HƯỚNG DẪN THỰC HÀNH THÍ NGHIỆM

Môn học: **XỬ LÝ SỐ TÍN HIỆU**

MỤC LỤC

MỞ ĐẦU

Xử lý số tín hiệu là môn học nghiên cứu về các phương trình toán học, các giải thuật và các tính toán dựa trên phương pháp tính gần đúng cho các tín hiệu và hệ thống rời rạc. Nội dung môn học Xử lý số tín hiệu được giảng dạy tại Khoa Điện tử - Viễn thông trường Đại học bách khoa Hà nội, chịu trách nhiệm chính bởi bộ môn Mạch và Xử lý tín hiệu, tập trung vào bao trùm các vấn đề sau:

- Phân tích tín hiệu và hệ thống
- Thiết kế bộ lọc.

Phương pháp học tốt nhất để sinh viên hiểu, nhớ, vận dụng và tự đánh giá được các kiến thức lý thuyết là trực tiếp bắt tay vào giải quyết các bài tập. Để hỗ trợ thêm cho việc nhìn nhận các vấn đề một cách trực quan, đồng thời giúp sinh viên hiểu sâu hơn về lý thuyết của môn học, chúng tôi đã biên soạn phần thực hành này. Phần thực hành bao gồm 2 phần lớn: 1. phân tích tín hiệu số và thiết kế hệ thống xử lý tín hiệu số bằng MATLAB; 2. làm quen với công việc thực hiện phát triển các hệ thống xử lý số tín hiệu bằng bộ xử lý tín hiệu số với tên gọi Digital Signal Processor – DSP.

Hiện nay có rất nhiều các công cụ phần mềm tiện ích rất mạnh để hỗ trợ tính toán. Hai trong số đó là MATHCAD của Mathsoft và MATLAB của MathWorks. Chúng là 2 gói phần mềm có thể dễ dàng kiếm được ở Việt Nam vào thời điểm hiện nay. Ngoài ra, gói phần mềm MATHEMATICA của Wolfram cũng được giới khoa học và kỹ thuật trên thế giới ưa dùng. Khả năng tính toán dựa trên các phương pháp tính gần đúng chính là điểm mạnh của các phần mềm này. Phần mềm MATHCAD có đặc điểm là hiển thị ngay kết quả tính toán sau khi người dùng trực tiếp đánh công thức vào giao diện người sử dụng. Tuy nhiên sử dụng phần mềm này có khó khăn khi người dùng muốn đóng gói rồi kế thừa và tái sử dụng các thiết kế trước đó. Về điểm này phần mềm MATLAB là tương đối mạnh, cho phép người dùng thiết kế phần mềm thông qua các câu lệnh, dễ dàng môđun hoá dưới dạng các kịch bản và các hàm để có thể sử dụng, hoặc phát triển qua các quá trình thiết kế và các bài toán thiết kế khác nhau. Vì lý do đó, MATLAB được lựa chọn cho phần thí nghiệm này.

Tốc độ xử lý nhanh trên các DSP cũng như tính linh hoạt và sự hỗ trợ đầy đủ của các phần mềm phát triển, dùng để khởi tạo các đề án, viết chương trình nguồn, gỡ rối và tối ưu hoá chương trình, của Texas Instrument (TI) đã làm một số lượng lớn các nhà nghiên cứu và phát triển về xử lý tín hiệu số lựa chọn DSP của TI như một công cụ dùng để nghiên cứu và phát triển sản phẩm của mình. Bằng chứng được thể hiện trên sự tăng trưởng của các con số tiêu thụ sản phẩm và thị phần DSP của TI được đăng ở các tạp chí chuyên ngành. Tốc độ xử lý của DSP được cải thiện không ngừng. Vào thời điểm hiện nay, dòng sản phẩm DSP mới nhất của Texas Instrument là TMS320C64xx thậm chí có thể thực hiện với xung đồng hồ lên đến 1GHz, không thua xa lắm so với các bộ vi xử lý mục đích chung thông thường và bù lại về tốc độ xung đồng hồ thì DSP có cấu trúc chuyên biệt cho các chức năng phục vụ xử lý số tín hiệu. Bộ DSP được sử dụng trong bài thí nghiệm là TSM320C50 được nhúng trong bo thí nghiệm của LABVOLT.

Về tổ chức các bài thí nghiệm, thí nghiệm Xử lý số tín hiệu được chia làm 2 bài:

- Bài 1: Mô phỏng hệ thống và tín hiệu rời rạc bằng MATLAB

- Bài 2: Thiết kế bộ lọc số bằng MATLAB
- Bài 3: Giới thiệu về Digital Signal Processor
- Bài 4: Làm quen với bộ thí nghiệm LABVOLT - DSP

Mỗi bài thí nghiệm lại chia làm một số phần. Phần A của bài 1 giới thiệu những đặc điểm chính của MATLAB, giúp sinh viên làm quen với công cụ tiện ích này. Phần B và phần C của bài 1 lần lượt trình bày các yêu cầu làm thí nghiệm để mô phỏng với tín hiệu và hệ thống ở miền thời gian và các miền gián tiếp bao gồm: miền Z, miền ω, và miền k. Phần A và phần B của bài 2 lần lượt trình bày các yêu cầu thí nghiệm để thiết kế bộ lọc FIR và bộ lọc IIR.

Với mỗi phần thí nghiệm được tổ chức theo các mục, lần lượt nêu rõ các yêu cầu về kiến thức cần chuẩn bị trước mối phần, mục đích sinh viên cần đạt được tại mỗi phần, một số lệnh và hàm của MATLAB có thể được sử dụng trong phần đó, các bước cần phải giải quyết trong buổi thí nghiệm và cuối cùng là gợi ý các thực hành có thể mở rộng cho phần này.

Đối với vấn đề làm quen với bộ xử lý tín hiệu số (Digital Signal Processor), bài 3 và bài 4 cũng được chia làm một số mục nhằm làm sinh viên quen dần với phần cứng, việc xử lý bằng phần mềm, đo đạc và đánh giá kết quả trên bo mạch thí nghiệm.

Trong điều kiện cơ sở vật chất của phòng thí nghiệm bộ môn Mạch và Xử lý tín hiệu điện tử, khi thực hành sinh viên có thể chia làm các nhóm từ 3 đến 5 sinh viên cùng nhau giải quyết các bước đưa ra trong mục ***Các bước thực hành*** ở mỗi phần. Chúng tôi cho rằng để hoàn thành tốt mỗi phần thí nghiệm, mỗi sinh viên cần chuẩn bị ở nhà **ít nhất 1 giờ đồng hồ** cho phần thí nghiệm đó. Công việc chuẩn bị có thể bao gồm: Đọc và tổng kết lại các kiến thức lý thuyết trong sách giáo trình, tìm hiểu kỹ yêu cầu, mục đích của bài thí nghiệm, xem lai phần tóm tắt lý thuyết được trình bày trong phần thí nghiệm đó và hình dung các công việc phải làm trong buổi thực hành. Nếu có điều kiện và có máy tính, đồng thời có phần mềm MATLAB sinh viên có thể chuẩn bị trước một số bước sẽ làm trong buổi thí nghiệm.

Đánh giá kết quả của mỗi bài thực hành dựa trên hai tiêu chí: phần thực hành đã hoàn thành và trả lời các câu hỏi được đặt ra bởi các giáo viên hướng dẫn thí nghiệm. Sau buổi thực hành, mỗi nhóm sinh viên cần nộp một báo cáo trong đó trình bày lại các chương trình, các kết quả và các đồ thị theo từng câu hỏi của các phần ***Các bước thực hành***. Tại cuối mỗi buổi thực hành từng sinh viên phải trả lời các câu hỏi do giáo viên hướng dẫn đặt về các vấn đề sau:

- Kiến thức lý thuyết về Xử lý số tín hiệu trong bài thực hành
- Các câu lệnh và hàm của MATLAB sinh viên sử dụng trong bài thực hành.

Phần viết báo cáo được đánh giá với thang điểm tối đa là 40 dành cho tất cả các thành viên trong nhóm, phần trả lời câu hỏi được đánh giá với thang điểm tối đa là 60 dành cho mỗi cá nhân. Nếu đạt được ít nhất 60 điểm của tổng cộng cả hai phần, sinh viên coi như đạt yêu cầu của bài thực hành.

BÀI 1. MÔ PHỎNG HỆ THỐNG VÀ TÍN HIỆU RỜI RẠC BẰNG MATLAB

A. GIỚI THIỆU VỀ MATLAB:

MABLAB, viết tắt của Matrix Labotary, là một công cụ phần mềm hỗ trợ tính toán trên ma trận. MATLAB được tích hợp trên một môi trường chung một loạt các khả năng bao gồm tính toán, hiển thị kết quả và lập trình nhằm giải quyết các vấn đề liên quan đến toán học. Các vấn đề đó bao gồm:

- Các phương trình toán học và tính toán
- Phát triển các giải thuật
- Thu thập dữ liệu
- Mô hình hoá, mô phỏng và tạo các mẫu theo thiết kế
- Phân tích, khảo sát và thể hiện dữ liệu bằng hình ảnh
- Biểu diễn các biểu đồ mang tính khoa học và tính kỹ thuật
- Phát triển các ứng dụng, bao gồm việc phát triển với các giao diện với người sử dụng

Ưu điểm nổi bật của MATLAB, như đã được đề cập ở trên, là khả năng tính toán, đặc biệt là những bài toán liên quan đến ma trận và vector, với thời gian ít hơn nhiều lần so với cùng một công việc tính toán trên các ngôn ngữ lập trình khác như C hay Fortran. Khả năng lập trình của MATLAB cũng rất linh hoạt, cụ thể là trong việc tạo ra các câu lệnh riêng và các hàm của riêng người sử dụng.

Hệ thống MATLAB bao gồm 5 phần chính sau:

- **Môi trường phát triển:** Là một tập hợp các công cụ, phần lớn trong chúng là các giao diện đồ hoạ, giúp người dùng sử dụng các câu lệnh và các hàm của MATLAB.
- **Thư viện các hàm toán học:** Là một tập hợp các hàm toán học bao gồm từ các hàm cơ bản như sin, cosin, các phép tính đại số phức đến các hàm phức tạp như tìm ma trận đảo, tìm ma trận riêng, hàm Bessel và biến đổi Fourier nhanh (Fast Fourier Transform – FFT).
- **Ngôn ngữ lập trình:** Là một ngôn ngữ bậc cao liên quan đến ma trận và mảng. Trong MATLAB có đầy đủ những đặc trưng của một ngôn ngữ lập trình bao gồm các lệnh rẽ nhánh, các hàm, cấu trúc dữ liệu, nhập/xuất dữ liệu, và các đặc tính liên quan đến lập trình hướng đối tượng (object-oriented programming).
- **Đồ hoạ:** Là một tập hợp các công cụ để biểu diễn ma trận và vector bằng đồ hoạ. Bên cạnh các công cụ ở mức thấp để thể hiện dữ liệu dạng 2 chiều và 3 chiều, xử lý hình ảnh tĩnh, ảnh động còn có các công cụ ở mức cao dùng để

tạo ra các biểu diễn đồ hoạ theo ý đồ của người sử dụng cũng như tạo ra các giao diện đồ hoạ người sử dụng.

- **Các API:** Là một thư viện cho phép người sử dụng gọi các hàm viết trên ngôn ngữ C và Fortran. Chúng bao gồm cả các công cụ cho phép gọi các hàm từ MATLAB dưới dạng liên kết động, và để đọc và ghi các tệp .MAT.

MATLAB, bên cạnh khả năng tính toán trên ma trận, đồng thời cũng là một ngôn ngữ lập trình mạnh. Các tệp chương trình của MATLAB được ghi dưới dạng đuôi .m, được gọi là M-files. Có hai loại tệp dạng đuôi .m:

- **Tệp kịch bản (scripts):** Loại tệp này không có các biến đầu vào và đầu ra, nó đơn thuần chỉ xử lý dữ liệu với các biến trên vùng làm việc hiện thời (work space) của MATLAB. Khi gõ tên tệp tại cửa sổ lệnh (command window), các lệnh được lưu trong nội dung của tệp lần lượt được gọi ra theo một kịch bản tuần tự từ trên xuống dưới.
- **Tệp mô tả hàm (functions):** Loại tệp này cần khai báo các biến đầu vào và đầu ra. Các biến được khai bên trong loại tệp này là các biến địa phương (local variables) và chỉ có phạm vi ảnh hưởng tại chính hàm số đó. Nội dung trong các tệp này nhằm mục đích tính toán các thông số đầu ra dựa trên các tham số đầu vào của hàm số. Tên của tệp loại này cần trùng với tên của hàm số được khai báo và mô tả bên trong nội dung của tệp.

