phỏng bộ điều chế QPSK sao cho giản đồ hình sao có các bit như sau: bit 00, pha 0° ; bit 01, pha 90° ; bit 11, pha 180° ; bit 10, pha 270° .
- 2) Kết hợp mô hình bộ điều chế ASK và PSK như trên để thực hiện mô hình mô phỏng bộ điều chế QAM-16.

BÀI 4. CÁC KỸ THUẬT MÃ HÓA DỮ LIỆU TUẦN TỰ

Sau khi học xong bài này, sinh viên có thể:

- Năm được các kỹ thuật mã hóa dữ liệu tuần tự sang tín hiệu số như: điều chế xung mã PCM, điều chế Delta DM.
- Năm được các kỹ thuật mã hóa dữ liệu tuần tự sang tín hiệu tuần tự như: AM, FM và PM.

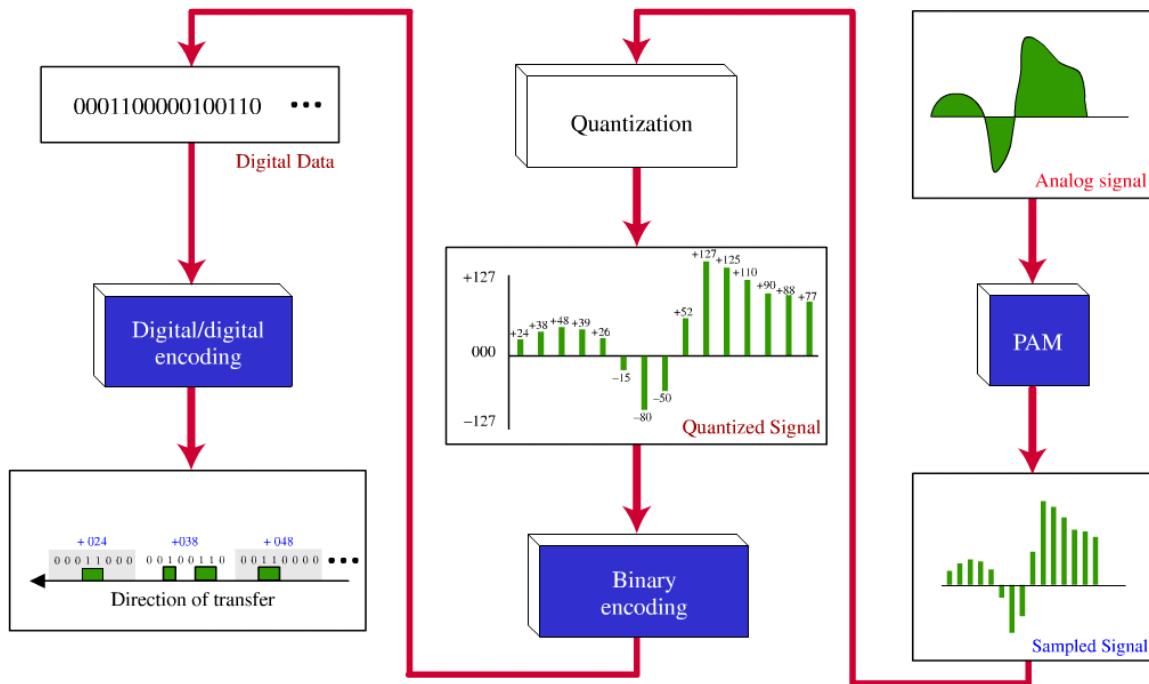
4.1 Các kỹ thuật mã hóa dữ liệu tuần tự sang tín hiệu số

Điều chế xung mã PCM (Pulse Code Modulation)

Ở kỹ thuật điều chế xung mã PCM, nếu tín hiệu được lấy mẫu đều với tốc độ lấy mẫu cao hơn tối thiểu 2 lần tần số tín hiệu cao nhất thì các mẫu thu được chứa đủ thông tin của tín hiệu ban đầu.

Ví dụ dữ liệu tiếng nói có giới hạn tần số là 4000 Hz thì khi điều chế xung mã, chúng ta sử dụng tốc độ lấy mẫu là 8000 mẫu/giây để đảm bảo chứa đủ thông tin của tín hiệu ban đầu, các mẫu này được gọi là PAM (Pulse Amplitude Modulation) samples.

Mô hình điều chế xung mã được tiến hành theo sơ đồ dưới đây:



Hình 4-1: Mô hình điều chế xung mã PCM

- Đầu vào của mô hình là tín hiệu/dữ liệu tuần tự.
- Đầu ra là dữ liệu số được mã hóa bằng các kỹ thuật mã hóa dữ liệu số sang tín hiệu số để truyền đi.
- Cụ thể các bước của mô hình như sau:
 - + Bước 1: Nhận tín hiệu/dữ liệu tuần tự đầu vào.
 - + Bước 2: Tiến hành lấy mẫu PAM → có được các mẫu rời rạc theo tần số lấy mẫu – Sampled Signal.
 - + Bước 3: Tiến hành lượng tử hóa dựa vào biên độ của các mẫu trước đó – Quantized Signal.
 - + Bước 4: Tiến hành nhị phân hóa dữ liệu sử dụng 8 bit đối với mỗi mẫu – Digital Data.
 - + Bước 5: Tiến hành mã hóa dữ liệu số sang tín hiệu số sử dụng các kỹ thuật mã hóa dữ liệu số sang tín hiệu số để truyền dữ liệu đi.

Mã nguồn Matlab dưới đây tương ứng với kỹ thuật PCM:

```

clc;
close all;
clear all;
n=input('Nhập số lượng bit tương ứng với mỗi mẫu : ');
n1=input('Nhập tần số lấy mẫu : ');
L=2^n;
x=0:2*pi/n1:4*pi;
```

```

s=8*sin(x);

subplot(3,1,1);

plot(s);

title('Analog Signal');

ylabel('Amplitude--->');

xlabel('Time--->');

subplot(3,1,2);

stem(s);grid on; title('Sampled Sinal');

ylabel('Amplitude--->'); xlabel('Sample--->');

%quá trình quantization

vmax=8;

vmin=-vmax;

del=(vmax-vmin)/L;

part=vmin:del:vmax;

code=vmin-(del/2):del:vmax+(del/2);

[ind,q]=quantiz(s,part,code);

l1=length(ind);

l2=length(q);

for i=1:l1

if(ind(i)~=0)

ind(i)=ind(i)-1;

```

```
    end  
    i=i+1;  
end  
for i=1:l2  
if(q(i)==vmin-(del/2))  
    q(i)=vmin+(del/2);  
end  
end  
subplot(3,1,3);  
stem(q);grid on;  
title('Quantized Signal');  
ylabel('Amplitude--->');  
xlabel(Sample--->');
```

```
figure  
code=de2bi(ind,'left-msb');  
k=1;  
for i=1:l1  
    for j=1:n  
        coded(k)=code(i,j);  
        j=j+1;
```

```

k=k+1;

end

i=i+1;

end

subplot(2,1,1); grid on;
stairs(coded);

axis([0 100 -2 3]); title('Encoded Signal');

ylabel('Amplitude--->');

xlabel('Time--->');

qunt=reshape(coded,n,length(coded)/n);

index=bi2de(qunt,'left-msb');

q=del*index+vmin+(del/2);

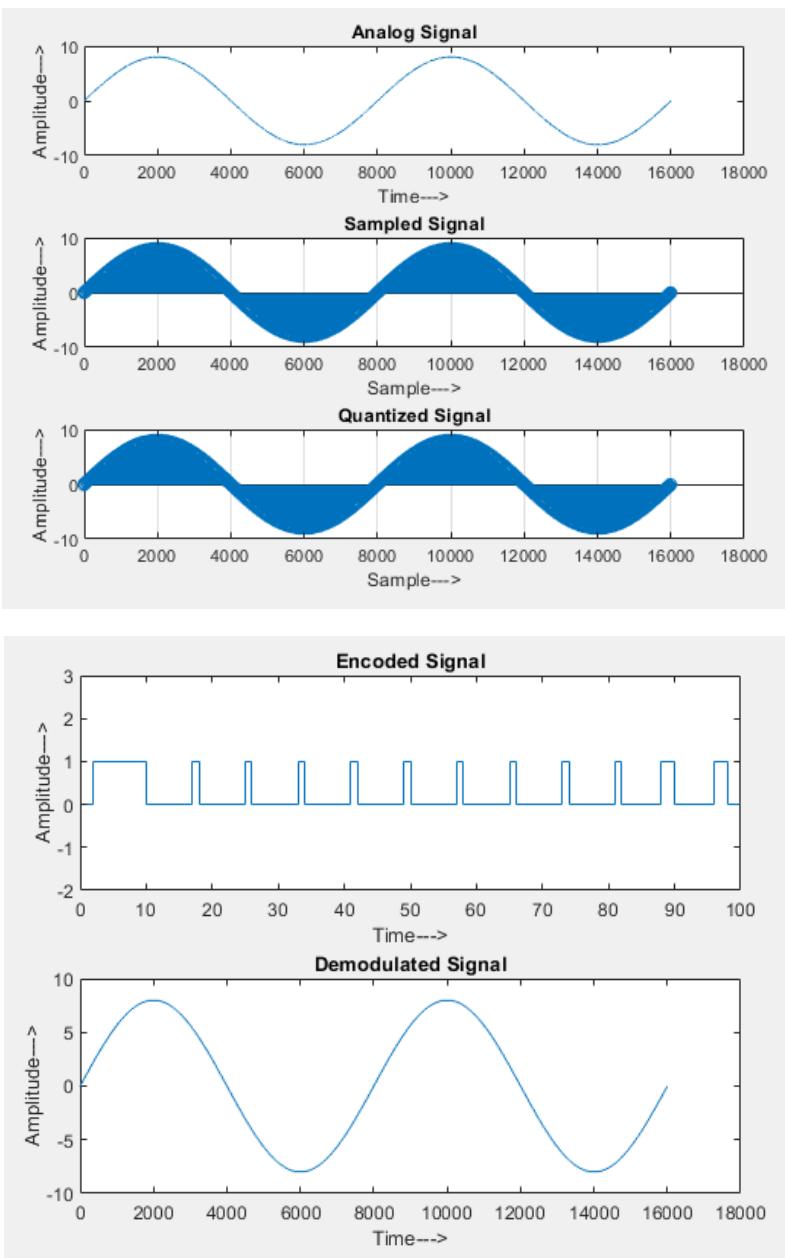
subplot(2,1,2); grid on;
plot(q);
title('Demodulated Signal');

ylabel('Amplitude--->');

xlabel('Time--->');