Để khởi động MATLAB, người sử dụng có thể nháy đúp chuột vào biểu tượng **MATLAB 6.5** trên màn hình desktop hoặc vào menu **Start -> All Programs -> MATLAB 6.5 -> MATLAB 6.5** từ giao diện của Windows. Sau khi MATLAB được khởi động, trên màn hình người sử dụng sẽ hiển thị lên môi trường phát triển tích hợp của MATLAB bao gồm một số cửa sổ, trong đó có các cửa số quan trọng sau:

- **Cửa sổ lệnh (Command Window):** có chức năng thể hiện dấu nhắc để nhập vào các lệnh từ bàn phím, và hiển thị kết quả tính toán sau khi gõ một lệnh hoặc gọi một hàm.
- **Cửa sổ các lệnh đã dùng (Command History):** thể hiện danh mục các lệnh đã gõ hoặc các hàm đã được gọi theo các phiên làm việc.
- **Cửa sổ thư mục hiện thời (Current Directory):** thể hiện danh sách các tệp dạng đuôi .m đang tồn tại trong thư mục hiện thời. Để thay đổi thư mục hiện thời trên cửa sổ nhỏ nằm ngay bên trên **cửa số lệnh**.
- **Vùng làm việc (Workspace):** thể hiện danh mục tất cả các biến bao gồm: tên biến, giá trị hiện thời của biến, kiểu biến đang tồn tại ở phiên làm việc hiện tại.

Ngoài ra còn một loạt các cửa sổ khác sẽ được kích hoạt và hiển thị khi gọi một lệnh hoặc chọn một mục trong phần Menu của MATLAB. Để biết thêm về các cửa số có thể tham khảo thêm trong **phần trợ giúp (Help)** của MATLAB bằng cách nhấn phím **F1**.

Để soạn thảo một kịch bản hoặc một hàm, thực hiện chọn menu **File -> New -> M-File** hoặc nhắp chuột vào biểu tượng New M-File trên thanh công cụ (Toolbar). Trên

màn hình sẽ hiển thị lên cửa sổ soạn thảo (Editor) có đầy đủ các chức năng soạn thảo giống như bất cứ môi trường soạn thảo của ngôn ngữ lập trình nào khác.

Để xem trợ giúp về một lệnh hay một hàm có sẵn nào đó của MATLAB, gõ lệnh **help** kèm theo tên của lệnh hoặc hàm từ cửa sổ lệnh của MATLAB, ví dụ:

```
>>  help fft
```

trên cửa số lệnh sẽ đưa ra nội dung về chức năng, cú pháp cho các tham số vào/ra cho hàm thực hiện phép biến đổi Fourier nhanh được MATLAB đặt dưới tên **fft**.

B. TÍN HIỆU VÀ HỆ THỐNG RỜI RẠC Ở MIỀN THỜI GIAN RỜI RẠC n

1. Yêu cầu trước khi làm thí nghiệm

Sinh viên nắm vững kiến thức về **"Tín hiệu và hệ thống rời rạc"** bao gồm:

- Các tín hiệu cơ bản
- Hệ thống tuyến tính bất biến và Đáp ứng xung của hệ thống tuyến tính bất biến
- Phương trình sai phân tuyến tính hệ số hằng

2. Mục đích của phần thí nghiệm

Sinh viên dùng MATLAB mô phỏng các nội dung sau:

- Các tín hiệu cơ bản ở miền thời gian
- Tính tích chập
- Đáp ứng của hệ thống được mô tả bởi phương trình sai phân tuyến tính hệ số hằng

3. Tóm tắt lý thuyết

Xử lý số tín hiệu, về bản chất, là tìm hiểu về các phép toán và giải thuật liên quan đến các tín hiệu rời rạc và các hệ thống rời rạc. Các tín hiệu rời rạc thường được thể hiện dưới dạng một dãy số như sau:

$$\{\ldots, x(-3), x(-2), x(-1), x(0), x(1), x(2), x(3), \ldots\}$$

Tuy nhiên, MATLAB chỉ có khả năng biểu diễn một dãy số với độ dài hữu hạn. Khi đó dãy số được khai báo và lưu trữ dưới dạng vector, ví dụ:

```
>> x = [3, 2, -1, 7, -5]
```

Với cách khai báo như vậy, dãy số không thể hiện được chỉ số của các thành phần trong dãy. Vì vậy, để biểu diễn một dãy rời rạc có độ dài hữu hạn, ta cần khởi tạo và lưu trữ chúng dưới dạng 2 vector. Ví dụ:

```
>> n = [-2:2]
>> x = [3, 2, -1, 7, -5]
```

được hiểu là một dãy gồm 5 phần tử xuất phát từ -2 đến 2 có: x(-2)=3, x(-1)=2, x(0)=-1, x(1)=7 và x(2)=-5. Trong tất cả các bài thí nghiệm trên MATLAB của môn học này, chúng ta nên tuân theo một nguyên tắc như vậy.

Định nghĩa một số dãy cơ bản

a. Dãy xung đơn vị:

$$\delta(n)=\begin{cases}1 & n=0\\ 0 & n\neq 0\end{cases}$$

Dãy xung đơn vị trễ (dịch) đi n_0 mẫu:

$$\delta(n-n_0)=\begin{cases}1 & n=n_0\\ 0 & n\neq n_0\end{cases}$$

b. Dãy nhảy đơn vị:

$$u(n)=\begin{cases}1 & n\geq 0\\ 0 & n<0\end{cases}$$

c. Dãy hàm mũ thực:

$$x(n)=a^n,\forall n \quad a\in \mathrm{R}$$

d. Dãy hàm mũ phức:

$$x(n)=e^{(\sigma+j\omega_0)n},\forall n$$

σ là độ suy giảm của tín hiệu, ω_0 là tần số góc tính theo đơn vị radians

e. Dãy lượng giác: Dãy lượng giác là dãy thể hiện tín hiệu có dạng hàm toán học là tổ hợp tuyến tính của các hàm sin và cosin. Một ví dụ về dãy lượng giác như sau:

$$x(n)=\cos(\omega_0 n+\theta),\forall n$$

với θ là pha ban đầu của tín hiệu

f. Dãy ngẫu nhiên: Là dãy mà các phần tử của dãy có giá trị ngẫu nhiên. Sự phân bố ngẫu nhiên có thể được điều chỉnh là phân bố đều hay tuân theo một quy luật phân bố xác suất nào đó. Trong MATLAB có sẵn một số hàm cho phép khởi tạo ra một dãy ngẫu nhiên theo phân bố đều và theo phân bố Gauss.

g. Dãy tuần hoàn: Dãy tuần hoàn là một dãy có giá trị của các phần tử lặp lại tuần hoàn sau một số mẫu nhất định.

$$x(n)=x(n+mN) \quad m\in Z$$

Dãy tuần hoàn thường được ký hiệu là $\tilde{x}(n)$ và được đọc là '*x ngã*'. Chúng ta có thể biểu diễn một dãy với một số chu kỳ tuần hoàn trong MATLAB bằng cách đặt liên tiếp nhau một số hữu hạn các dãy xuất phát từ một dãy có chiều dài hữu hạn. Mỗi dãy này thể hiện một chu kỳ của dãy tuần hoàn.

Một số định nghĩa khác

a. Cộng hai dãy: Dãy thu được có mỗi phần tử là tổng của hai giá trị tương ứng với từng chỉ số của hai dãy ban đầu. Vấn đề đặt ra là đôi khi ta cần mô phỏng trong MATLAB việc tìm dãy tổng của hai dãy có các chỉ số bắt đầu và kết thúc khác nhau. Khi đó với những phần tử của dãy thứ nhất mà tại dãy thứ hai không có phần tử có chỉ số tương ứng, chúng ta cần bổ sung vào dãy thứ hai phần tử có giá trị bằng không. Quá trình đó thực hiện sao cho hai dãy có chỉ số của phần tử đầu và chỉ số của phần tử cuối bằng nhau.

b. Nhân hai dãy: Dãy thu được có mỗi phần tử là tổng của hai giá trị tương ứng với từng chỉ số của hai dãy ban đầu. Tương tự như việc cộng hai dãy, ta cũng cần có quá trình xử lý khi mô phỏng trong MATLAB sao cho hai dãy có chỉ số đầu và chỉ số cuối bằng nhau.

c. Nhân với hằng số: Một dãy đem nhân với hằng số thu được dãy mới có giá trị của từng phần tử bằng giá trị phần tử tương ứng của dãy ban đầu nhân với hằng số.

$$a\{x(n)\} = \{ax(n)\}$$

d. Dịch (Trễ): Làm trễ một dãy đi một khoảng n_0 mẫu thu được dãy mới:

$$\{y(n)\} = \{x(n - n_0)\}$$

hay phần tử thứ m của dãy ban đầu trở thành phần tử thứ $m+n_0$ của dãy mới.

e. Biến số n đảo: Dãy mới thu được là dãy ban đầu được lấy đối xứng qua trục vuông góc với trục biểu diễn chỉ số n tại gốc toạ độ (trục tung)

$$\{y(n)\} = \{x(-n)\}$$

f. Năng lượng: Dãy được tính năng lượng có thể là dãy thực hoặc dãy phức:

$$E_x = \sum_{n=-\infty}^{\infty} x(n)x^*(n) = \sum_{n=-\infty}^{\infty} |x(n)|^2$$

g. Công suất: Công suất trung bình của một dãy tuần hoàn:

$$P_x = \frac{1}{N}\sum_{n=0}^{N-1} |x(n)|^2$$

Hệ thống rời rạc

Trong xử lý tín hiệu, khái niệm hệ thống (system) để chỉ đến một khối, được thể hiện trên hình vẽ bằng một khối chữ nhật trông như một hộp đen có các ký hiệu đầu vào và đầu ra, có chức năng tiếp nhận các tín hiệu từ đầu vào, xử lý chúng và đưa các tín hiệu đã xử lý tới đầu ra. Xử lý số tín hiệu liên quan tới các tín hiệu rời rạc nên các hệ thống được xét đến là các hệ thống rời rạc. Tín hiệu vào được gọi là đầu vào (input) hay kích thích (excitation) của hệ thống. Tín hiệu ra được gọi là đầu ra (output) hay đáp ứng (response) của hệ thống. Trong MATLAB, hệ thống được định chung bởi khái niệm **"filter"**.

Một hệ thống là tuyến tính bất biến (Linear Time-Invariant – LTI) nếu nó hội đủ cả hai tính chất tuyến tính (linearity) và bất biến theo thời gian (time-invariance). Tính chất tuyến tính nói lên rằng đáp ứng của hệ thống với kích thích là một tổ hợp tuyến tính

các tín hiệu rời rạc sẽ bằng với tổ hợp tuyến tính của các đáp ứng, với mỗi đáp ứng này là đầu ra khi cho từng thành phần của đầu vào qua hệ thống. Tính chất bất biến theo thời gian nói lên rằng đáp ứng của hệ thống có dạng giống hệt nhau với cùng một kích thích mà không phụ thuộc vào thời điểm đưa kích thích tới đầu vào. Trong môn học Xử lý số tín hiệu, tất cả các hệ thống được xét tới đều là tuyến tính bất biến.

Một hệ thống tuyến tính bất biến luôn có đáp ứng ra y(n) là tích chập (convolution sum) giữa đầu vào x(n) với dãy đáp ứng xung h(n) của hệ thống, là đáp ứng của hệ thống khi đưa kích thích δ(n) tới đầu vào. Thể hiện tích chập bởi công thức:

$$y(n) = T[x(n)] = x(n) * h(n) = h(n) * x(n) = \sum_{n=-\infty}^{\infty} x(k)h(n-k) = \sum_{n=-\infty}^{\infty} h(k)x(n-k)$$

Một hệ thống là nhân quả nếu đáp ứng ra tại thời điểm hiện tại không phụ thuộc vào kích thích vào tại các thời điểm tương lai. Một hệ thống tuyến tính bất biến là nhân quả nếu đáp ứng xung thoả mãn:

$$h(n) = 0 \quad khi \quad n < 0$$

Một hệ thống là ổn định (Bounded In Bounded Out Stable – BIBO Stable) nếu với một kích thích bị chặn luôn sinh ra một đáp ứng cũng bị chặn, tức là giá trị của đáp ứng ra không tiến đến vô cùng. Một hệ thống tuyến tính bất biến là ổn định nếu đáp ứng xung thoả mãn:

$$\sum_{n=-\infty}^{\infty} |h(n)| < \infty$$

Nói chung, tất cả các hệ thống tuyến tính bất biến có thể thực hiện được, thông qua phần cứng hoặc mô tả phần mềm, đều được mô tả bởi phương trình sai phân tuyến tính hệ số hằng có dạng như sau:

$$\sum_{k=0}^{N} a_k y(n-k) = \sum_{r=0}^{M} b_r x(n-k)$$

hay có thể viết dưới dạng sau thích hợp với thể hiện mô hình sơ đồ khối của hệ thống:

$$y(n) = \sum_{r=0}^{M} b_r x(n-k) - \sum_{k=1}^{N} a_k y(n-k)$$

Các bước để giải phương trình sai phân tuyến tính hệ số hằng đã được trình bày rất cụ thể trong sách giáo trình. Trong MATLAB có hàm **filter** cho phép tìm dãy đáp ứng đầu ra y(n) nếu biết trước các biến đầu vào là các hệ số của phương trình sai phân, dãy a_k và b_r, và kích thích đầu vào x(n). Chúng ta có thể dùng lệnh này để phác hoạ định dạng đầu ra của hệ thống với các tham số nêu trên.

4. Một số lệnh và hàm của MATLAB

Phần này đưa ra danh mục các lệnh các hàm của MATLAB có thể sử dụng trong phần thí nghiệm này. Để biết cụ thể hơn về chức năng của hàm và cú pháp của lệnh gọi hàm, gõ lệnh **help** kèm theo tên của hàm tại cửa sổ lệnh của MATLAB.

`zeros`: tạo một ma trận với toàn bộ các phần tử có giá trị bằng 0.

`ones`: tạo một ma trận với toàn bộ các phần tử có giá trị bằng 1.

`rand`: tạo một ma trận với các phần tử nhận các giá trị ngẫu nhiên được phân bố đều trong khoảng từ 0 đến 1.

`randn`: tạo một ma trận với các phần tử nhận các giá trị ngẫu nhiên theo phân bố Gauss có giá trị trung bình bằng 0, phương sai bằng 1.

`min`: trả về giá trị nhỏ nhất trong một ma trận.

`max`: trả về giá trị lớn nhất trong một ma trận.

`fliplr`: lộn ngược lại thứ tự các phần tử trong một ma trận theo hướng xuất phát từ phải qua trái trở thành từ trái qua phải.

`plot` và `stem`: vẽ đồ thị của một dãy số, plot để thể hiện dạng liên tục, stem để thể hiện dạng rời rạc, thường sử dụng hàm stem để vẽ tín hiệu ở miền n.

`conv`: trả về tích chập của 2 vector.

`filter`: trả về đáp ứng theo thời gian của hệ thống được mô tả bởi một phương trình sai phân tuyến tính hệ số hằng.

Ngoài ra, sinh viên cần tìm hiểu một cách rất cẩn thận các phép toán trên ma trận và vector trong phần trợ giúp (Help) của MATLAB bằng cách nhấn F1 rồi vào mục **MATLAB -> Getting Started -> Matrices and Arrays**.

5. Các bước thực hành

1.1. Tạo các dãy xung đơn vị và dãy nhảy đơn vị theo chương trình mẫu bằng cách gõ các dòng lệnh cho ở 2 bảng dưới đây vào cửa sổ soạn thảo (Editor) và ghi lại theo các tên tệp lần lượt là **impseq.m** và **stepseq.m**:

Dãy xung đơn vị:

```
function [x,n] = impseq(n0,n1,n2)
%Tao ra day x(n) = delta(n-n0); n1 <= n <= n2
%---------------------------------------------
%[x,n] = impseq(n0,n1,n2)
n = [n1:n2]; x = [(n-n0) == 0];
```

Dãy nhảy đơn vị:

```
function [x,n] = stepseq(n0,n1,n2)
%Tao ra day x(n) = u(n-n0); n1 <= n <= n2
%---------------------------------------------
%[x,n] = stepseq(n0,n1,n2)
n = [n1:n2]; x = [(n-n0) >= 0];
```

1.2. Viết chương trình tạo dãy hàm mũ thực với các tham số đầu vào và đầu ra được nhập theo câu lệnh:

```
[x,n] = expseq(a,n1,n2)
```

Chú ý: tham số a có thể thực hoặc phức

1.3. Viết chương trình tạo một dãy thực ngẫu nhiên xuất phát từ n1 đến n2 và có giá trị của biên độ theo phân bố Gauss với trung bình bằng 0, phương sai bằng 1. Các tham số đầu vào và đầu ra được nhập theo câu lệnh:

```
[x,n] = randnseq(n1,n2)
```

1.4. Tạo các hàm cộng 2 dãy và nhân 2 dãy với các chỉ số đầu và chỉ số cuối của hai dãy tương ứng khác nhau, hàm tạo trễ và hàm biến số n đảo theo chương trình mẫu bằng cách gõ các dòng lệnh cho ở 4 bảng dưới đây vào cửa sổ soạn thảo (Editor) và ghi lại theo các tên tệp lần lượt là **sigadd.m**, **sigmult.m**, **sigshift.m**, và **sigfold.m**:

Cộng 2 dãy:

```
function [y,n] = sigadd(x1,n1,x2,n2)
%Thuc hien y(n) = x1(n)+x2(n)
%-------------------------------------------
%[y,n] = sigadd(x1,n1,x2,n2)
% y = day tong co vector chi so n
%x1 = day thu nhat co vector chi so n1
%x2 = day thu hai co vector chi so n2 (n2 co the khac n1)
n = min(min(n1),min(n2)):max(max(n1),max(n2));
y1 = zeros(1,length(n)); y2 = y1;
y1(find((n>=min(n1))&(n<=max(n1))==1)) = x1;
y2(find((n>=min(n2))&(n<=max(n2))==1)) = x2;
y = y1+y2;
```

Nhân 2 dãy:

```
function [y,n] = sigmult(x1,n1,x2,n2)
%Thuc hien y(n) = x1(n)*x2(n)
%-------------------------------------------
% y = day tich co vector chi so n
%x1 = day thu nhat co vector chi so n1
%x2 = day thu hai co vector chi so n2 (n2 co the khac n1)
n = min(min(n1),min(n2)):max(max(n1),max(n2));
y1 = zeros(1,length(n)); y2 = y1;
y1(find((n>=min(n1))&(n<=max(n1))==1)) = x1;
y2(find((n>=min(n2))&(n<=max(n2))==1)) = x2;
y = y1.*y2;
```

Trễ (dịch):

```
function [y,n] = sigshift(x,m,n0)
%Thuc hien y(n) = x(n-n0)
%-------------------------------------------
%[y,n] = sigshift(x,m,n0)
n = m + n0; y = x;
```

Biến số n đảo:

```
function [y,n] = sigfold(x,n)
%Thuc hien y(n) = x(-n)
%-------------------------------------------
%[y,n] = sigfold(x,n)
y = fliplr(x); n = -fliplr(n);
```

1.5. Viết chương trình tạo hàm năng lượng của một dãy với các tham số đầu vào và đầu ra được nhập vào theo câu lệnh:

```
Ex = energy(x,n);
```

1.6. Thể hiện trên đồ thị dãy $x(n) = 2\delta(n+2) - 2\delta(n-4)$, $-5 \le n \le 5$ theo chương trình mẫu bằng cách gõ các dòng lệnh theo bảng dưới đây vào cửa sổ soạn thảo (Editor) và ghi lại theo tên tệp **Solution_1_6**.

```
n = [-5:5];
x = 2*impseq(-2,-5,5) - impseq(4,-5,5);
stem(n,x);
title('Day so theo dau bai 1.5');
xlabel('n'); ylabel('x(n)');
```

Gõ **Solution_1_6** tại cửa sổ lệnh của MATLAB để chạy kịch bản nói trên và xem đồ thị của dãy số.

1.7. Viết chương trình thể hiện trên đồ thị các dãy sau đây:

a. $x(n) = n[u(n) - u(n-10)] + 10\left(\frac{1}{3}\right)^{n-10}[u(n-10) - u(n-20)] \quad ,0 \le n \le 20$

b. $x(n) = \cos(0.04\pi n) + 0.2w(n) \quad ,0 \le n \le 20$, với w(n) là hàm có giá trị ngẫu nhiên theo phân bố Gauss, trung bình bằng 0, phương sai bằng 1

Sau đó tính năng lượng của từng dãy.

1.8. Thể hiện trên đồ thị 4 chu kỳ của dãy tuần hoàn với chu kỳ N=5 $\tilde{x}(n) = \{...,5,4,3,2,1,\underset{\uparrow}{5},4,3,2,1,5,4,3,2,1,...\} \quad ,-10 \le n \le 9$ theo chương trình mẫu bằng cách gõ các dòng lệnh theo bảng dưới đây vào cửa sổ soạn thảo (Editor) và ghi lại theo tên tệp **Solution_1_8**.

```
n = [-10:9]; x = [5 4 3 2 1];
P = 4;
xtilde = x'*ones(1,P)
xtilde = xtilde(:)'
stem(n,xtilde);title('Day so theo dau bai 1.8');
xlabel('n'); ylabel('xtilde(n)');
```

Gõ lệnh **Solution_1_8** tại cửa sổ lệnh của MATLAB để chạy kịch bản nói trên và xem đồ thị của dãy số.

Tính công suất trung bình của dãy đã cho ở trên.

1.9. Cho dãy $x(n) = \{1,2,3,4,5,\underset{\uparrow}{6},7,6,5,4,3,2,1\} \quad -2 \le n \le 10$. Viết chương trình thể trên đồ thị các dãy sau đây:

a. $x_1(n) = 2x(n-5) - 3x(n+4)$

b. $x_2(n) = x(3-n) - x(n)x(n-2)$

1.10. Trong MATLAB có hàm **conv** thực hiện trả về một dãy là kết quả của phép tính tích chập giữa 2 dãy được cho theo tham số đầu vào của hàm **conv**. Tuy nhiên, các dãy đầu vào và đầu ra cũng như dãy kết quả đều không nói lên chỉ số bắt đầu và chỉ số kết thúc của dãy mà chỉ được ngầm hiểu là các dãy được bắt đầu từ chỉ số 0. Tạo hàm tính tích chập có tên **conv_m** thực hiện việc tính tích chập của hai dãy, mà mỗi dãy được thể hiện bởi 2 vector, một vector thể hiện chỉ số, một vector thể hiện giá trị của dãy, giống như các dãy được biểu diễn ở các bước tiến hành trước bằng cách gõ các dòng lệnh theo bảng dưới đây vào cửa sổ soạn thảo (Editor) và ghi lại theo tên tệp **conv_m.m**.

```
function [y,ny] = conv_m(x,nx,h,nh)
%Ham tinh tich chap da duoc sua doi danh cho
%xu ly so tin hieu
%-----------------------------------------------
%[y,ny] = conv_m(x,nx,h,nh)
%[y,ny] = day ket qua
%[x,nx] = day thu nhat
%[h,nh] = day thu hai
%
nyb = nx(1)+nh(1); nye = nx(length(x))+nh(length(h));
ny = [nyb:nye];
y = conv(x,h);
```

1.11. Viết chương trình thể hiện trên đồ thị kết quả phép tính tích chập giữa 2 dãy sau:

$$x(n) = rect_6(n)$$

$$h(n) = \begin{cases} 1 - \dfrac{n}{4} & 0 \le n < 4 \\ 0 & n\ còn\ lại \end{cases}$$

với $-4 \le n \le 10$

1.12. Viết chương trình thể hiện trên đồ thị kết quả hàm tự tương quan của dãy sau:

$$x(n) = \{6,8,2,\underset{\uparrow}{-5},4,-7,1\} \quad -3 \le n \le 3$$

1.13. Cho hệ thống được mô tả bởi phương trình sai phân tuyến tính hệ số hằng như sau:

$$y(n) - y(n-1) + 0.9y(n-2) = x(n)$$

Viết chương trình sử dụng hàm **filter** của MATLAB thực hiện các công việc sau:

a. Biểu diễn bằng đồ thị hàm đáp ứng xung đơn vị của hệ thống với $-20 \le n \le 100$

b. Biểu diễn bằng đồ thị dãy đáp ứng của hệ thống với $-20 \le n \le 100$ khi dãy đầu vào là dãy nhảy đơn vị.

6. Mở rộng

Xem xét việc giải phương trình vi phân tuyến tính hệ số hằng với các điều kiện đầu cho trước. Gợi ý: dựa trên hàm **filter** và **filtic** nằm trong bộ công cụ Signal Processing Toolbox.