```

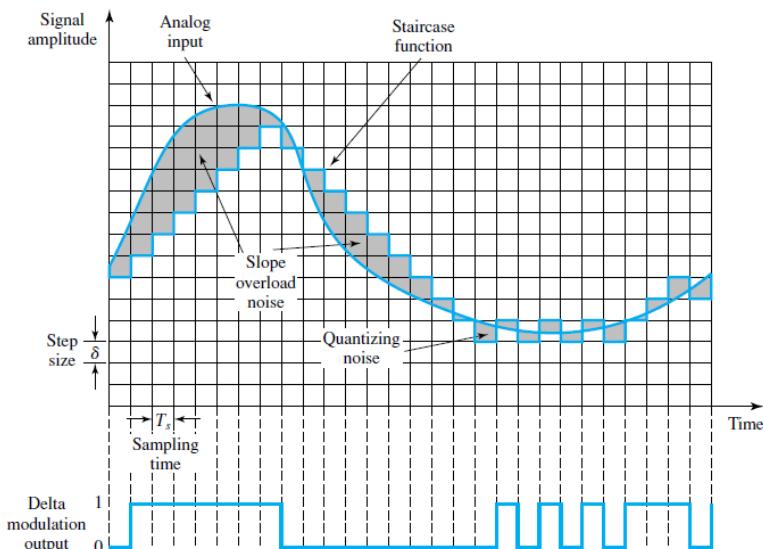
Kết quả của chương trình khi chạy thử với $n = 8$ và $n_1 = 8000$ như hình bên dưới:



Hình 4-2: Quá trình điều chế và giải điều chế tín hiệu tuần tự
sử dụng PCM

Điều chế Delta DM (Delta Modulation)

Ở kỹ thuật điều chế này, tín hiệu/dữ liệu tuần tự sẽ được xấp xỉ bởi hàm bậc thang (staircase). Tín hiệu sẽ đi lên hoặc đi xuống một lượng delta (δ) ứng với thời gian của mỗi mẫu(T_s) như hình bên dưới.



Hình 4-3: Quá trình điều chế sử dụng Delta Modulation

Mã nguồn Matlab dưới đây tương ứng với kỹ thuật Delta Modulation:

```
clc;  
clear all;  
close all;  
a=2;  
subplot(2,1,1);
```

```

t=0:2*pi/50:2*pi;
x=a*sin(t);
l=length(x);
plot(x,'r');
delta=0.2;
hold on
xn=0;
for i=1:l
    if x(i)>=xn(i)
        d(i)=1;
        xn(i+1)=xn(i)+delta;
    else
        d(i)=0;
        xn(i+1)=xn(i)-delta;
    end
end
stairs(xn)
hold on
plot(xn,'c');

```

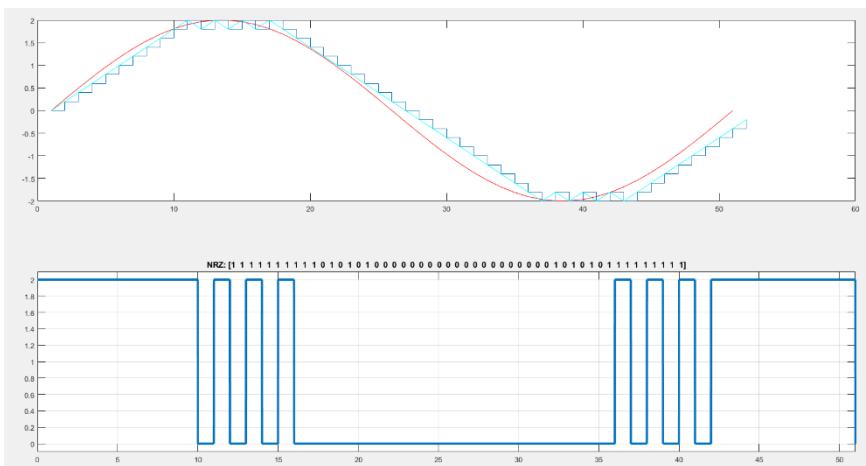
subplot(2,1,2);

```

bitrate = 1;
[t,s] = nrz(d,bitrate);
plot(t,s,'LineWidth',3);
axis([0 t(end) -0.1 2.1])
grid on;
title(['NRZ: [' num2str(d) ']']);

```

Kết quả của chương trình được thể hiện như hình bên dưới:

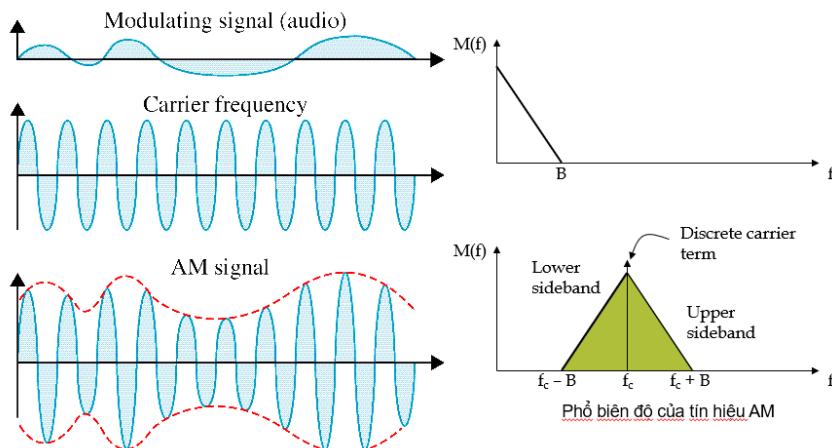


Hình 4-4: Quá trình điều chế sử dụng Delta Modulation

4.2 Các kỹ thuật mã hóa dữ liệu tuần tự sang tín hiệu tuần tự

■ Điều chế biên độ AM (Amplitude Modulation)

Điều chế biên độ (AM) là hình thức điều chế đơn giản nhất, được mô tả trong hình sau:



Hình 4-5: Dạng tín hiệu điều chế AM

Về mặt toán học, quá trình có thể được thể hiện dưới dạng sau:

$$s(t) = [1 + n_a x(t)] \cos 2\pi f_c t \quad (1)$$

Với $\cos 2\pi f_c t$ là sóng mang và $x(t)$ là tín hiệu đầu vào; n_a : hệ số điều chế là tỉ số của biên độ tín hiệu đầu vào và biên độ sóng mang. Số 1 thể hiện thành phần DC của tín hiệu.

Nếu $n_a < 1$ đường bao là một bán sao chính xác của tín hiệu gốc. Nếu $n_a > 1$ đường bao sẽ vượt qua trực thời gian và thông tin bị mất.

Giả sử một thành phần của tín hiệu mang đi điều chế (Modulating signal) có dạng $\cos 2\pi f_m t$. Ta có

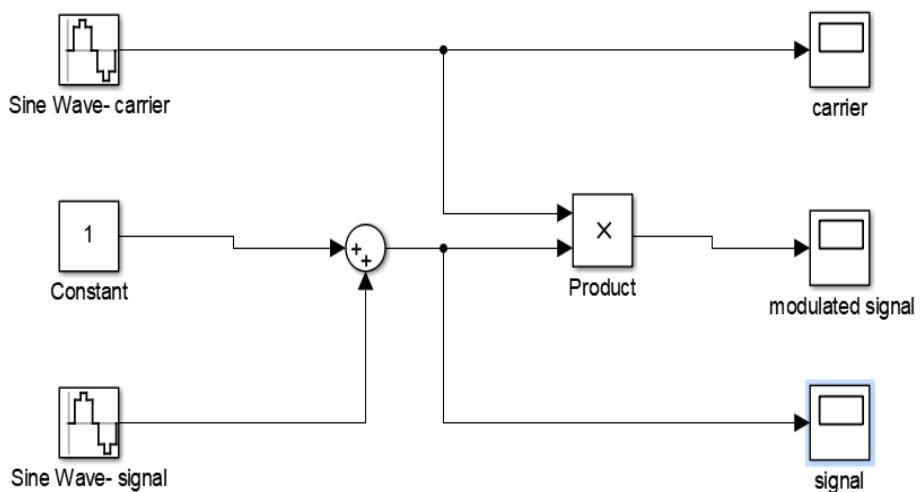
$$s(t) = [1 + n_a \cos 2\pi f_m t] \cos 2\pi f_c t$$

Triển khai biểu thức trên ta được:

$$s(t) = \cos 2\pi f_c t + (n_a/2) \cos 2\pi(f_m - f_c)t + (n_a/2) \cos 2\pi(f_m + f_c)t$$

Vì vậy ta sẽ có phô của tín hiệu điều chế AM như hình trên.

Thiết kế bộ điều chế AM sử dụng Simulink theo mô hình bên dưới.



Hình 4-6: Sơ đồ mô phỏng hệ thống điều chế AM

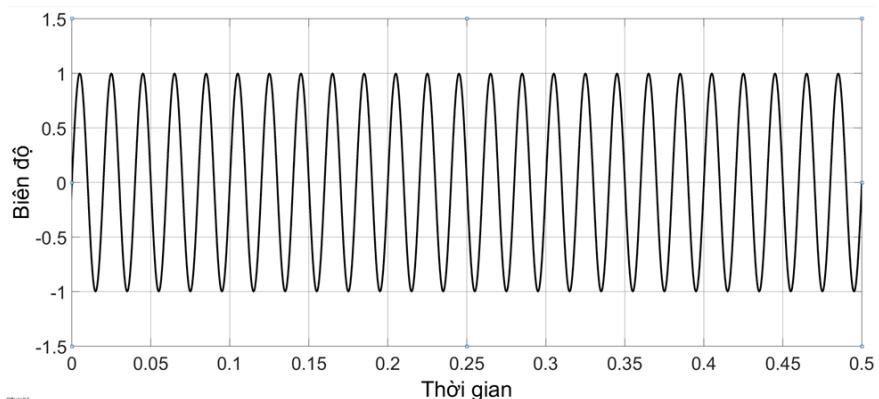
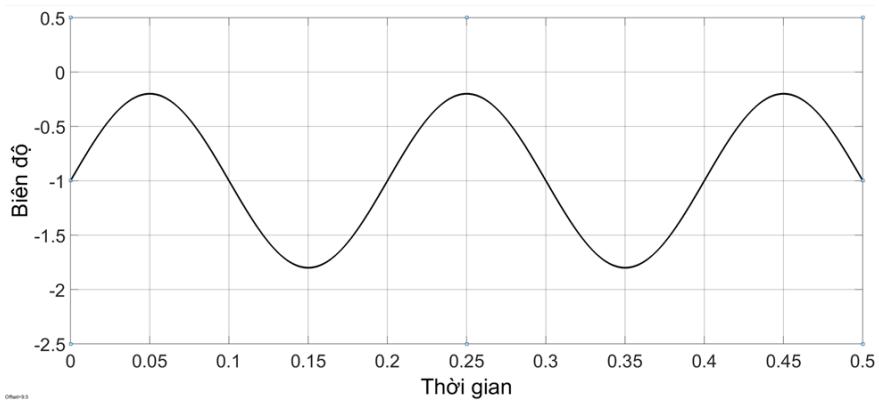
Trong đó:

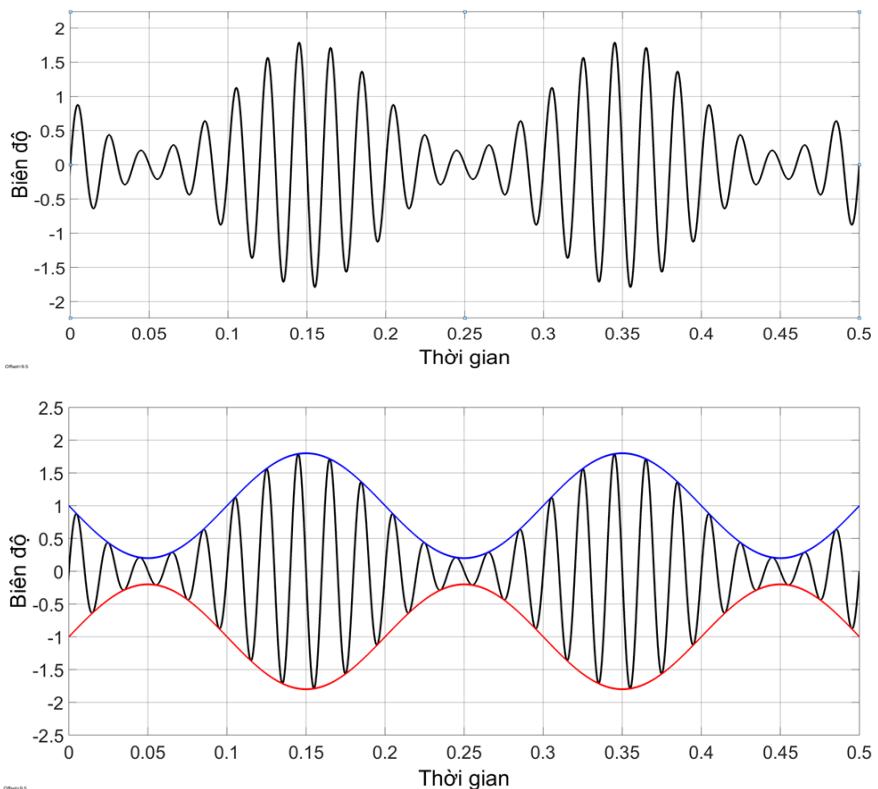
- Khối Sine wave-carrier tạo sóng mang hình sin: $O(t) = A * \sin(F*t + Phase)$ (2)

Với biên độ sóng mang: $A = 1$, Tần số góc: $F = 100\pi$ rad/sec, $Phase = 0$

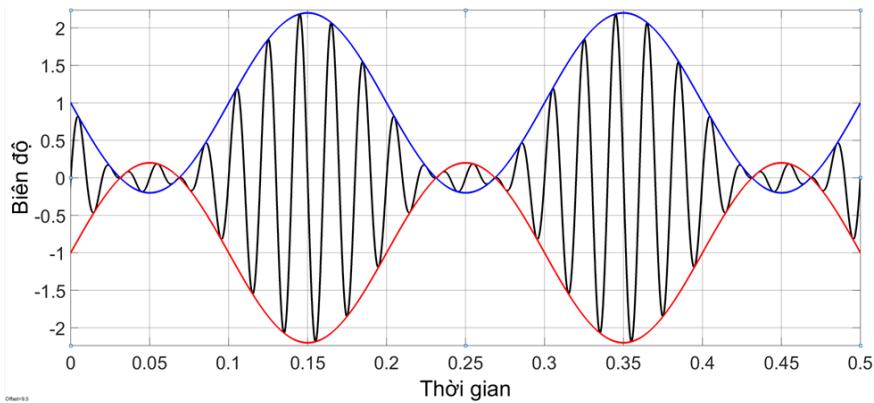
- Biên độ của bộ phát tín hiệu Sine wave signal bằng 0.8 (tương ứng với n_a không lớn hơn 1).
- Khối “+” đưa thành phần DC vào tín hiệu.

Kết quả mô phỏng tương ứng với $n_a = 0.8$ và $n_a = 1.2$ được thể hiện lần lượt ở hai hình dưới.





Hình 4-7: Dạng tín hiệu điều chế AM với hệ số điều chế $n_a = 0.8$



Hình 4-8: Dạng tín hiệu điều chế AM với hệ số điều chế $n_a = 1.2$

Điều chế tần số FM (Frequency Modulation)

Điều chế tần số (FM) là trường hợp đặc biệt của điều chế góc. Tín hiệu điều chế góc được biểu thị là:

$$s(t) = A_c \cos[2\pi f_c t + \phi(t)] \quad (3)$$

Đối với điều chế tần số, đạo hàm của pha tỷ lệ với tín hiệu điều chế:

$$\phi'(t) = n_f m(t)$$

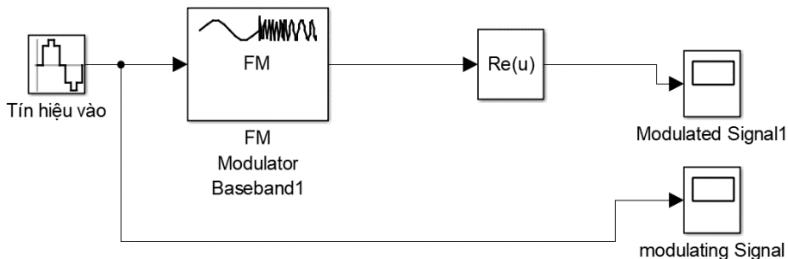
Với n_f là hệ số điều chế tần số và $\phi'(t)$ là đạo hàm của $\phi(t)$, $m(t)$ là tín hiệu được mang đi điều chế.

Nếu $\phi'(t) = -n_f \sin 2\pi f_m t$ là tín hiệu điều chế tần số thì biểu thức cho $s(t)$ như sau:

$$s(t) = \cos[2\pi f_c t + (n_f / 2\pi f_m) \cos 2\pi f_m t]$$

Độ lệch tần số tức thời của tín hiệu sóng mang là $n_f \sin 2\pi f_m t$.

Chúng ta có thiết kế bộ điều chế FM sử dụng Simulink theo mô hình bên dưới:



Hình 4-9: Sơ đồ mô phỏng hệ thống điều chế FM

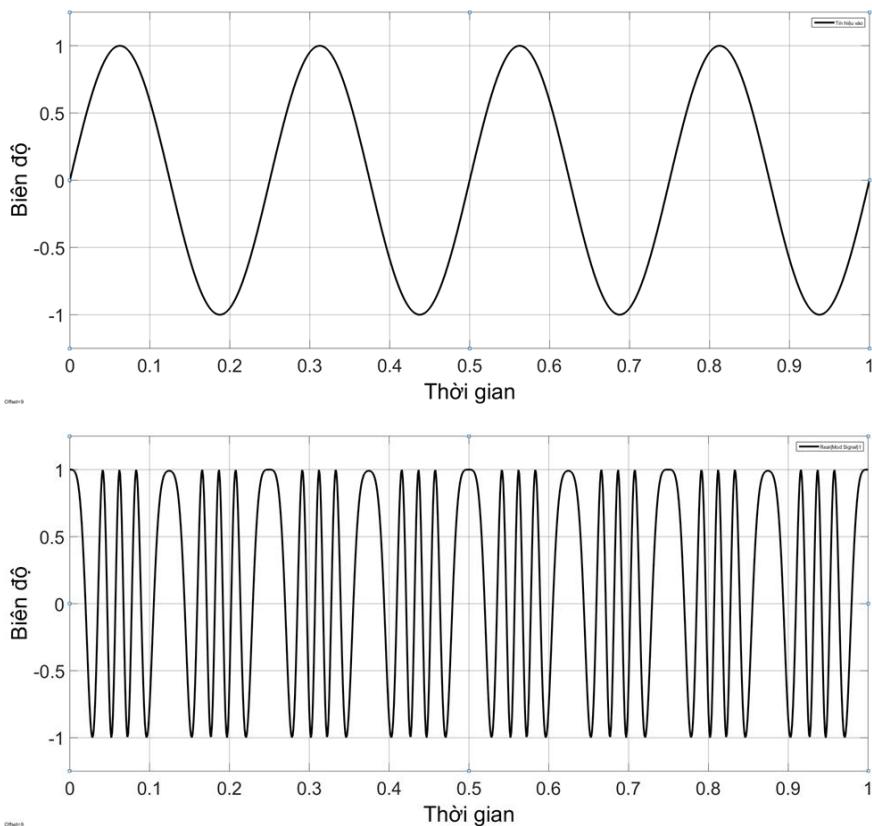
Trong đó:

- Khối Sine wave (tín hiệu vào) tạo nguồn tín hiệu điều chế hình sin: $O(t) = A * \sin(F*t + Phase)$ (4)

Với biên độ sóng mang : $A = 1$, Tần số góc: $F = 8\pi \text{ rad/sec}$, $Phase = 0$

- Khối FM Modulator là bộ điều chế FM có tần số điều chế 50 hz.
- Khối Re(u) để lấy phần thực của tín hiệu điều chế do khối FM cho ra tín hiệu phức.

Kết quả mô phỏng như sau:



Hình 4-10: Biểu diễn dạng tín hiệu điều chế FM

■ Điều chế pha PM (Phase Frequency Modulation)

Điều chế pha (PM) cũng là trường hợp đặc biệt của điều chế góc. Tín hiệu điều chế góc được biểu thị là:

$$s(t) = A_c \cos[2\pi f_{ct} + \phi(t)](5)$$

Đối với điều chế pha, pha tỷ lệ với tín hiệu điều chế:

$$\phi(t) = n_p m(t)$$

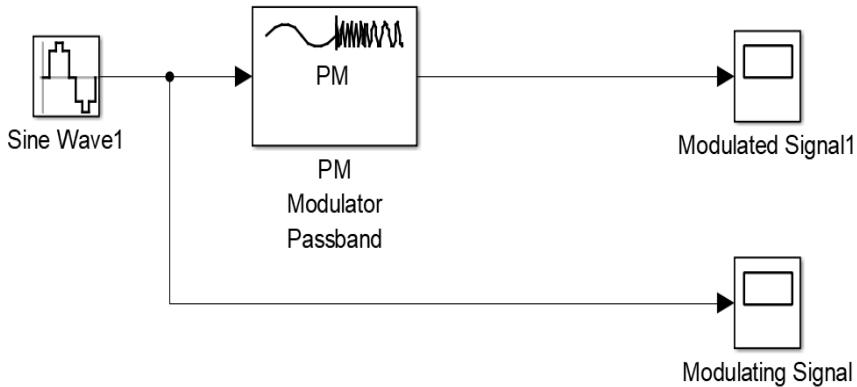
Với n_p là hệ số điều chế pha, $m(t)$ là tín hiệu điều chế. Trong PM, độ lệch pha tức thời $\phi(t)$ với $m(t)$.