C. TÍN HIỆU VÀ HỆ THỐNG RỜI RẠC Ở MIỀN Z, MIỀN TẦN SỐ LIÊN TỤC ω, VÀ MIỀN TẦN SỐ RỜI RẠC k

1. Yêu cầu trước khi làm thí nghiệm

Sinh viên nắm vững kiến thức về các phép biến đổi trong xử lý số tín hiệu, và ứng dụng của các phép biến đổi đó trong việc biểu diễn các hệ thống và tín hiệu một cách gián tiếp ở các miền khác nhau bao gồm: biểu diễn hệ thống và tín hiệu rời rạc trong miền Z, biểu diễn hệ thống và tín hiệu rời rạc trong miền tần số liên tục (miền ω), biểu diễn tín hiệu rời rạc trong miền tần số rời rạc (miền k).

2. Mục đích của phần thí nghiệm

Sinh viên dùng MATLAB mô phỏng các nội dung sau:

- Biểu diễn bằng đồ thị hàm phổ biên độ và phổ pha của một dãy tín hiệu khi biết trước hàm ảnh qua phép biến đổi Fourier của hàm số đó
- Viết chương trình tính gần đúng và biểu diễn bằng đồ thị biến đổi Fourier của một dãy có chiều dài hữu hạn
- Biểu diễn bằng đồ thị phân bố các điểm cực và điểm không của một hệ thống
- Biểu diễn bằng đồ thị hàm đáp ứng tần số của một hệ thống
- Biểu diễn bằng đồ thị ảnh của phép biến đổi Fourier rời rạc của một dãy có chiều dài hữu hạn
- Đánh giá hiệu quả của thuật toán biến đổi Fourier nhanh với chiều dài dãy thay đổi.

3. Tóm tắt lý thuyết

Tất cả các hệ thống được xét đến trong môn học Xử lý số tín hiệu đều là Hệ thống tuyến tính bất biến. Điều đó có nghĩa khi kích thích đầu vào của một hệ thống là tổ hợp tuyến tính của các thành phần tín hiệu khác nhau thì đầu ra là tổ hợp tuyến tính của các đáp ứng khi cho từng tín hiệu thành phần qua hệ thống. Việc xem xét quy luật của tín hiệu và hệ thống đối với các tín hiệu thành phần cơ bản thông thường là dễ dàng hơn khi xem xét tổng thể tín hiệu ban đầu.

Có một số cách thức để phân tích một tín hiệu thành tổ hợp tuyến tính của các tín hiệu thành phần. Trong những cách đó, lựa chọn tín hiệu thành phần là các hàm xung đơn vị tại các thời điểm khác nhau là một ví dụ điển hình. Một hệ thống tương đương với toán tử T tác động lên dãy x(n) tại đầu vào sẽ có dãy đáp ứng ra y(n) là:

$$y(n) = T[x(n)] = T\left[\sum_{k=-\infty}^{\infty} x(k)\delta(n-k)\right] = \sum_{k=-\infty}^{\infty} x(k)T[\delta(n-k)] = \sum_{k=-\infty}^{\infty} x(k)h(n-k) = x(n) * h(n)$$

với $h(n) = \mathrm{T}[\delta(n)]$ - đáp ứng xung của hệ thống.

Đối với hệ thống tuyến tính bất biến, một cách phân tích đã được tiêu chuẩn hoá và rất hữu ích trong việc xét đến hầu hết các tín hiệu và hệ thống đó là phân tích các tín

hiệu thành tổ hợp tuyến tính của các tín hiệu thành phần mà mỗi tín hiệu thành phần là các hàm lượng giác ($e^{j\omega t}$ - với ω là các giá trị tần số khác nhau). Công cụ để thực hiện việc phân tích trên là biến đổi Fourier, một phép biến đổi biến một dãy số rời rạc theo thời gian thành một hàm số phức với biến số thực liên tục, tuần hoàn ở miền tần số.

Phép biến đổi Fourier cho dãy số x(n), với x(n) thoả mãn điều kiện $\sum_{n=-\infty}^{\infty}|x(n)| < \infty$:

$$X(e^{j\omega}) = FT[x(n)] = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n}$$

Biến đổi Fourier ngược đối với hàm $X(e^{j\omega})$:

$$x(n) = IFT[X(e^{j\omega})] = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})e^{-j\omega n}d\omega$$

$X(e^{j\omega})$ là hàm phức với biến số thực nên nó thường được thể hiện bởi 2 thành phần phổ biên độ và phổ pha dưới dạng sau đây:

$$X(e^{j\omega}) = |X(e^{j\omega})|e^{j\arg[X(e^{j\omega})]} = |X(e^{j\omega})|e^{j\varphi(\omega)}$$

- $|X(e^{j\omega})|$: Là phổ biên độ của tín hiệu x(n)
- $\arg[X(e^{j\omega})] = \varphi(\omega)$: Là phổ pha của tín hiệu x(n)

Khi quan tâm đến các thành phần tần số của một tín hiệu, ta cần quan tâm đến hàm phổ biên độ và hàm phổ pha của tín hiệu đó đối với các tần số. Có hai điểm cần lưu ý đối với biểu diễn tín hiệu ở miền tần số:

- Do x(n) là rời rạc nên $X(e^{j\omega})$ là hàm tuần hoàn chu kỳ 2π theo biến số ω.
- Do tính chất đối xứng của phép biến đổi Fourier nên nếu dãy x(n) là thực thì hàm $X(e^{j\omega})$ có tính chất đối xứng Hermit (Hermitian Symmetric), điều này có nghĩa phổ biên độ là một hàm thực chẵn và phổ pha là một hàm thực lẻ.

Hai tính chất trên nói lên rằng nếu x(n) là một dãy tín hiệu thực thì chỉ cần khảo sát hàm $X(e^{j\omega})$ trong phạm vi $0 \le \omega \le \pi$ là đã có đầy đủ thông tin về toàn bộ hàm $X(e^{j\omega})$ với $-\infty \le \omega \le \infty$. Trên thực tế khi xem xét đồ thì phổ biên độ và phổ pha của tín hiệu, chúng ta thường thể hiện đồ thị trong một vài chu kỳ tuần hoàn.

MATLAB, cũng như mọi phần mềm hỗ trợ tính toán và các ngôn ngữ lập trình khác không có khả năng tính toán trực tiếp cũng như thể hiện một hàm số với biến số liên tục biến thiên từ -∞ đến ∞. Điều này có nghĩa MATLAB không thể trực tiếp tính $X(e^{j\omega})$ từ x(n). Tuy nhiên, nếu biết được biểu thức của hàm ảnh của tín hiệu qua phép biến đổi Fourier (hàm phổ của tín hiệu), ta có thể tính các giá trị của hàm phổ tín hiệu tại các điểm rời rạc trong một khoảng nào đó và thể hiện gần đúng trên đồ thị phổ biên độ và phổ pha của tín hiệu gốc.

Trong trường hợp x(n) là một dãy có chiều dài hữu hạn, ta có thể tính gần đúng $X(e^{j\omega})$ tại M+1 giá trị gần đúng trong khoảng $[0,\pi]$ theo nguyên tắc sau:

- Do x(n) chỉ xác định trong một khoảng hữu hạn $0 \le n \le N-1$ nên:

$$X\left(e^{j\omega}\right) = FT\left[x(n)\right] = \sum_{n=0}^{N-1} x(n) e^{-j\omega n}$$

- Khi lấy M+1 điểm rời rạc cách đều nhau trong khoảng [0,π], biến liên tục ω trở thành biến rời rạc ω_k với $\omega_k = \frac{\pi}{M} k \quad , k = 0,1,...,M$

- Giá trị của $X(e^{j\omega})$ tại các điểm rời rạc là: $X\left(e^{j\omega_k}\right) = \sum_{n=-\infty}^{\infty} x(n) e^{-j\frac{\pi}{M}kn}$

- Công thức trên có thể viết dưới dạng phương trình ma trận như sau:

$$\begin{bmatrix} X\left(e^{j\omega_0}\right) \\ X\left(e^{j\omega_1}\right) \\ \vdots \\ X\left(e^{j\omega_M}\right) \end{bmatrix} = \begin{bmatrix} e^{-j\frac{\pi}{M}00} & e^{-j\frac{\pi}{M}01} & \cdots & e^{-j\frac{\pi}{M}0(N-1)} \\ e^{-j\frac{\pi}{M}10} & e^{-j\frac{\pi}{M}11} & & e^{-j\frac{\pi}{M}1(N-1)} \\ \vdots & & & \\ e^{-j\frac{\pi}{M}M0} & e^{-j\frac{\pi}{M}M1} & & e^{-j\frac{\pi}{M}M(N-1)} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix}$$

lấy chuyển vị của cả hai vế, phương trình trên trở thành

$$\begin{bmatrix} X(0) & X(1) & \cdots & X(M) \end{bmatrix} = \begin{bmatrix} x(0) & x(1) & \cdots & x(N-1) \end{bmatrix} \begin{bmatrix} e^{-j\frac{\pi}{M}00} & e^{-j\frac{\pi}{M}10} & \cdots & e^{-j\frac{\pi}{M}(N-1)0} \\ e^{-j\frac{\pi}{M}01} & e^{-j\frac{\pi}{M}11} & \cdots & e^{-j\frac{\pi}{M}(N-11)} \\ \vdots & \vdots & & \vdots \\ e^{-j\frac{\pi}{M}0(N-1)} & e^{-j\frac{\pi}{M}1(N-1)} & \cdots & e^{-j\frac{\pi}{M}M(N-1)} \end{bmatrix}$$

Đoạn chương trình sau nhằm thực hiện việc tính giá trị của hàm $X(e^{j\omega})$ của dãy x(n) có chiều dài hữu hạn từ n_1 đến n_2 với M+1 giá trị rời rạc trong khoảng [0,π]:

```
>> k = [0:M]; n=[n1:n2];
>> X = x * exp(-j*pi/M) .^ (n'*k);
```

Dù cho việc phân tích tín hiệu và hệ thống bằng phép biến đổi Fourier là thuận tiện và rất hữu ích trong rất nhiều trường hợp, công cụ này đôi khi cũng gặp một số khó khăn:

- Một số dãy tín hiệu trong thực tế ví dụ như u(n) và nu(n) là không có biến đổi Fourier, dẫn đến không phân tích được các thành phần tần số của tín hiệu.
- Đáp ứng của hệ thống trong thời gian quá độ gây bởi điều kiện đầu của hệ thống hoặc đột ngột thay đổi dạng tín hiệu dãy đầu vào là không khảo sát được bằng biến đổi Fourier.

Phép biến đổi Z cho phép chúng ta có thể giải quyết được bài toán trong các trường hợp như vậy. Định nghĩa phép biến đổi Z cho dãy số x(n) là:

$$X(z) = ZT[x(n)] = \sum_{n=-\infty}^{\infty} x(n) z^{-n}$$

X(z) là một hàm phức với biến số (độc lập) phức. Tập các giá trị z để chuỗi hàm bên tay phải của biểu thức trên hội tụ về một hàm số, hay nói một cách khác để X(z) tồn tại gọi là miền hội tụ RC (Region of Convergence) của biến đổi Z. Có thể chứng tỏ được rằng, trong trường hợp tổng quát miền hội tụ của biến đổi Z của một dãy số nằm bên trong một hình vành khuyên $R_{x-} < z < R_{x+}$, với R_{x-} và R_{x+} là các số thực dương.

Biến đổi Z ngược đối với hàm X(z):

$$x(n) = ZT[X(z)] = \frac{1}{2\pi} \oint_C X(z) z^{-n} dz$$

với C là một đường cong kín lấy theo chiều ngược chiều kim đồng hồ, bao quanh gốc toạ độ và nằm hoàn toàn trong miền hội tụ của X(z) (RC[X(z)]).

Trên thực tế, phương pháp được sử dụng trong hầu hết các trường hợp tìm biến đổi Z ngược của một hàm phân thức hữu tỷ X(z) là phân tích thành tổng của các phân thức đơn giản. Hàm **residuez** của MATLAB cho phép nhanh chóng tìm ra các điểm cực và các hệ số trong khai triển ứng với các điểm cực đó của một hàm phân thức hữu tỷ X(z).

Trong trường hợp đường tròn đơn vị nằm trong miền hội tụ của biến đổi Z thì biến đổi Fourier chính là biến đổi Z đánh giá trên đường tròn đơn vị.

Đối với một hệ thống, hàm truyền đạt H(z) của hệ thống được định nghĩa là biến đổi Z của hàm đáp ứng xung:

$$H(z) = ZT[h(n)] = \sum_{n=-\infty}^{\infty} h(n) z^{-n}$$

Hàm truyền đạt của hệ thống chính là tỷ số giữa biến đổi Z của tín hiệu đầu ra trên biến đổi Z của tín hiệu đầu vào:

$$H(z) = \frac{Y(z)}{X(z)}$$

Như ở phần trước đã đề cập tất cả các hệ thống tuyến tính bất biến có thể thực hiện được đều được mô tả bởi phương trình sai phân tuyến tính hệ số hằng. Các hệ thống này có ảnh của đáp ứng xung qua phép biến đổi Z đều có dạng phân thức hữu tỷ mà tử số và mẫu số là các đa thức theo z (hoặc z^{-1}). Các điểm không, tại đó giá trị của X(z) bằng 0, chính là các nghiệm của tử số. Các điểm cực, tại đó giá trị của X(z) tiến tới vô cùng, chính là các nghiệm của mẫu số. Sự phân bố các điểm cực và điểm không của biến đổi Z đối với một tín hiệu, hoặc hàm truyền đạt của hệ thống, quyết định đến toàn bộ các tính chất của tín hiệu hay hệ thống được xét đến. Vì vậy, xem xét phân bố điểm cực và điểm không của một hàm X(z) cũng là một nội dung cần được thực hiện trong phần thí nghiệm này bằng hàm **zplane** của MATLAB.

Hai phép biến đổi nói trên, biến đổi Fourier và biến đổi Z, về bản chất là biến đổi một dãy số trở thành một hàm phức với biến số thực, đối với biến đổi Fourier, và một hàm phức với biến số phức, đối với biến đổi Z. Các miền mới được xét đến là miền ω và miền Z. Đặc điểm chung của các hàm số trên hai miền mới là hàm số với biến số liên tục, do đó, MATLAB cũng như tất cả các ngôn ngữ lập trình và công cụ phần mềm hỗ trợ bằng máy tính không thể tính toán chính xác toàn bộ hàm số ảnh của các phép biến đổi nói trên, thay vì đó ta chỉ thu được kết quả gần đúng tại các điểm rời rạc.

Biến đổi Fourier rời rạc, ứng dụng trên dãy tuần hoàn và dãy có chiều dài hữu hạn là phép biến đổi cho phép máy tính tìm được chính xác mọi giá trị của hàm ảnh của phép biến đổi tại tất cả các biến của hàm số bởi hàm ảnh là hàm trên miền rời rạc, miền này gọi là miền k. Công thức biến đổi Fourier rời rạc cho một dãy số x(n) có chiều dài hữu hạn hữu hạn từ 0 đến N-1 được cho như sau:

$$X(k)_N = DFT[x(n)_N] = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1} x(n) \mathrm{W}_N^{kn}$$

Từ N giá trị rời rạc của dãy số X(k), ta hoàn toàn có thể xây dựng lại được dãy gốc x(n) ban đầu. Công thức biến đổi Fourier rời rạc ngược đối với dãy $X(k)_N$ là:

$$x(n)_N = IDFT[X(k)_N] = \frac{1}{N}\sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi}{N}kn} = \frac{1}{N}\sum_{k=0}^{N-1} X(k) \mathrm{W}_N^{-kn}$$

với $\mathrm{W_N} = \mathrm{e}^{-\mathrm{j}\frac{2\pi}{\mathrm{N}}}$, dẫn đến $\mathrm{W}_N^{kn} = \mathrm{e}^{-\mathrm{j}\frac{2\pi}{\mathrm{N}}kn}$, $\mathrm{W}_N^{-kn} = \mathrm{e}^{\mathrm{j}\frac{2\pi}{\mathrm{N}}kn}$, x(n) và X(k) chỉ khác 0 trong khoảng từ 0 đến N-1.

Dưới dạng ma trận các công thức trên được thể hiện:

$$[X] = [W_N][x] \text{ và } [x] = [W_N]^{-1}[X] = \frac{1}{N}[W_N]^*[X]$$

với X, x, và $\mathrm{W_N}$ là các vector và ma trận được định nghĩa:

$$[X] = \begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ X(N-1) \end{bmatrix}, [\mathrm{W_N}] = \begin{bmatrix} \mathrm{W_N}^{00} & \mathrm{W_N}^{01} & \cdots & \mathrm{W_N}^{0(N-1)} \\ \mathrm{W_N}^{10} & \mathrm{W_N}^{11} & \cdots & \mathrm{W_N}^{1(N-1)} \\ \vdots & \vdots & & \vdots \\ \mathrm{W_N}^{(N-1)0} & \mathrm{W_N}^{(N-1)1} & \cdots & \mathrm{W_N}^{(N-1)(N-1)} \end{bmatrix}, \text{ và } [x] = \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix}$$

Chúng ta hoàn toàn có thể xây dựng thuật toán biến đổi Fourier rời rạc thuận và và ngược một cách trực tiếp xây dựng từ công thức nhân ma trận trên, giống như thuật toán tính gần đúng của biến đổi Fourier đã được đề cập đến ở đầu phần tóm tắt lý thuyết này. Tuy nhiên, số phép tính để tính toán là rất lớn, tương đương với NxN phép nhân trên số phức và N(N-1) phép cộng trên số phức cho biến đổi Fourier rời rạc đối với dãy có độ dài là N mẫu. Năm 1965, Cooley và Turkey đã đưa ra một thuật toán rút gọn số lượng phép tính trong biến đổi Fourier đi rất nhiều. Thuật toán này được biết đến với tên gọi biến đổi Fourier nhanh (Fast Fourier Transform – FFT). Tư tưởng của thuật toán này cũng có thể áp dụng cho phép tính biến đổi Fourier gần đúng trên M+1 điểm rời rạc trong khoảng $[0,\pi]$.

Hàm **fft** của MATLAB cho phép thực hiện việc biến đổi Fourier rời rạc theo thuật toán biến đổi Fourier nhanh. Hàm **fft** được viết bằng ngôn ngữ máy chứ không phải bằng ngôn ngữ MATLAB nên nó quá trình thực hiện biến đổi Fourier rời rạc được tiến hành rất nhanh. Nếu N là luỹ thừa của 2, hàm **fft** sẽ giải quyết bài toán theo thuật toán cơ số 2. Nếu N không phải là luỹ thừa của 2, hàm **fft** tách N thành tích của các thừa số nguyên tố và thuật toán cơ số hỗn hợp được áp dụng trong trường hợp này. Cuối cùng, khi N là một số nguyên tố, hàm **fft** sẽ suy giảm về thuật toán biến đổi Fourier rời rạc dạng nguyên thể theo đúng như công thức của định nghĩa ở trên. Hàm **ifft** thực hiện quá trình ngược lại, biến đổi Fourier ngược. Đánh giá tốc độ thời gian tính toán của hàm **fft** là một trong những nội dung thực hành của phần này.

4. Một số lệnh và hàm của MATLAB

Phần này đưa ra danh mục các lệnh các hàm của MATLAB có thể sử dụng trong phần thí nghiệm này. Để biết cụ thể hơn về chức năng của hàm và cú pháp của lệnh gọi hàm, gõ lệnh **help** kèm theo tên của hàm tại cửa số lệnh của MATLAB.

`abs, angle`: trả về các hàm thể hiện Mođun và Agumen của một số phức

`real, imag`: trả về các hàm thể hiện phần thực và phần ảo của một số phức

`residuez`: trả về các điểm cực và các hệ số tương ứng với các điểm cực đó trong phân tích một hàm phân thức hữu tỷ ở miền Z thành các thành phần là các hàm phân thức đơn giản, ngược lại nếu đầu vào là danh sách các điểm cực và các hệ số, hàm residuez sẽ trả về hàm phân thức hữu tỷ ở miền Z

`poly`: xây dựng một đa thức từ danh sách các nghiệm của nó

`ztrans`: trả về biến đổi Z của một hàm số được định nghĩa theo công thức của một biểu tượng (symbol)

`iztrans`: hàm ngược lại của hàm `ztrans`

`zplane`: thể hiển phân bố điểm cực và điểm không của một hàm phân thức hữu tỷ lên mặt phẳng Z

`freqz`: trả về đáp ứng tần số của một hệ thống tại một số hữu hạn các điểm rời rạc trên vòng tròn đơn vị khi biết hàm truyền đạt của nó

`fft`: thực hiện biến đổi Fourier rời rạc của một dãy số có độ dài hữu hạn theo thuật toán biến đổi Fourier nhanh và trả về kết quả biến đổi Fourier rời rạc của dãy số đó

`clock:` trả về thời gian thực hiện tại

`etime:` trả về thời gian tính bằng giây giữa 2 thời điểm.

5. Các bước thực hành

1.14. Cho dãy $x(n) = 0{,}5^n u(n)$

a. Dựa trên định nghĩa của biến đổi Z, tìm biến đổi Z của dãy trên

b. Kiểm chứng lại kết quả câu a bằng hàm `ztrans`

c. Từ kết quả trên, tìm biến đổi Fourier của x(n)

d. Dùng MATLAB thể hiện trên đồ thị phổ $X(e^{j\omega})$ tại 501 điểm rời rạc trong khoảng $[0,\pi]$ theo chương trình mẫu bằng cách gõ các dòng lệnh theo bảng dưới đây vào cửa sổ soạn thảo (Editor) và ghi lại theo tên tệp **Solution_1_14**.

```
w = [0:1:500]*pi/500;
X = exp(j*w) ./ (exp(j*w)- 0.5*ones(1,501));
magX = abs(X); angX = angle(X);
realX = real(X); imagX = imag(X);
%
subplot(2,2,1); plot(w/pi,magX); grid;
title('Magnitude Part'); xlabel('frequency in pi units');
ylabel('Magnitude');
subplot(2,2,3); plot(w/pi,angX); grid;
title('Angle Part'); xlabel('frequency in pi units');
ylabel('Radians');
subplot(2,2,2); plot(w/pi,realX); grid;
title('Real Part'); xlabel('frequency in pi units');
ylabel('Real');
subplot(2,2,4); plot(w/pi,imagX); grid;
title('Imaginary Part'); xlabel('frequency in pi units');
ylabel('Imaginary');
```

Gõ lệnh **Solution_1_14** tại cửa sổ lệnh của MATLAB để chạy kịch bản nói trên và xem các đồ thị.

1.15. Cho phổ $X(e^{j\omega})$ có dạng sau:

$$X\left(e^{j\omega}\right) = e^{-j\frac{\omega}{2}} \sin 3\omega$$

Viết chương trình thể hiện trên đồ thị các hàm phổ biên độ, phổ pha, phần thực và phần ảo của $X(e^{j\omega})$, tính tại 2001 điểm rời rạc trong khoảng $[-2\pi,2\pi]$.

1.16. Cho dãy x(n) có dạng như sau:

$$x(n) = \left\{..,0,0,1,\underset{\uparrow}{2},3,4,5,0,0,...\right\}$$

Đây là một dãy số xác định trong một khoảng hữu hạn từ -1 đến 3.

Tính và thể hiện phổ của dãy x(n) tại 501 điểm rời rạc trong khoảng $[0,\pi]$ theo chương trình mẫu bằng cách gõ các dòng lệnh theo bảng dưới đây vào cửa sổ soạn thảo (Editor) và ghi lại theo tên tệp **Solution_1_16**.

```
n = -1:3; x = 1:5;
k = 0:500; w = (pi/500)*k;
X = x*(exp(-j*pi/500)).^(n'*k);
magX = abs(X); angX = angle(X);
realX = real(X); imagX = imag(X);
%
subplot(2,2,1); plot(k/500,magX); grid;
title('Magnitude Part'); xlabel('frequency in pi units');
ylabel('Magnitude');
subplot(2,2,3); plot(k/500,angX); grid;
title('Angle Part'); xlabel('frequency in pi units');
ylabel('Radians');
subplot(2,2,2); plot(k/500,realX); grid;
title('Real Part'); xlabel('frequency in pi units');
ylabel('Real');
subplot(2,2,4); plot(k/500,imagX); grid;
title('Imaginary Part'); xlabel('frequency in pi units');
ylabel('Imaginary');
```

Gõ lệnh **Solution_1_16** tại cửa sổ lệnh của MATLAB để chạy kịch bản nói trên và xem các đồ thị.

1.17. Cho dãy $x(n) = rect_7(n)$

Viết chương trình tính và thể hiện phổ của dãy x(n) tại 501 điểm rời rạc trong khoảng $[0,\pi]$ tương tự như bài 1.16.