Nếu $\phi(t) = n_p \cos 2\pi f_{mt} t$ là tín hiệu điều chế pha và $A_c = 1$ thì biểu thức cho $s(t)$ như sau:

$$s(t) = \cos[2\pi f_{ct} t + n_p \cos 2\pi f_{mt} t]$$

Độ lệch pha tức thời của tín hiệu sóng mang là $n_p \cos 2\pi f_{mt}$, độ lệch pha đỉnh là n_p .

Chúng ta có thiết kế bộ điều chế PM sử dụng Simulink theo mô hình bên dưới:



Hình 4-11: Bộ điều chế PM sử dụng simulink

Trong đó:

- Khối Sine wave1 tạo nguồn tín hiệu điều chế hình sin: $O(t) = A * \sin(F*t + Phase)$ (6)

Với biên độ sóng mang: $A = 1$, Tần số góc: $F = 20\pi$ rad/sec, $Phase = 0$

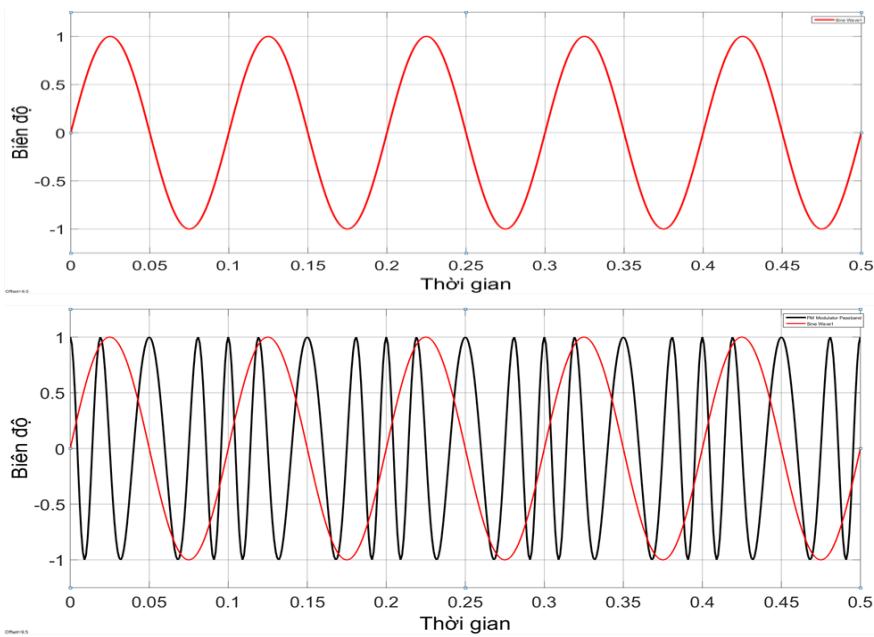
- Khối PM Modulator là bộ điều chế pha PM: $s(t) = \cos(2\pi f_C t + K_C u(t) + \theta)$

Với $f_C = 40$ hz: tần số sóng mang

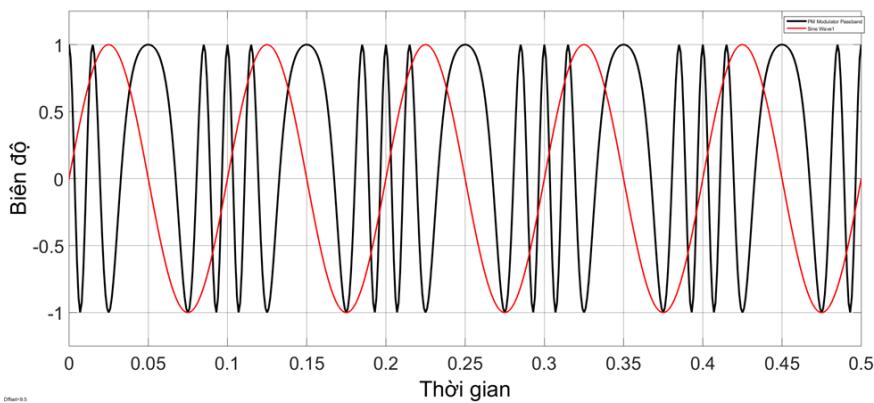
K_C : độ lệch pha (hay còn gọi là độ biến thiên pha).

$\theta = 0$: pha ban đầu của tín hiệu.

Kết quả mô phỏng ta có được như sau:



Hình 4-12: Biểu diễn dạng tín hiệu điều chế PM với độ lệch pha $K_C = \pi/2$



Hình 4-13: Biểu diễn dạng tín hiệu điều chế PM với độ lệch pha $K_C = \pi$

4.3 Thực hành

■ Phản thực hành tại lớp

- 1) Cho biết sự khác nhau giữa Sampled Signal và Quantized Signal?
- 2) Ở PCM nếu sử dụng 4 bit với mỗi mẫu thì các bước của PCM sẽ thay đổi như thế nào?
- 3) Cho biết mối quan hệ giữa step size và Sampling time trong điều chế Delta?
- 4) Thực hiện mô hình mô phỏng bộ điều chế AM bằng bộ điều chế DSB AM Modulator Passband (trong mục Communications System Toolbox\Modulation của thư viện simulink) với hệ số $n_a = 0.7$

■ Phản thực hành về nhà

- 1) Thực hiện lại mô hình mô phỏng bộ điều chế AM bằng bộ điều chế SSB AM Modulator Passband (trong mục Communications System Toolbox\Modulation của thư viện simulink).
- 2) Thực hiện mô hình mô phỏng bộ điều chế FM bằng bộ điều chế FM Modulator Baseband (trong mục Communications System Toolbox\Modulation của thư viện simulink). với tín hiệu là sóng sin có tần số 10 Hz, tần số điều chế (deviation) là 100Hz.

BÀI 5. CÁC KỸ THUẬT GHÉP KÊNH

Sau khi học xong bài này, sinh viên có thể:

- Nắm được các kỹ thuật ghép kênh phân chia theo tần số FDM
- Nắm được các kỹ thuật ghép kênh phân chia theo thời gian TDM

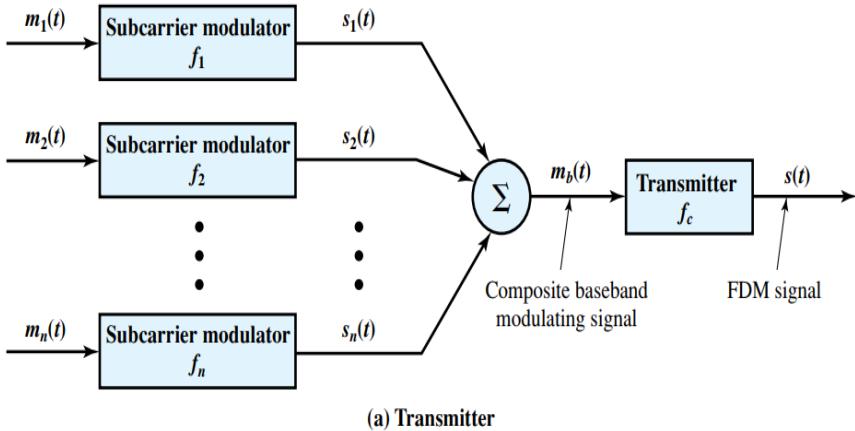
5.1 Ghép kênh phân chia theo tần số FDM (Frequency Division Multiplexing)

FDM có thể được thực hiện khi băng thông hiệu dụng của môi trường truyền lớn hơn nhiều băng thông yêu cầu của tín hiệu được truyền. Nhiều tín hiệu có thể được mang đồng thời nếu mỗi tín hiệu được điều chế trên một tần số sóng mang khác nhau và các tần số sóng mang cách nhau đủ để băng thông của tín hiệu không trùng nhau đáng kể.

Tín hiệu tổng hợp truyền qua môi trường tuần tự. Tuy nhiên, các tín hiệu đầu vào có thể là tín hiệu số hoặc tuần tự. Trong trường hợp đầu vào tín hiệu số, các tín hiệu đầu vào phải được truyền qua modem để được chuyển đổi sang tuần tự.

Mỗi tín hiệu đầu vào sau đó phải được điều chế để di chuyển nó đến dải tần số thích hợp.

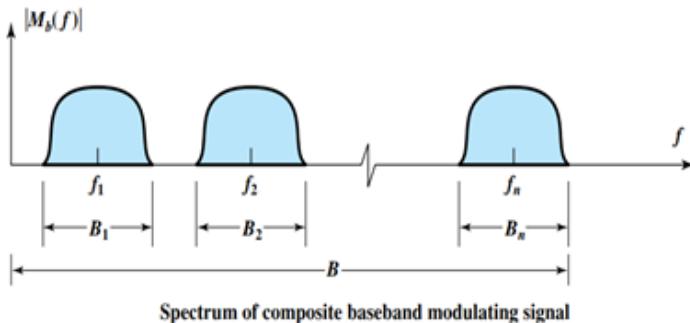
Để ngăn chặn nhiễu, các kênh được phân tách bằng các dải bảo vệ, là các phần không được sử dụng của phổ.



Hình 5-1: Sơ đồ nguyên lý bộ ghép kênh FDM

Trong bộ ghép kênh FDM được hiển thị trong Hình 5-1, các tín hiệu $[m_i(t), i=1,n]$ sẽ được ghép vào cùng một phương tiện truyền dẫn theo cách sau:

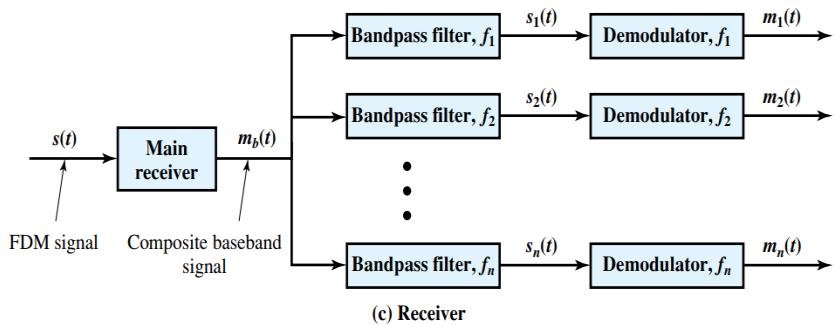
- Mỗi tín hiệu $m_i(t)$ được điều chế trên một sóng mang f_i gọi là sóng mang phụ.
- Bất kỳ loại điều chế nào cũng có thể được sử dụng.
- Các tín hiệu tuần tự, được điều chế sau đó được tổng hợp để tạo ra tín hiệu dải nền tông hợp $m_b(t)$.
- Tín hiệu tổng hợp sau đó có thể được chuyển toàn bộ sang tần số sóng mang khác bằng một bước điều chế bổ sung.