1.18. Một hàm ở miền Z được cho với công thức sau đây:

$$X(z) = \frac{z}{3z^2 - 4z + 1}$$

Hàm số X(z) có thể viết dưới dạng tỷ số của hai đa thức theo z^{-1} như sau

$$X(z) = \frac{z}{3z^2 - 4z + 1} = \frac{z^{-1}}{3 - 4z^{-1} + z^{-2}} = \frac{0 + z^{-1}}{3 - 4z^{-1} + z^{-2}}$$

a. Sử dụng lệnh **residuez** của MATLAB, tính các điểm cực, thặng dư tại các điểm cực theo chương trình mẫu bằng cách gõ các dòng lệnh theo bảng dưới đây vào cửa số soạn thảo (Editor) và ghi lại theo tên tệp **Solution_1_18**.

```
b = [0 1]; a = [3 -4 1];
[R,p,C] = residuez(b,a)
%
[b a] = residuez(R,p,C)
```

Gõ lệnh **Solution_1_18** tại cửa sổ lệnh của MATLAB để chạy kịch bản nói trên và xem kết quả tính toán. Từ đó hãy viết dạng tổng các hàm phân thức đơn giản của X(z).

b. Từ kết quả câu trên, viết công thức khai triển X(z) thành tổng các phân thức đơn giản, từ đó tìm biến đổi Z ngược của X(z) trên miền sao cho x(n) là một dãy nhân quả.

c. Kiểm chứng lại kết quả câu b bằng hàm `iztrans`

1.19. Cho hàm X(z) với công thức như sau:

$$X(z) = \frac{1}{\left(1-0{,}9z^{-1}\right)^2\left(1-0{,}9z^{-1}\right)}$$

a. Viết chương trình tính các điểm cực, thặng dư của các điểm cực của hàm X(z) trên

(gợi ý: có thể dùng hàm **poly** của MATLAB để khôi phục lại đa thức mẫu số từ một mảng các nghiệm của đa thức - mảng các điểm cực của X(z))

b. Từ kết quả câu trên, viết công thức khai triển X(z) thành tổng các phân thức đơn giản, từ đó tìm biến đổi Z ngược của X(z) trên miền $|z| > 0{,}9$.

1.20. Cho hệ thống nhân quả biểu diễn bởi phương trình sau:

$$y(n) - 0{,}9y(n-1) = x(n)$$

a. Tìm hàm truyền đạt của hệ thống

Sau đó thực hiện các công việc sau:

b. Dùng lệnh **zplane** của MATLAB biểu diễn trên đồ thị mặt phẳng Z sự phân bố các điểm cực và điểm không

```
b = [1 0]; a = [1 -0.9];
% Tim phan bo diem cuc va diem khong
subplot(1,2,1);
zplane(b,a);
title('Z plane');
% Tim dap ung tan so bang cach danh gia 200 diem roi rac
% cua H(z) tren duong tron don vi
[H, w] = freqz(b,a,200,'whole');
magH = abs(H(1:101)); phaH= angle(H(1:101));
% Ve dap ung tan so
subplot(2,2,2); plot(w(1:101)/pi,magH); grid;
title('Magnitude Response');
xlabel('frequency in pi units');
ylabel('Magnitude');
subplot(2,2,4); plot(w(1:101)/pi,phaH/pi); grid;
title('Phase Response');
xlabel('frequency in pi units');
ylabel('Phase in pi units');
```

c. Dùng lệnh **freqz** tính và biểu diễn trên đồ thị hàm đáp ứng tần số $H(e^{j\omega})$ của hệ thống (bao gồm đáp ứng biên độ - tần số và đáp ứng pha - tần số) tại 200 điểm rời

rạc trên đường tròn đơn vị theo chương trình mẫu bằng cách gõ các dòng lệnh theo bảng trên vào cửa sổ soạn thảo (Editor) và ghi lại theo tên tệp **Solution_1_20**.

Gõ lệnh **Solution_1_20** tại cửa sổ lệnh của MATLAB để chạy kịch bản nói trên và xem các đồ thị.

1.21. Cho hệ thống nhân quả biểu diễn bởi phương trình sau:

$$y(n) - 0{,}81y(n-2) = x(n) - x(n-2)$$

a. Viết công thức hàm truyền đạt H(z) của hệ thống

Viết các chương trình bằng MATLAB thực hiện các công việc sau:

b. Tính vị trí các điểm cực, các hệ số trong khai triển H(z) thành tổng các phân thức đơn giản.

c. Biểu diễn phân bố điểm cực và điểm không trên mặt phẳng Z

d. Tính và biểu diễn trên đồ thị hàm đáp ứng tần số $H(e^{j\omega})$ của hệ thống (bao gồm đáp ứng biên độ - tần số và đáp ứng pha - tần số) tại 200 điểm rời rạc trên đường tròn đơn vị.

Từ kết quả thu được ở câu b. tìm hàm đáp ứng xung h(n) của hệ thống.

1.22. Tạo các hàm thực hiện việc biến đổi Fourier rời rạc thuận và Fourier rời rạc ngược theo chương trình mẫu bằng cách gõ các dòng lệnh cho ở 2 bảng dưới đây vào cửa sổ soạn thảo (Editor) và ghi lại theo các tên tệp lần lượt là **dft.m**, và **idft.m**:

Tìm biến đổi Fourier rời rạc thuận:

```
function [Xk] = dft(xn,N)
% Tim bien doi Fourier roi rac thuan
% -----------------------------------------
% [Xk] = dft(xn,N)
% Xk = day cac he so DFT tren doan 0<=k<=N-1
% xn = day huu han N diem
%  N = chieu dai DFT
%
n = [0:1:N-1];
k = [0:1:N-1];
WN = exp(-j*2*pi/N);
nk = n' * k;
WNnk = WN .^ nk;         % ma tran DFT
Xk = xn * WNnk;
```

Tìm biến đổi Fourier rời rạc ngược:

```
function [xn] = idft(Xk,N)
% Tim bien doi Fourier roi rac nguoc
% ------------------------------------------
% [xn] = idft(Xk,N)
% xn = day co chieu dai huu han tren doan 0<=n<=N-1
% Xk = day cac he so DFT tren doan 0<=k<=N-1
%  N = chieu dai DFT
%
n = [0:1:N-1];
k = [0:1:N-1];
WN = exp(-j*2*pi/N);
nk = n' * k;
WNnk = WN .^ (-nk);          % ma tran IDFT
xn = (Xk * WNnk)/N;
```

1.23. Dựa trên các hàm **dft** được xây dựng ở trên, tìm biến đổi Fourier rời rạc của dãy có chiều dài N=20:

$$x(n) = \begin{cases} 1 & 0 \le n \le 4 \\ 0 & n \text{ còn lại} \end{cases}$$

theo chương trình mẫu bằng cách gõ các dòng lệnh theo bảng dưới đây vào cửa số soạn thảo (Editor) và ghi lại theo tên tệp **Solution_1_23**.

```
L = 5; N = 20;
n = [0:N-1];
xn = [ones(1,L), zeros(1,N-L)];
k = n;
Xk = dft(xn,N);
magXk = abs(Xk);
%
subplot(4,2,1); stem(n,xn);
axis([min(n),max(n)+1,-0.5,1.5]);
title('Sequence x(n)');
xlabel('n'); ylabel('x(n)');
subplot(4,2,3); stem(k,magXk);
axis([min(k),max(k)+1,-0.5,5.5]);
title('DFT of SQ. wave: L=5, N=20');
xlabel('k'); ylabel('X(k)');
```

Gõ lệnh **Solution_1_23** tại cửa sổ lệnh của MATLAB để chạy kịch bản nói trên và xem các đồ thị.

1.24. Viết chương trình tính và thể hiện trên đồ thị biến đổi Fourier rời rạc của các dãy sau:

a. Dãy có chiều dài N = 40 và $x(n)=\begin{cases}1 & 0\le n\le 4\\ 0 & n\ còn\ lại\end{cases}$

b. Dãy có chiều dài N = 60 và $x(n)=\begin{cases}1 & 0\le n\le 4\\ 0 & n\ còn\ lại\end{cases}$

c. Dãy có chiều dài N = 60 và $x(n)=\begin{cases}1 & 0\le n\le 6\\ 0 & n\ còn\ lại\end{cases}$

1.25. Biểu diễn trên đồ thị biểu đồ thể hiện mối quan hệ giữa chiều dài dãy N, N biến thiên từ 1 đến 2048, với thời gian thực hiện biến đổi Fourier của hàm MATLAB theo chương trình mẫu bằng cách gõ các dòng lệnh theo bảng dưới đây vào cửa sổ soạn thảo (Editor) và ghi lại theo tên tệp **Solution_1_25**.

```
Nmin = 1;
Nmax = 2048;
fft_time = zeros(1,Nmax-Nmin+1);
for n = Nmin:1:Nmax
    x = rand(1,n);
    t = clock;
    fft(x);
    fft_time(n-Nmin+1) = etime(clock,t);
end %for
n = [Nmin:1:Nmax];
plot(n,fft_time,'.')
xlabel('N');ylabel('Time in Secs');
title('FFT execution times');
```

Gõ lệnh **Solution_1_25** tại cửa sổ lệnh của MATLAB để chạy kịch bản nói trên và xem kết quả trên đồ thị. Nhận xét về kết quả thể hiện trên đồ thị.

6. Mở rộng

1. Viết các chương trình mô phỏng để kiểm chứng lại các tính chất của: Biến đổi Z, biến đổi Fourier, và biến đổi Fourier rời rạc.
2. Viết chương trình mô phỏng giải thuật tính tích chập nhanh (hay tích chập phân đoạn), có sử dụng ứng dụng của **fft** và **ifft** trong tính tích chập từng dãy con.

BÀI 2. THIẾT KẾ BỘ LỌC SỐ BẰNG MATLAB

A. THIẾT KẾ BỘ LỌC SỐ CÓ ĐÁP ỨNG XUNG CHIỀU DÀI HỮU HẠN (BỘ LỌC SỐ FIR)

1. Yêu cầu trước khi làm thí nghiệm

Sinh viên cần nắm vững các kiến thức về các đặc trưng của bộ lọc số có đáp ứng xung chiều dài hữu hạn, cụ thể:

- Tính chất tuần hoàn của hàm đáp ứng tần số, tính chất hàm chẵn của hàm đáp ứng biên độ - tần số, tính chất hàm lẻ của hàm đáp ứng pha - tần số
- Với bộ lọc FIR pha tuyến tính, tính chất đối xứng hoặc phản đối xứng của dãy đáp ứng xung
- Quan hệ giữa đáp ứng tần số của từng loại bộ lọc theo đáp ứng xung
- Phân bố điểm không của hàm truyền đạt trên mặt phẳng Z.

Đồng thời, sinh viên cần nắm vững các kỹ thuật tổng hợp bộ lọc có đáp ứng xung chiều dài hữn hạn bao gồm các phương pháp sau:

- Phương pháp cửa sổ
- Phương pháp lấy mẫu tần số
- Phương pháp lặp

2. Mục đích của phần thí nghiệm

Sinh viên dùng MATLAB mô phỏng các nội dung sau:

- Thiết kế bộ lọc bằng phương pháp cửa sổ
- Thiết kế bộ lọc bằng phương pháp lấy mẫu tần số
- Thiết kế bộ lọc bằng phương pháp lặp.

3. Tóm tắt lý thuyết

Quá trình lọc tín hiệu (filtering) nhằm tiến hành việc phân bố lại các thành phần tần số của tín hiệu. Quá trình lọc tín hiệu được thực hiện thông qua các **bộ lọc (filters)**. Dựa trên dãy đáp ứng xung của bộ lọc, có hai kiểu bộ lọc được quan tâm trong quá trình thiết kế đó là: Bộ lọc số có đáp ứng xung chiều dài hữu hạn, còn gọi là bộ lọc FIR và Bộ lọc số có đáp ứng xung chiều dài vô hạn, còn gọi là bộ lọc IIR. Phần này quan tâm đến các kỹ thuật để tổng hợp bộ lọc số FIR. Các kỹ thuật để tổng hợp bộ lọc IRR được xem xét ở phần sau.

Về mặt lý thuyết, dựa trên đặc điểm của đáp ứng tần số, Xử lý số tín hiệu quan tâm đến 4 loại bộ lọc lý tưởng sau đây:

a. Bộ lọc thông thấp lý tưởng

Đáp ứng biên độ - tần số: $\left|H_d\left(e^{j\omega}\right)\right| = \begin{cases} 1 & ,\left|\omega\right| \le \omega_c \\ 0 & ,n\ còn\ lại \end{cases}$

Khi đó đáp ứng xung của bộ lọc thông thấp lý tưởng với trễ nhóm α là:

$$h_d(n) = \frac{\omega_c}{\pi}\operatorname{sinc}\omega_c(n-\alpha)$$

b. Bộ lọc thông cao lý tưởng

Đáp ứng biên độ - tần số: $\left|H_d\left(e^{j\omega}\right)\right| = \begin{cases} 1 & ,\left|\omega\right| \ge \omega_c \\ 0 & ,n\ còn\ lại \end{cases}$

Khi đó đáp ứng xung của bộ lọc thông cao lý tưởng với trễ nhóm α là:

$$h_d(n) = \delta(n-\alpha) - \frac{\omega_c}{\pi}\operatorname{sinc}\omega c(n-\alpha)$$

c. Bộ lọc thông dải lý tưởng

Đáp ứng biên độ - tần số: $\left|H_d\left(e^{j\omega}\right)\right| = \begin{cases} 1 & ,\omega_{c1} \le \left|\omega\right| \le \omega_{c2} \\ 0 & ,n\ còn\ lại \end{cases}$

Khi đó đáp ứng xung của bộ lọc thông dải lý tưởng với trễ nhóm α là:

$$h_d(n) = \frac{\omega_{c2}}{\pi}\operatorname{sinc}\omega_{c2}(n-\alpha) - \frac{\omega_{c1}}{\pi}\operatorname{sinc}\omega_{c1}(n-\alpha)$$

d. Bộ lọc chắn dải lý tưởng

Đáp ứng biên độ - tần số: $\left|H_d\left(e^{j\omega}\right)\right| = \begin{cases} 0 & ,\omega_{c1} < \left|\omega\right| < \omega_{c2} \\ 1 & ,n\ còn\ lại \end{cases}$

Khi đó đáp ứng xung của bộ lọc chắn dải lý tưởng với trễ nhóm α là:

$$h_d(n) = \delta(n-\alpha) - \frac{\omega_{c2}}{\pi}\operatorname{sinc}\omega_{c2}(n-\alpha) + \frac{\omega_{c1}}{\pi}\operatorname{sinc}\omega_{c1}(n-\alpha)$$

Ngoài ra, bộ vi phân và bộ biến đổi Hilbert cũng được xem xét đến bởi chúng được ứng dụng rất nhiều trong truyền thông.

e. Bộ vi phân lý tưởng

Đáp ứng tần số: $\left|H_d\left(e^{j\omega}\right)\right| = \begin{cases} j\omega & ,0 < \omega \le \pi \\ -j\omega & ,-\pi < \omega \le 0 \end{cases}$

Khi đó đáp ứng xung của bộ vi phân lý tưởng với trễ nhóm α là:

$$h_d(n) = \begin{cases} \dfrac{\cos\pi(n-\alpha)}{(n-\alpha)} & ,n \ne \alpha \\ 0 & ,n = \alpha \end{cases}$$

f. Bộ biến đổi Hilbert

Đáp ứng tần số: $\left|H_d\left(e^{j\omega}\right)\right| = \begin{cases} -j & ,0 < \omega \le \pi \\ j & ,-\pi < \omega \le 0 \end{cases}$

Khi đó đáp ứng xung của bộ biến đổi Hilbert lý tưởng với trễ nhóm α là:

$$h_d(n) = \begin{cases} \dfrac{2}{\pi} \dfrac{\sin^2\left[\dfrac{\pi(n-\alpha)}{2}\right]}{(n-\alpha)} & ,n \ne \alpha \\ 0 & ,n = \alpha \end{cases}$$

Chúng ta có nhận xét là đáp ứng xung của các bộ lọc lý tưởng nói trên có chiều dài vô hạn, xuất phát từ chỉ số -∞ đến +∞, và không nhân quả, dẫn đến không thể thực hiện được về mặt vật lý. Khi tổng hợp bộ lọc thực tế, ta phải chấp nhận đáp ứng xung phải xuất phát từ chỉ số 0 để đáp ứng điều kiện nhân quả. Khi đó, đáp ứng tần số của bộ lọc thực tế có phần quá độ từ dải thông đến dải chắn, hoặc ngược lại, và được gọi là dải chuyển tiếp (transition band). Đồng thời phải có sự gợn sóng (ripple) ở cả dải thông và dải chắn hoặc ít nhất tại một trong hai, dải thông hoặc dải chắn.

Việc thiết kế bộ lọc là quá trình tìm ra các tham số, hay dãy đáp ứng xung của bộ lọc, thoả mãn các yêu cầu chỉ tiêu kỹ thuật cho trước, cụ thể là một số hoặc tất cả các **tham số tuyệt đối (absolute specification)** sau:

- Tần số cắt dải thông ω_p
- Tần số cắt dải thông ω_s
- Bề rộng dải quá độ $\Delta\omega$
- Độ gợn sóng dải thông δ_1
- Độ gợn sóng dải chắn δ_2

Trên thực tế, các tham số thường được cho dưới dạng **tương đối (relative specification)** tính theo đơn vị decibels dưới dạng sau đây:

- Độ gợn sóng dải thông theo dB, được tính bằng công thức:

$$R_p = -20\log\frac{1-\delta_1}{1+\delta_1}[dB]$$

- Độ suy giảm dải chắn theo dB được tính bằng công thức:

$$A_s = -20\log\frac{\delta_2}{1+\delta_1}[dB]$$

Do đáp ứng tần số là biến đổi Fourier của dãy đáp ứng xung, mà dãy này có tính chất rời rạc theo thời gian, nên hàm đáp ứng tần số của một hệ thống là tuần hoàn với chu kỳ 2π. Bởi yêu cầu thiết kế là bộ lọc tổng hợp được về mặt thực tế, hay nói một cách khác xử lý các dãy tín hiệu với giá trị thực, nên dãy đáp ứng xung h(n) là dãy thực, do đó hàm đáp ứng tần số $H(e^{j\omega})$ có tính chất đối xứng Hermit, có nghĩa là $H^*(e^{j\omega}) = H(e^{-j\omega})$.

Điều này dẫn đến với mọi bộ lọc thực tế đáp ứng biên độ - tần số là một hàm chẵn, đáp ứng pha - tần số là một hàm lẻ. Vì vậy, khi xem xét hàm đáp ứng tần số của bộ lọc, chỉ cần xem xét ω trong khoảng [0,π] là đủ.

Bộ lọc FIR có một số ưu điểm về mặt thực hiện như sau:

- Đáp ứng pha là tuyến tính
- Tương đối dễ thiết kế và luôn luôn là hệ thống ổn định
- Thực hiện được với hiệu quả cao
- Có thể thực hiện được trên cơ sở áp dụng biến đổi Fourier rời rạc.

Với bộ lọc FIR ta luôn đặt được điều kiện pha tuyến tính, điều này có nghĩa đáp ứng pha - tần số là một hàm số bậc nhất theo tần số ω, tương đương với thực hiện việc trễ hàm đáp ứng xung ở miền thời gian. Khi một hệ thống có pha tuyến tính, trễ nhóm (group delay) là một hằng số, thì có ưu điểm là các thành phần tần số khác nhau của tín hiệu tại đầu vào có cùng thời gian trễ như nhau sau khi cho qua hệ thống tại đầu ra. Hàm đáp ứng pha - tần số của bộ lọc FIR có dạng như sau:

$$\theta(\omega) = \beta - \alpha\omega$$, với α, β là các hằng số

và hàm đáp ứng tần số của bộ lọc FIR được cho dưới dạng độ lớn và pha như sau:

$$H(e^{j\omega}) = A(e^{j\omega})e^{j\theta(\omega)}$$, với $A(e^{j\omega})$ là hàm thực

Khi áp đặt thêm điều kiện pha tuyến tính vào bộ lọc FIR, dãy đáp ứng xung của bộ lọc chỉ có thể đối xứng hoặc phản đối xứng. Dựa trên tính chất đối xứng hay phản đối xứng của dãy đáp ứng xung và chiều dài N của dãy đáp ứng xung, người ta phân loại bộ lọc FIR làm 4 loại sau đây:

- Bộ lọc FIR loại 1: h(n) đối xứng, N lẻ, $\beta = 0$, $\alpha = \frac{N-1}{2}$
- Bộ lọc FIR loại 2: h(n) đối xứng, N chẵn, $\beta = 0$, $\alpha = \frac{N-1}{2}$
- Bộ lọc FIR loại 3: h(n) phản đối xứng, N lẻ, $\beta = \frac{\pi}{2}$, $\alpha = \frac{N-1}{2}$
- Bộ lọc FIR loại 4: h(n) phản đối xứng, N lẻ, $\beta = \frac{\pi}{2}$, $\alpha = \frac{N-1}{2}$

Đáp ứng tần số của bộ lọc FIR cho từng loại là như sau:

a. FIR loại 1:

$$H(e^{j\omega}) = \left[\sum_{n=0}^{\frac{N-1}{2}} a(n)\cos n\omega\right] e^{-j\frac{N-1}{2}\omega}$$

với $a(0) = h\left(\frac{N-1}{2}\right)$, $h\left(\frac{N-1}{2}\right)$ là mẫu giữa của dãy đáp ứng xung.

$a(n) = 2h\left(\frac{N-1}{2} - n\right)$, với $1 \le n \le \frac{N-1}{2}$

dẫn đến $A\left(e^{j\omega}\right) = \sum_{n=0}^{\frac{N-1}{2}} a(n)\cos n\omega$, và $\theta(\omega) = -\frac{N-1}{2}\omega$

b. *FIR loại 2:*

$$H\left(e^{j\omega}\right) = \left[\sum_{n=1}^{\frac{N}{2}} b(n)\cos\left(n - \frac{1}{2}\right)\omega\right] e^{-j\frac{N-1}{2}\omega}$$

với $b(n) = 2h\left(\frac{N}{2} - n\right)$, với $1 \le n \le \frac{N}{2}$

dẫn đến $A\left(e^{j\omega}\right) = \sum_{n=1}^{\frac{N}{2}} b(n)\cos\left(n - \frac{1}{2}\right)\omega$, và $\theta(\omega) = -\frac{N-1}{2}\omega$

c. *FIR loại 3:*

$$H\left(e^{j\omega}\right) = \left[\sum_{n=1}^{\frac{N-1}{2}} c(n)\sin n\omega\right] e^{j\left(\frac{\pi}{2} - \frac{N-1}{2}\omega\right)}$$

với $c(n) = 2h\left(\frac{N-1}{2} - n\right)$, với $1 \le n \le \frac{N-1}{2}$

dẫn đến $A\left(e^{j\omega}\right) = \sum_{n=1}^{\frac{N-1}{2}} c(n)\sin n\omega$, và $\theta(\omega) = \frac{\pi}{2} - \frac{N-1}{2}\omega$

d. *FIR loại 4:*

$$H\left(e^{j\omega}\right) = \left[\sum_{n=1}^{\frac{N}{2}} d(n)\sin\left(n - \frac{1}{2}\right)\omega\right] e^{j\left(\frac{\pi}{2} - \frac{N-1}{2}\omega\right)}$$

với $d(n) = 2h\left(\frac{N}{2} - n\right)$, với $1 \le n \le \frac{N}{2}$

dẫn đến $A\left(e^{j\omega}\right) = \sum_{n=1}^{\frac{N}{2}} d(n)\sin\left(n - \frac{1}{2}\right)\omega$, và $\theta(\omega) = \frac{\pi}{2} - \frac{N-1}{2}\omega$

Đối với các hệ thống FIR nói chung, hàm truyền đạt H(z) có duy nhất một điểm cực tại 0, bậc N-1, và miền hội tụ RC: $|z| > 0$.Bởi dãy đáp ứng xung là dãy thực nên hàm truyền đạt có tính chất đối xứng Hermit, cho nên nếu z_0 là một điểm không của H(z) thì z_0^* cũng là một điểm không của H(z). Do dãy đáp ứng xung của bộ lọc FIR pha tuyến tính là đối xứng hoặc phản đối xứng, dẫn đến nếu z_0 là một điểm không của H(z) thì $1/z_0$ cũng là một điểm không của H(z). Trường hợp tổng quát, nếu như biết một điểm không của đáp ứng tần số bộ lọc FIR pha tuyến tính H(z) tại $z_0 = re^{\theta}$, chúng ta suy ra rằng H(z) có 4 điểm không là: $z_{01} = z_0 = re^{\theta}$, $z_{02} = z_0^{-1} = \frac{1}{r}e^{-\theta}$, $z_{03} = z_0^{*} = re^{-\theta}$, và $z_{04} = \left(z_0^{-1}\right)^{*} = \frac{1}{r}e^{\theta}$. Trường hợp đặc biệt, điểm không nằm trên trục thực hoặc đường tròn đơn vị, số điểm không được suy ra suy giảm còn 2, và khi điểm không là 1 hoặc -1, không suy ra được thêm điểm không nào. Tính chất này có thể áp dụng trong thiết kế bộ lọc bằng cách mắc nối tiếp nhiều khâu, mỗi khâu có đáp ứng pha tuyến tính.