Hình 5-2: Phổ tín hiệu điều chế dải nền tổng hợp

Trong Hình 5-2 cho thấy phổ tín hiệu $m_i(t)$ được dịch chuyển xung quanh f_i . Các tần số f_i phải được chọn sao cho băng thông của các tín hiệu khác nhau không bị chồng chéo đáng kể. Nếu không, sẽ không thể phục hồi các tín hiệu ban đầu.

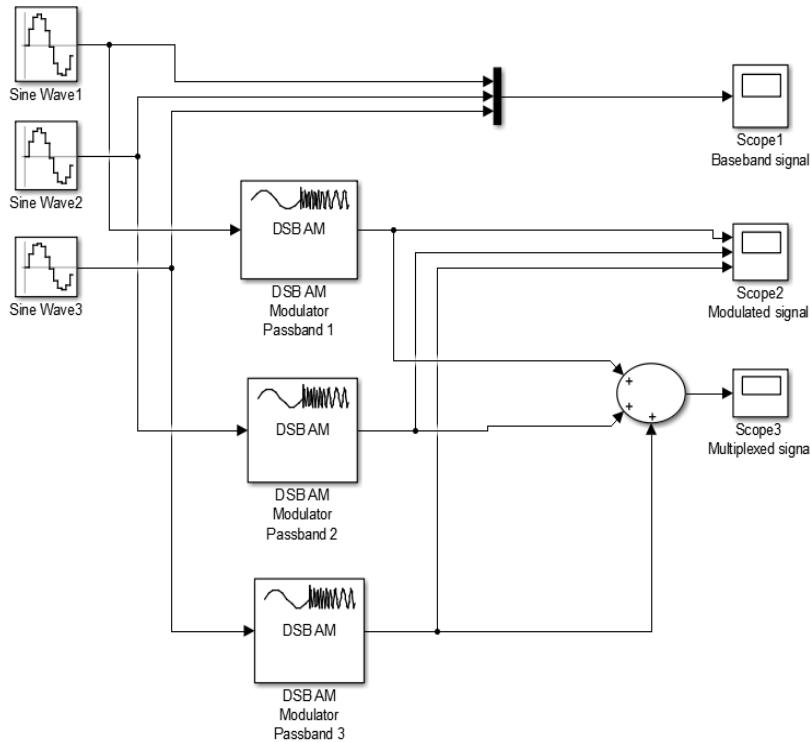
Ở phía thu, tín hiệu điều chế FDM được giải điều chế để truy xuất $m_b(t)$, sau đó được truyền qua n bộ lọc thông dài, mỗi bộ lọc thông dài có tần số trung tâm là f_i . Tín hiệu tổng hợp sẽ được tách thành các thành phần của nó. Mỗi thành phần sau đó được giải điều chế để phục hồi tín hiệu ban đầu.



Hình 5-3: Sơ đồ nguyên lý bộ thu và tách kênh FDM

■ Mô hình bộ ghép kênh FDM sử dụng kỹ thuật điều chế AM

Chúng ta có thể sử dụng Simulink để mô phỏng bộ ghép kênh FDM sử dụng kỹ thuật điều chế biên độ như sau:



Hình 5-4: Mô hình Simulink bộ ghép kênh FDM sử dụng điều biến AM

Trong đó:

- Khối Sine wave: giả lập tín hiệu (mang dữ liệu) hình sin: $O(t) = A * \sin(F*t + Phase)$ (1)

Với biên độ sóng mang: $A = 1$, Tần số góc: $F = 200\pi$ rad/sec, $Phase = 0$

- Khối Sine wave1: giả lập tín hiệu (mang dữ liệu)
hình sin: $O(t) = A * \sin(F*t + Phase)$ (2)

Với biên độ sóng mang: $A = 1$, Tân số góc: $F = 200\pi$
 rad/sec , $Phase = \pi/2$

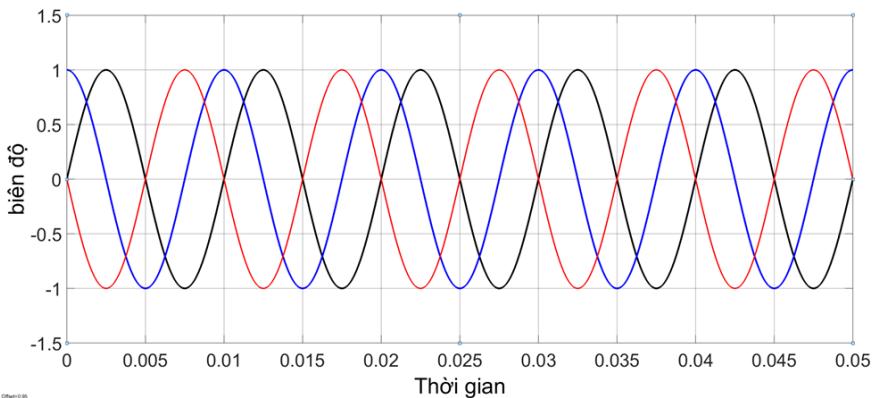
- Khối Sine wave2: giả lập tín hiệu (mang dữ liệu)
hình sin: $O(t) = A * \sin(F*t + Phase)$ (3)

Với biên độ sóng mang: $A = 1$, Tân số góc: $F = 200\pi$
 rad/sec , $Phase = \pi$

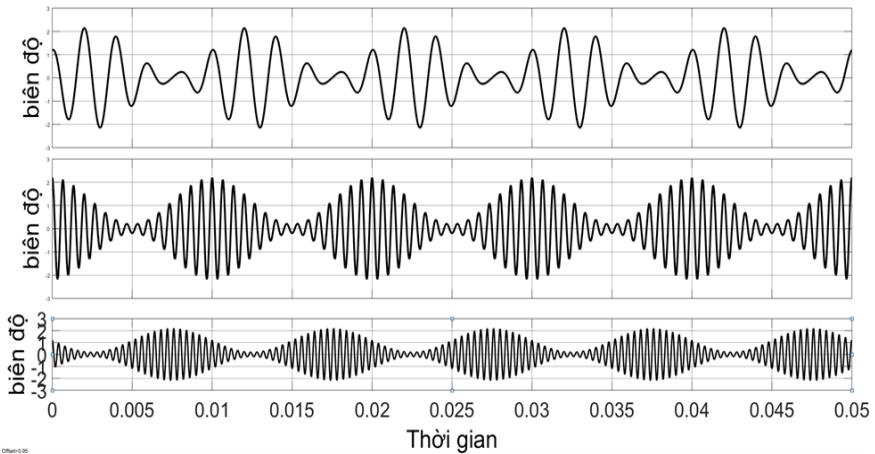
- Khối DSB AM Modulator Passband1: là bộ điều chế AM có tần số sóng mang 500 hz, input signal offset = 1.2 (thành phần DC), pha ban đầu = 0.
- Khối DSB AM Modulator Passband2: là bộ điều chế AM có tần số sóng mang 1500 hz, input signal offset = 1.2 (thành phần DC), pha ban đầu = 0.
- Khối DSB AM Modulator Passband3: là bộ điều chế AM có tần số sóng mang 2500 hz, input signal offset = 1.2 (thành phần DC), pha ban đầu = 0.
- Scope1 Baseband signal: biểu diễn 3 nguồn tín hiệu ban đầu trước khi ghép kênh.
- Scope2 Modulated signal: biểu diễn 3 kênh tín hiệu sau khi đã điều chế AM.

- Scope3 Multiplexed signal: biểu diễn tín tổng hợp FDM sau khi 3 kênh tín hiệu được cộng thành một tín hiệu duy nhất.

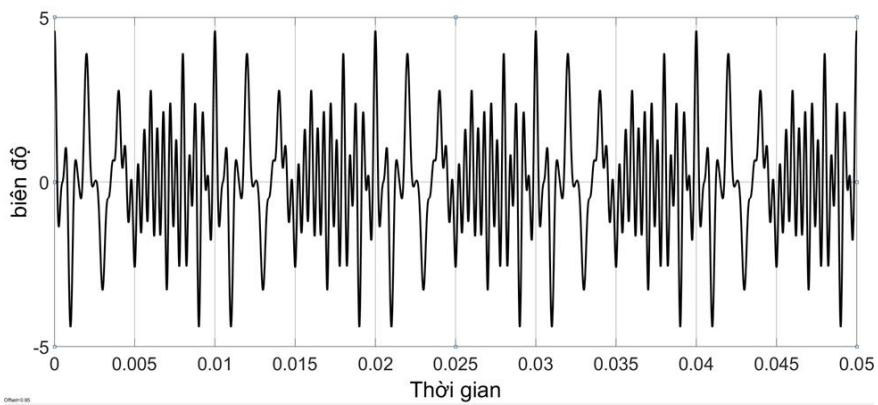
Kết quả mô phỏng như sau:



Hình 5-5: Biểu diễn 3 tín hiệu đầu vào

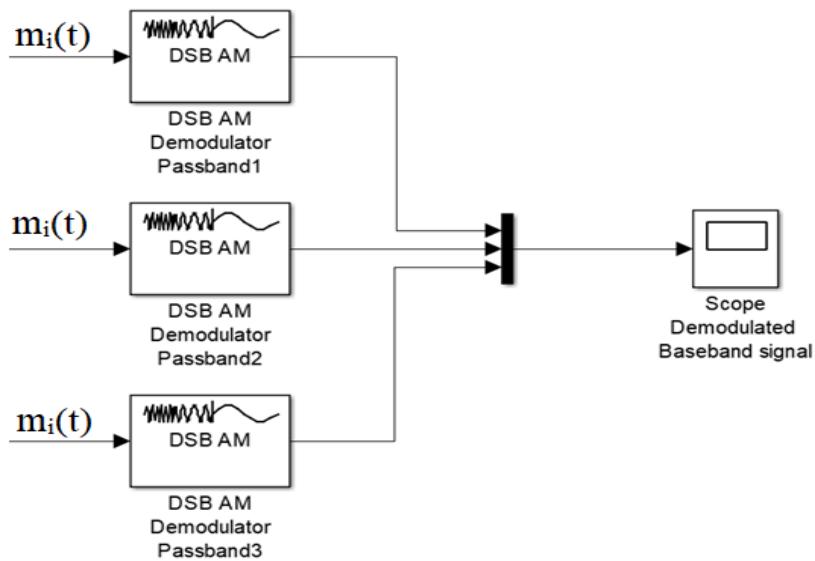


Hình 5-6: Ba tín hiệu điều chế AM tương ứng với 3 tín hiệu đầu vào



Hình 5-7: Biểu diễn tín hiệu tổng hợp sau ghép kênh ở đầu ra

Chúng ta có thể sử dụng mô hình sau để mô phỏng bộ tách kênh FDM sử dụng điều chế AM:

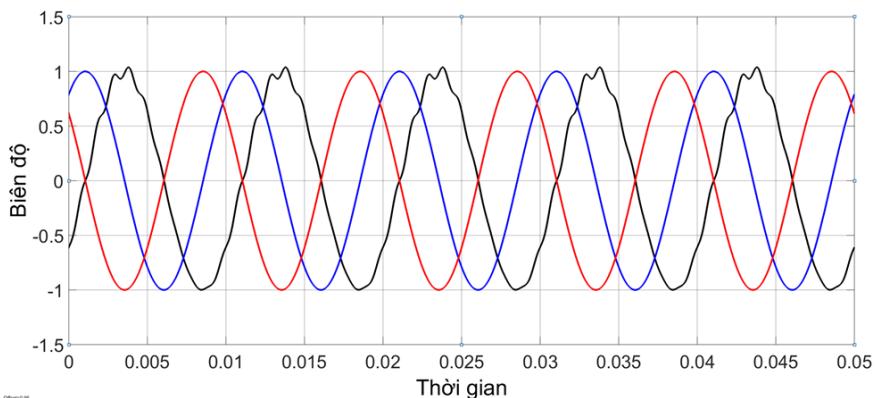


Hình 5-8: Mô hình Simulink bộ tách kênh FDM sử dụng điều biến AM

Trong đó:

- $m_i(t)$: tín hiệu thành phần sau khi tín hiệu tổng hợp ở đầu ra của bộ ghép kênh như Hình 5-4. đã qua bộ lọc thông dải.
- Khối DSB AM Demodulator Passband1: là bộ giải điều chế AM có tần số sóng mang 500 Hz, input signal offset = 1.2, pha ban đầu = 0, bộ lọc thông thấp sau khi giải điều chế AM là bộ lọc Butterworth có bậc lọc là 4 và tần số cắt là 400 Hz.
- Khối DSB AM Demodulator Passband2: là bộ giải điều chế AM có tần số sóng mang 1500 Hz, input signal offset = 1.2, pha ban đầu = 0, bộ lọc thông thấp sau khi giải điều chế AM là bộ lọc Butterworth có bậc lọc là 4 và tần số cắt là 400 Hz.
- Khối DSB AM Demodulator Passband1: là bộ giải điều chế AM có tần số sóng mang 2500 Hz, input signal offset = 1.2, pha ban đầu = 0, bộ lọc thông thấp sau khi giải điều chế AM là bộ lọc Butterworth có bậc lọc là 4 và tần số cắt là 400 Hz.
- Scope Demodulated Baseband signal: biểu diễn 3 nguồn tín hiệu (mang dữ liệu) sau khi đã tách kênh và lọc.

Kết quả mô phỏng như sau:



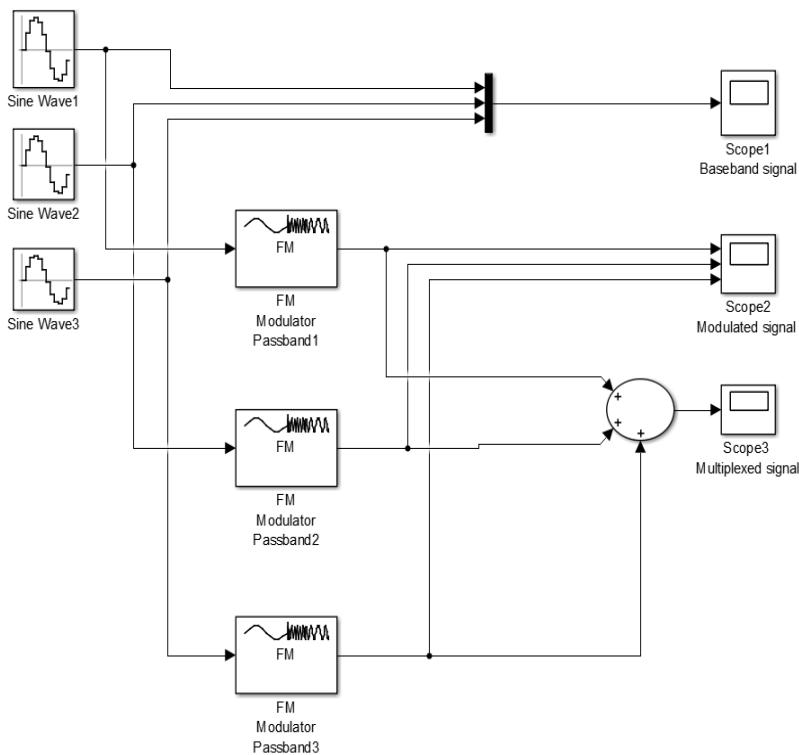
Hình 5-9: Biểu diễn 3 tín hiệu (mang dữ liệu) sau khi tách kênh

■ Mô hình bộ ghép kênh FDM sử dụng kỹ thuật điều chế FM

Chúng ta có thể sử dụng Simulink để mô phỏng bộ ghép kênh FDM sử dụng kỹ thuật điều chế tần số FM như sau:

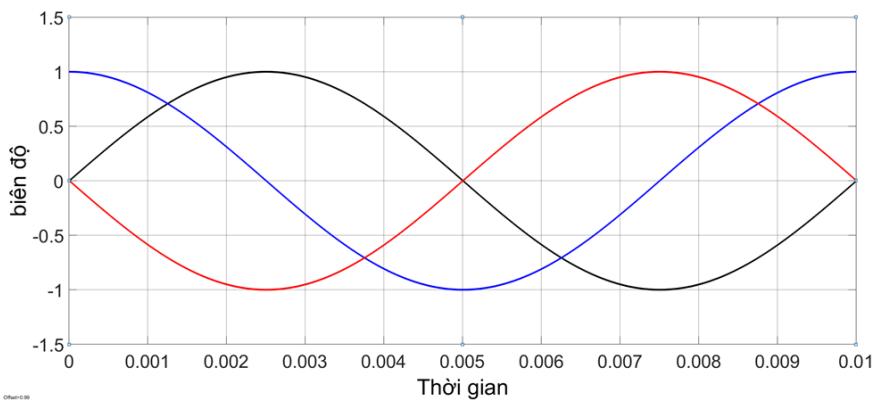
- Khối FM Modulator Passband1: là bộ điều chế FM có tần số sóng mang 1000 Hz, pha ban đầu = 0, độ dịch tần (Frequency Deviation) = 300 Hz.
- Khối FM Modulator Passband2: là bộ điều chế FM có tần số sóng mang 2000 Hz, pha ban đầu = 0, độ dịch tần (Frequency Deviation) = 300 Hz.

- Khối FM Modulator Passband1: là bộ điều chế FM có tần số sóng mang 3000 Hz, pha ban đầu = 0, độ dịch tần (Frequency Deviation) = 300 Hz.
- Scope1 Baseband signal, Scope2 Modulated signal, Scope3 Multiplexed signal: tương tự như đối với mô hình ghép kênh sử dụng kỹ thuật điều biên.

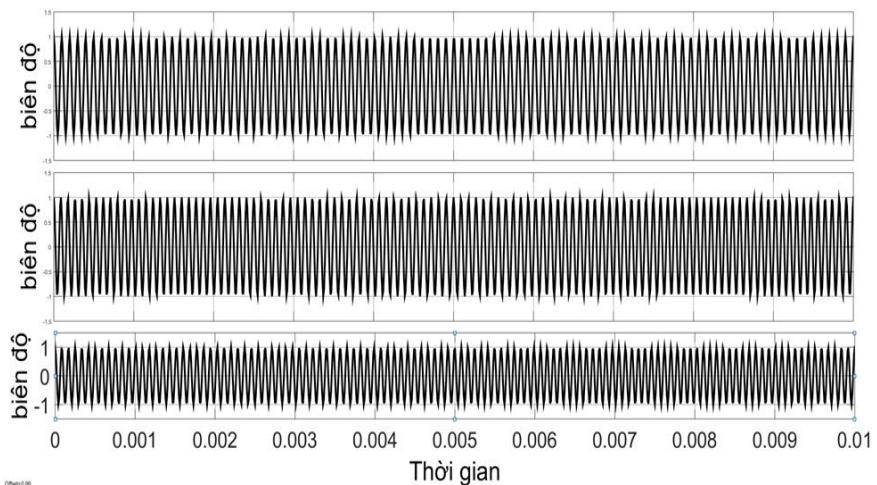


Hình 5-10: Mô hình Simulink bộ ghép kênh FDM sử dụng
điều chế FM

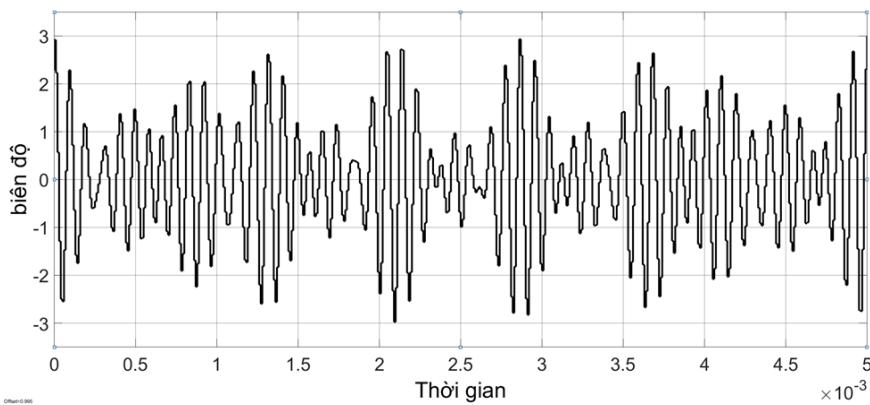
Kết quả mô phỏng như sau:



Hình 5-11: Biểu diễn 3 tín hiệu đầu vào

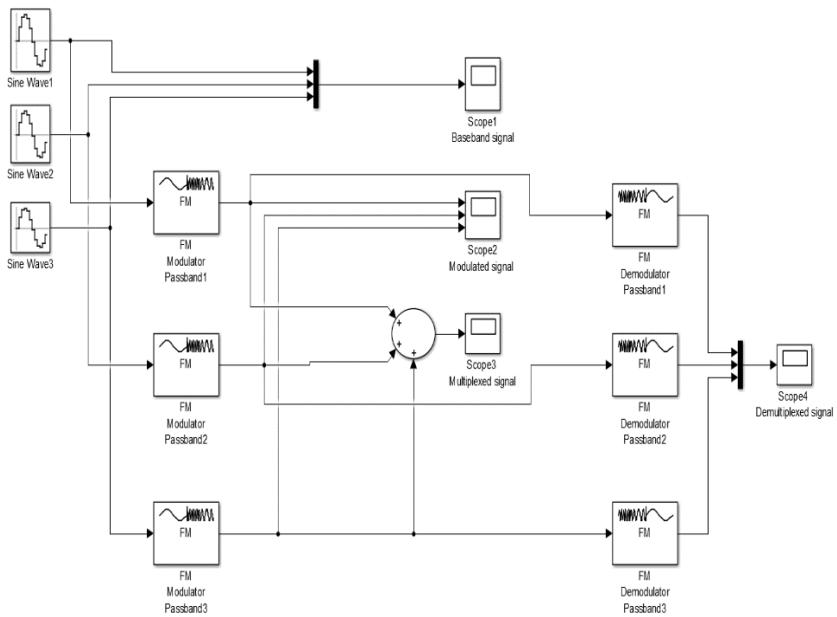


Hình 5-12: Ba tín hiệu điều chế FM tương ứng với 3 tín hiệu đầu vào



Hình 5-13: Biểu diễn tín hiệu tổng hợp sau ghép kênh ở đầu ra (FM)

Chúng ta có thể sử dụng mô hình sau để mô phỏng bộ tách kênh FDM sử dụng điều chế FM:

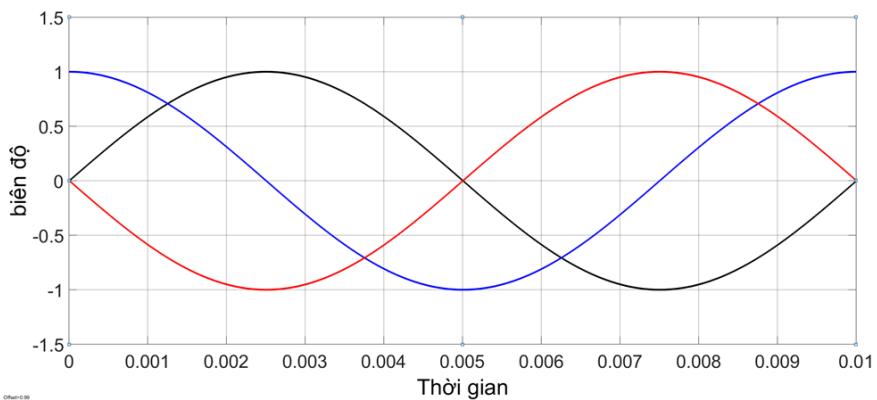


Hình 5-14: Mô hình Simulink bộ tách kênh FDM sử dụng điều tàn FM

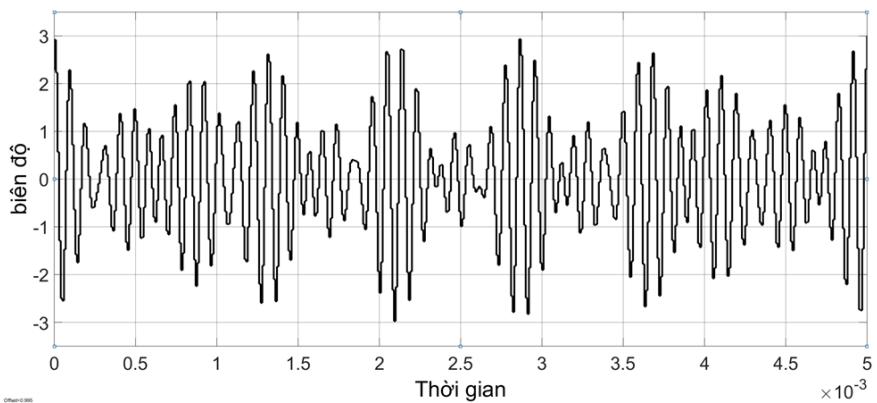
Trong đó:

- Các khối FM Modulator Passband1, 2, 3: là bộ điều chế FM có tần số sóng mang lần lượt là 10000 Hz, 11000 Hz , 12000 Hz pha ban đầu = 0, độ dịch tần (Frequency Deviation) = 300 Hz.
- Khối FM Demodulator Passband1: là bộ giải điều chế FM có tần số sóng mang **10000Hz**, pha ban đầu = 0, độ dịch tần (Frequency Deviation) = 300 Hz, bộ lọc hilbert có bậc lọc = 100.
- Khối FM Demodulator Passband2: là bộ giải điều chế FM có tần số sóng mang **11000Hz**, pha ban đầu = 0, độ dịch tần (Frequency Deviation) = 300 Hz, bộ lọc hilbert có bậc lọc = 100.
- Khối FM Demodulator Passband3: là bộ giải điều chế FM có tần số sóng mang **12000Hz**, pha ban đầu = 0, độ dịch tần (Frequency Deviation) = 300 Hz, bộ lọc hilbert có bậc lọc = 100.

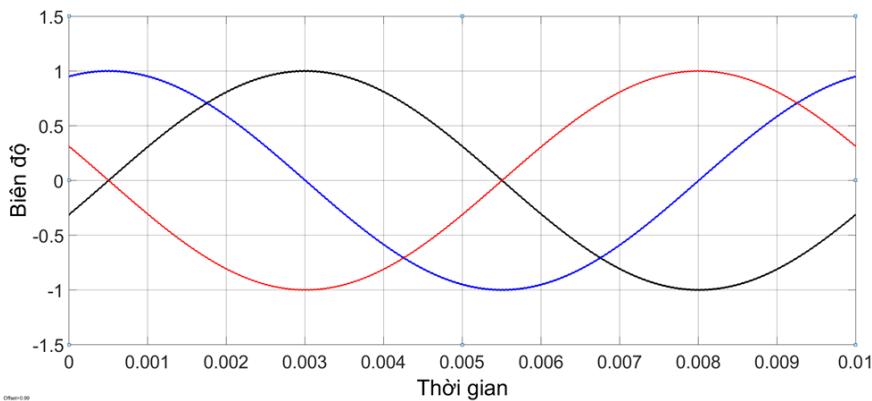
Kết quả mô phỏng như sau:



Hình 5-15: Biểu diễn 3 tín hiệu đầu vào



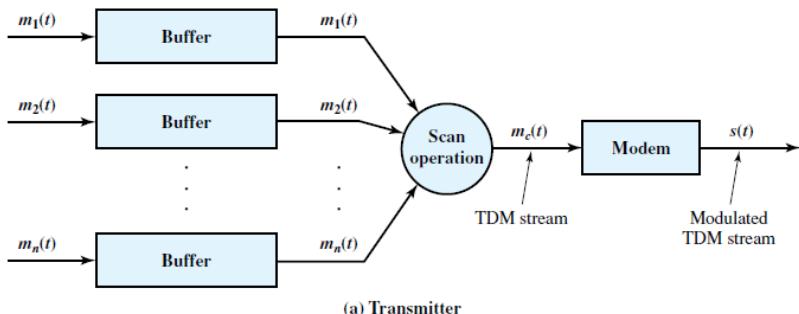
Hình 5-16: Biểu diễn tín hiệu tổng hợp sau ghép kênh FM ở đầu ra



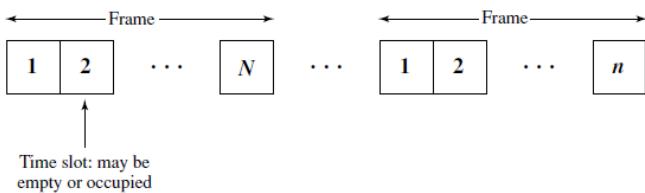
Hình 5-17: Biểu diễn 3 tín hiệu mang dữ liệu sau khi tách kênh FM

5.2 Ghép kênh đồng bộ phân chia theo thời gian TDM

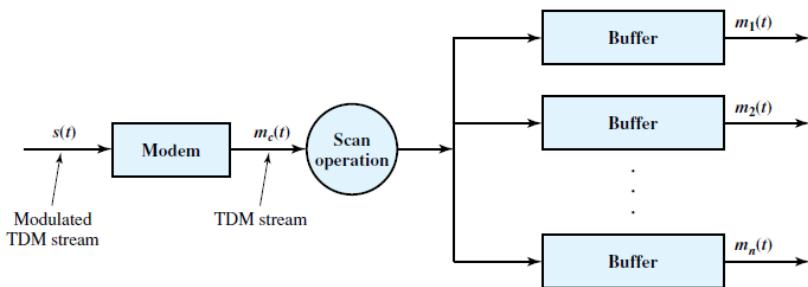
Kỹ thuật ghép kênh đồng bộ phân chia theo thời gian TDM (Synchronous time division multiplexing) yêu cầu tốc độ đường truyền phải lớn hơn tổng của tốc độ tất cả các tín hiệu cần ghép kênh. Ví dụ chúng ta cần ghép kênh TDM đồng bộ cho 6 tín hiệu, mỗi tín hiệu có tốc độ dữ liệu là 9.6 kbps. Như vậy, về mặt lý thuyết đường truyền cần phải có tốc độ ít nhất là 57.6 kbps để có thể ghép được 6 tín hiệu này.



(a) Transmitter



(b) TDM frames



(c) Receiver

Hình 5-18: Mô hình hoạt động của hệ thống ghép kênh
đồng bộ phân chia theo thời gian

Ở hình 5.18a, giả sử n tín hiệu cần ghép kênh trên đường truyền [$m_i(t), i = 1, n$]. Dữ liệu của từng nguồn tín hiệu đầu tiên sẽ được lưu trữ trong bộ nhớ đệm của từng tín hiệu. Mỗi bộ nhớ đệm có độ lớn thông thường là 1 bit. Sau đó, những bộ nhớ đệm này sẽ được quét lần lượt để tạo thành tín hiệu $m_c(t)$. Tốc độ quét cần phải đủ nhanh để làm trống bộ đệm của mỗi tín hiệu

trước khi có dữ liệu đến bộ đệm. Do vậy, tốc độ của $m_c(t)$ ít nhất sẽ bằng tổng của các tín hiệu $m_i(t)$ thành phần.

Tín hiệu số $m_c(t)$ có thể được truyền ngay cho bên nhận hoặc sử dụng modem để điều chế thành tín hiệu tuần tự và sau đó sẽ được truyền đi.

Dữ liệu được truyền đi sẽ có dạng như hình 5.18b và được tổ chức thành **frames**. Mỗi frame sẽ chứa một số **time slot**, mỗi time slot sẽ có độ dài bằng với độ lớn của bộ nhớ đệm (1 bit). Ở mỗi frame có thể có 1 hoặc nhiều slot cho mỗi nguồn tín hiệu. Dãy các slot truyền tải một nguồn tín hiệu (từ frame này sang frame khác) được gọi là **channel**.

Ở phía nhận (hình 5.18c) dữ liệu sẽ được tách ra về từng nguồn nhận tương ứng. Đối với từng nguồn truyền $m_i(t)$ sẽ có tương ứng nguồn nhận (có cùng tốc độ dữ liệu với nguồn truyền).

Ghép kênh đồng bộ ở đây mang ý nghĩa là những time slot sẽ được gán trước và không thay đổi cho mỗi nguồn. Time slot cho mỗi nguồn sẽ được giữ cho nguồn đó dù cho tại thời điểm đó nguồn có dữ liệu truyền hay là không. Điều này sẽ dẫn đến sự lãng phí băng thông. Trong thực tế, để giảm bớt sự lãng phí này, những nguồn tín hiệu có tốc độ lớn hơn sẽ được phân chia nhiều time slot hơn các nguồn tín hiệu có tốc độ nhỏ hơn để tận dụng tối đa các time slot giành cho từng nguồn.

Đoạn mã nguồn Matlab dưới đây sẽ minh họa cho mô hình ở hình 5.18.

```
clc;
```

```
close all;  
clear all;  
  
  
x=0:.5:4*pi;  
sig1=8*sin(x);  
l=length(sig1);  
sig2=8*triang(l);  
  
  
subplot(2,2,1);  
plot(sig1);  
title('Sinusoidal Signal');  
ylabel('Amplitude-->');  
xlabel('Time-->');  
subplot(2,2,2);  
plot(sig2);  
title('Triangular Signal');  
ylabel('Amplitude-->');  
xlabel('Time-->');  
  
  
subplot(2,2,3);  
stem(sig1);
```

```

title('Sampled Sinusoidal Signal');

ylabel('Amplitude--->');

xlabel("Time--->");

subplot(2,2,4);

stem(sig2);

title('Sampled Triangular Signal');

ylabel('Amplitude--->');

xlabel("Time--->");

l1=length(sig1);

l2=length(sig2);

for i=1:l1

sig(1,i)=sig1(i);

sig(2,i)=sig2(i);

end

```

```
tdmsig=reshape(sig,1,2*l1);
```

```

figure

stem(tdmsig);

title('TDM Signal');

ylabel('Amplitude--->');

```

```
xlabel('Time--->');

demux=reshape(tdmsig,2,11);

for i=1:11

sig3(i)=demux(1,i);

sig4(i)=demux(2,i);

end
```

```
figure

subplot(2,1,1)

plot(sig3);

title('Recovered Sinusoidal Signal');

ylabel('Amplitude--->');

xlabel('Time--->');

subplot(2,1,2)

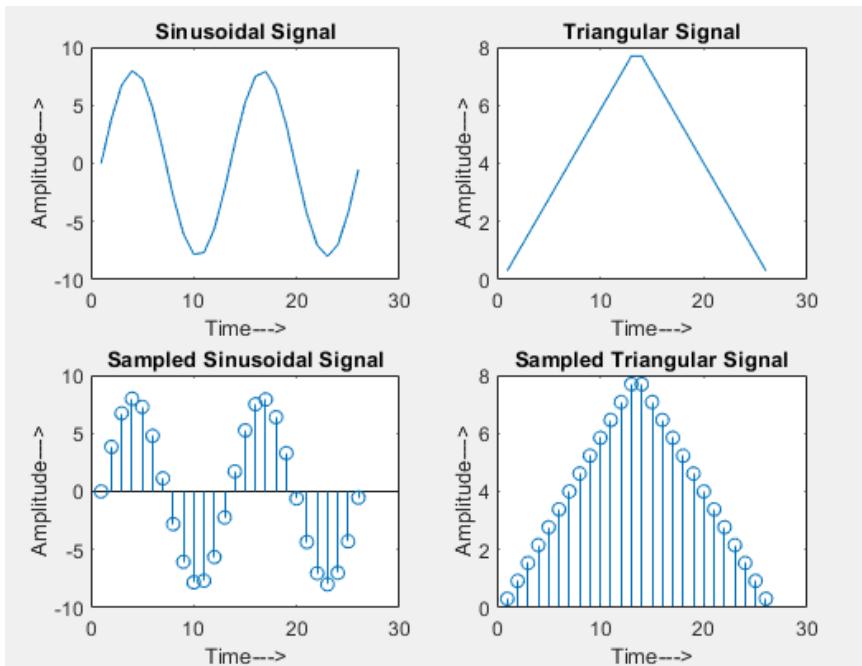
plot(sig4);

title('Recovered Triangular Signal');

ylabel('Amplitude--->');

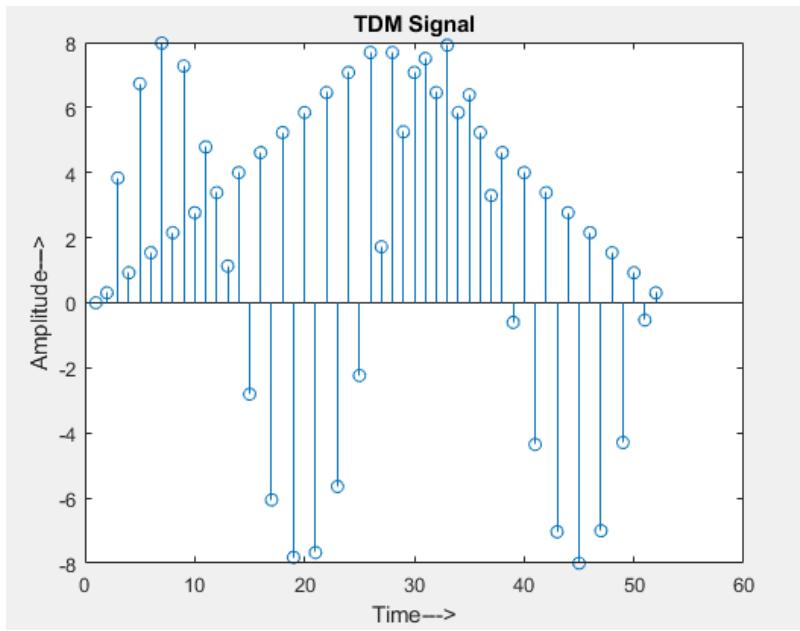
xlabel('Time--->');
```

Ở ví dụ này, chúng ta sẽ tiến hành ghép kênh hai tín hiệu: một tín hiệu hình sin và một tín hiệu triangular như hình dưới. Trong đó, hai tín hiệu tuần tự này sẽ được lấy mẫu dựa vào hàm *stem(signal)* của Matlab.



Hình 5-19: Tín hiệu đầu vào sử dụng để ghép kênh đồng bộ theo TDM

Sau đó, dữ liệu sẽ được tổng hợp thông qua việc ghép kênh TDM để cho ra kết quả như hình 5.20 bên dưới.



Hình 5-20: Tín hiệu tổng hợp sử dụng ghép kênh đồng bộ theo TDM

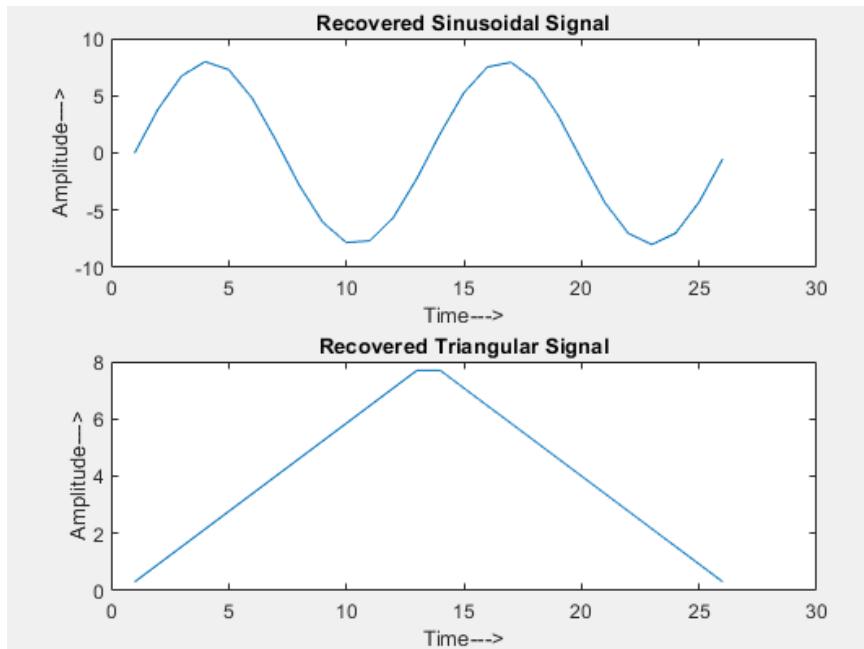
Chúng ta có thể thấy rõ sự xen kẽ lẫn nhau (chiếm từng time slot) của hai tín hiệu nguồn ở tín hiệu đồng bộ TDM, việc này được thực hiện dựa vào việc đọc bộ nhớ đệm lần lượt của từng nguồn như đã được mô tả ở mô hình 5.18a và được minh họa bằng đoạn mã nguồn sau đây:

```

for i=1:11
    sig(1,i)=sig1(i);
    sig(2,i)=sig2(i);
end

```

Tiếp theo đó, tín hiệu này sẽ được truyền sang cho bên nhận và được giải mã lại thành dạng tín hiệu ban đầu để chuyển tới từng nguồn nhận tương ứng như ở hình 5.21.



Hình 5-21: Dữ liệu được chia ra về từng nguồn nhận tương ứng

5.3 Thực hành

■ Phân thực hành tại lớp

- 1) Cho biết còn bao nhiêu dạng điều chế AM có thể dùng để ghép kênh?
- 2) Thực hiện mô hình ghép kênh dùng kỹ thuật điều biên DSBSC (DSB Suppressed carrier AM) với 4 kênh data tự chọn có tần số 50Hz và 4 pha khác nhau.

■ Phân thực hành về nhà

- 1) Thực hiện mô hình ghép, tách kênh dùng kỹ thuật điều tần FM với 5 kênh data tự chọn có tần số 20Hz và 5 pha khác nhau.(lưu ý tần số sóng mang của bộ giải điều chế phải lớn hơn 10% tần số lấy mẫu của nguồn tín hiệu đầu vào).
- 2) Ở ví dụ của hình 5.20, mô hình ghép kênh đang chia đều time slot cho từng tín hiệu. Dựa trên mã nguồn đã có, phân chia time slot cho hai tín hiệu trên (sine – triangular) theo tỷ lệ 2 : 1 ở phía gửi và tách kênh theo đúng tỷ lệ để cho ra tín hiệu nguồn ban đầu.

TÀI LIỆU THAM KHẢO

- [1] William Stallings. Data and Computer Communications (8th Edition). Prentice-Hall, Inc., USA. 2006.
- [2] MATLAB and Statistics Toolbox Release 2019b, The MathWorks, Inc., Natick, Massachusetts, United States.
- [3] Documentation, S., 2020. Simulation and Model-Based Design, MathWorks. Available at:

<https://www.mathworks.com/products/simulink.html>