Theo sách giáo trình **"Xử lý tín hiệu và lọc số"**, có 3 phương pháp để tổng hợp bộ lọc FIR pha tuyến tính, đó là:

- Phương pháp cửa sổ
- Phương pháp lấy mẫu tần số
- Phương pháp lặp

a. Phương pháp cửa sổ

Tư tưởng cơ bản của phương pháp cửa sổ là tìm ra đáp ứng xung của bộ lọc lý tưởng rồi sau đó cắt xén ở hai đầu (hay nhân với một hàm cửa số) dãy đáp ứng xung đó sao cho ta thu được một bộ lọc FIR pha tuyến tính, đồng thời là nhân quả. Điểm nhấn mạnh ở phương pháp này là tìm ra đáp ứng xung thích hợp của bộ lọc lý tưởng và lựa chọn hàm cửa số thích hợp. Về mặt lý tưởng, bộ lọc thông thấp lý tưởng pha tuyến tính có độ lợi dải thông bằng 1 và đáp ứng tần số bằng 0 trên toàn dải chắn, tức là:

$$H_d\left(e^{j\omega}\right) = \begin{cases} e^{-j\alpha\omega} & ,|\omega| \le \omega_c \\ 0 & ,n\,còn\,lại \end{cases}$$, với ω_c là tần số cắt và α là trễ nhóm

thì ta sẽ thu được dãy đáp ứng xung

$$h_d(n) = \frac{\omega_c}{\pi}\operatorname{sinc}\omega_c(n-\alpha)$$

có tính chất đối xứng tại α.

Với các bộ lọc số lý tưởng khác, bao gồm thông cao, thông dải, và chắn dải, dãy đáp ứng xung cũng có dạng tương tự như vậy và có thể suy ra từ dạng đáp ứng xung của bộ lọc thông thấp lý tưởng nói trên. Với bộ vi phân và bộ biến đổi Hilbert, bằng biến đổi toán học, ta cũng có được đáp ứng xung có tính chất phản đối xứng tại α.

Để thu được đáp ứng xung của bộ lọc FIR về mặt thực tế, phương pháp của sổ dùng kỹ thuật nhân hàm đáp ứng xung của bộ lọc lý tưởng $h_d(n)$ với một hàm cửa sổ w(n), với w(n) là một hàm đối xứng đối với α trong khoảng từ 0 đến N-1 và bằng 0 trong

khoảng còn lại. Kết quả là hàm đáp ứng xung của bộ lọc thực tế h(n) là đối xứng hoặc phản đối xứng đối với $\alpha = \frac{N-1}{2}$ trong khoảng [0,N-1].

Một số cửa số thông dụng được lựa chọn là:

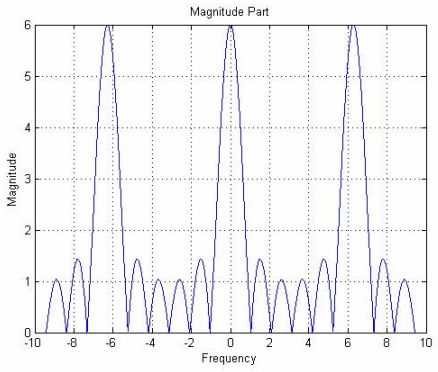
- **Cửa số chữ nhật**

$$w_r(n) = rect_N(n) = \begin{cases} 1 & ,0 \le n \le N-1 \\ 0 & ,n\ còn\ lại \end{cases}$$

Việc nhân cửa sổ chữ nhật với đáp ứng xung của bộ lọc lý tưởng ở miền thời gian tương đương với lấy tích chập liên tục tuần hoàn ở miền tần số giữa đáp ứng tần số của bộ lọc lý tưởng với ảnh qua phép biến đổi Fourier của hàm cửa sổ.

$$H(e^{j\omega}) = H_d(e^{j\omega}) \tilde{(*)} W_R(e^{j\omega}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\lambda}) W_R(e^{j(\omega-\lambda)}) d\lambda$$

Biến đổi Fourier của dãy chữ nhật được cho như hình vẽ dưới (đã được thực hành ở Bài 1). Ở đây ta có một số nhận xét về hàm biến đổi Fourier của dãy chữ nhật và đáp ứng tần số của bộ lọc thực tế khi dùng cửa số chữ nhật:



- Bởi cửa sổ w(n) có độ rộng bằng N nên trong khoảng [-π,π] ở miền tần số, $W_R(e^{j\omega})$ có một bướu chính biên độ rất cao, độ rộng 4π/N và các bướu bên có biên độ nhỏ hơn.
- Tích chập tuần hoàn giữa đáp ứng tần số của bộ lọc số lý tưởng $H_d(e^{j\omega})$ với $W_R(e^{j\omega})$ sẽ tạo ra đáp ứng tần số của bộ lọc $H(e^{j\omega})$ giống với dạng của $H_d(e^{j\omega})$ nhưng bị xô đi.
- Bướu chính của $W_R(e^{j\omega})$ sẽ tạo ra dải chuyển tiếp của $H(e^{j\omega})$, bướu chính của $W_R(e^{j\omega})$ càng hẹp, tương đương với N lớn, dải chuyển tiếp của $H(e^{j\omega})$ sẽ càng nhỏ.
- Các bướu bên của $W_R(e^{j\omega})$ sẽ tạo ra sự gợn sóng như nhau ở cả dải thông và dải chắn của $H(e^{j\omega})$.

Lý thuyết và thực tế chứng tỏ một số đặc điểm chính của bộ lọc thực tế được tổng hợp theo phương pháp cửa sổ chữ nhật như sau:

- Giá trị xấp xỉ của độ rộng dải chuyển tiếp (tính từ đỉnh gợn sóng cuối cùng của dải thông đến khi đáp ứng tần số giảm đến không) bằng độ rộng của bướu chính và bằng $\frac{4\pi}{N}$.
- Tỷ số giữa đỉnh bướu bên đầu tiên và đỉnh bướu chính là 13dB.
- Sau phép tính tích chập liên tục tuần hoàn, đáp ứng biên độ được tích luỹ với nhiều bướu liên tiếp và bướu bên đầu tiên ở dải chắn sẽ rơi vào vị trí suy giảm 21dB so với

đỉnh ở dải thông, giá trị này không phụ thuộc M. Do đó độ suy giảm dải chắn tối thiểu là 21dB và cũng không phụ thuộc M.

Những đặc điểm trên nói lên rằng dùng cửa sổ chữ nhật có một số nhược điểm sau:

- Cho dù chiều rộng của cửa sổ N tăng, độ rộng của mỗi một bướu bên giảm đi nhưng diện tích tương đối của từng bướu đối với bướu chính không hề thay đổi nên độ suy giảm dải chắn tối thiểu vẫn giữ nguyên không thay đổi là 21dB. Độ suy giảm dải chắn tối thiểu là 21dB trong nhiều trường hợp là không đủ với yêu cầu của thiết kế.
- Cửa sổ chữ nhật có sự thay đổi đột ngột ở viền cửa sổ, tức là đơn giản ta chỉ cắt ở cả hai đầu của đáp ứng xung bộ lọc lý tưởng $h_d(n)$, dẫn đến hiện tượng Gibb. Nhìn trên đáp ứng tần số sẽ thấy các bó gợn dày lên khi tiến ra cạnh của dải thông và dải chắn.

Nhằm tăng độ suy giảm dải chắn và hạn chế hiện tượng Gibb, một số dạng cửa sổ sau đã được đưa ra và được áp dụng rất nhiều trong thiết kế các bộ lọc thực tế:

- **Cửa sổ Bartlet (hay cửa sổ tam giác)**

$$w(n) = \begin{cases} \dfrac{2n}{N-1} & ,0 \le n \le \dfrac{N-1}{2} \\ 2 - \dfrac{2n}{N-1} & \dfrac{N-1}{2} \le n \le N-1 \\ 0 & ,n\ còn\ lại \end{cases}$$

- **Cửa sổ Hanning (hay cửa số Hann)**

$$w(n) = \begin{cases} 0{,}5 - 0{,}5\cos\left(\dfrac{2\pi n}{N-1}\right) & ,0 \le n \le N-1 \\ 0 & ,n\ còn\ lại \end{cases}$$

- **Cửa số Hamming**

$$w(n) = \begin{cases} 0{,}54 - 0{,}46\cos\left(\dfrac{2\pi n}{N-1}\right) & ,0 \le n \le N-1 \\ 0 & ,n\ còn\ lại \end{cases}$$

- **Cửa sổ Blackman:** cho đến hài bậc hai

$$w(n) = \begin{cases} 0{,}42 - 0{,}5\cos\left(\dfrac{2\pi n}{N-1}\right) + 0{,}08\cos\left(\dfrac{4\pi n}{N-1}\right) & ,0 \le n \le N-1 \\ 0 & ,n\ còn\ lại \end{cases}$$

Rõ ràng luôn có sự đánh đổi giữa tính chất hẹp của dải chuyển tiếp và tính gợn sóng ở dải thông và dải chắn. Các loại cửa số làm giảm hiệu ứng gợn sóng ở dải thông và dải chắn luôn có xu hướng làm cho bề rộng của dải chuyển tiếp tăng lên.

Dưới đây là bảng tổng kết các thông số về độ rộng dải chuyển tiếp và độ suy giảm dải chắn tối thiểu đối với từng loại cửa sổ:

Tên cửa sổ	Độ rộng dải chuyển tiếp		Độ suy giảm dải chắn tối thiểu
	Xấp xỉ	Chính xác	
Chữ nhật	$\frac{4\pi}{N}$	$\frac{1{,}8\pi}{N}$	21dB
Bartlett	$\frac{8\pi}{N}$	$\frac{6{,}1\pi}{N}$	25dB
Hanning	$\frac{8\pi}{N}$	$\frac{6{,}2\pi}{N}$	44dB
Hamming	$\frac{8\pi}{N}$	$\frac{6{,}6\pi}{N}$	53dB
Blackman	$\frac{12\pi}{N}$	$\frac{11\pi}{N}$	74dB

Dạng cửa sổ càng phức tạp, để bù cho độ suy giảm dải chắn thấp và giảm hiện tượng Gibb thì phải đánh đổi lấy dải chuyển tiếp có độ rộng lớn hơn hay cần độ dài đáp ứng xung N lớn hơn nếu muốn duy trì dải chuyển tiếp có độ rộng không đổi và đương nhiên là bộ lọc sẽ có thiết kế phức tạp hơn.

- Cửa sổ Kaiser:

Là dạng cửa sổ hiệu quả nhất cho phép thiết kế với bộ lọc có độ suy giảm dải chắn đòi hỏi rất nhỏ. Phương trình cửa sổ được cho bởi:

$$w(n) = \frac{I_0\left[\beta\sqrt{1-\left(1-\frac{2n}{N-1}\right)^2}\right]}{I_0[\beta]}$$

với $I_0[]$ là hàm Bessel bậc không.

Trong sách giáo trình ***"Xử lý tín hiệu và lọc số"****, tham số* β *được thay bằng* $\beta\frac{N-1}{2}$*, tuy nhiên phần tóm tắt lý thuyết này ghi là* β *cho phù hợp với công thức tính* β *dưới đây và hàm tạo cửa số* **kaiser** *của MATLAB.*

Các công thức để tính độ rộng cửa sổ N và tham số β được tính từ độ rộng dải chuyển tiếp, độ gợn sóng dải thông, và độ suy giảm dải chắn như sau:

- Độ rộng dải chuyển tiếp đã được chuẩn hoá: $\Delta f = \frac{\omega_s - \omega_p}{2\pi}$
- Bậc của bộ lọc: $N \cong \frac{A_s - 7{,}95}{14{,}36\Delta f} + 1$

- $\beta = \begin{cases} 0{,}1102(A_s - 8{,}7) & , A_s \geq 50 \\ 0{,}5842(A_s - 21)^{0{,}4} - 0{,}07886(A_s - 21) & , 21 < A_s < 50 \end{cases}$

Các hàm cửa sổ nói trên đều đã được MATLAB cung cấp sẵn.

b. Phương pháp lấy mẫu tần số

Tư tưởng của phương pháp này là xây dựng một bộ lọc có đáp ứng xung chiều dài N và có đáp ứng tần số xấp xỉ với đáp ứng tần số của bộ lọc lý tưởng. Cụ thể, ta có thể xét tại N mẫu rời rạc cách đều nhau trong khoảng từ 0 đến 2π, hàm đáp ứng tần số của bộ lọc thực tế bằng đúng với hàm đáp ứng xung của bộ lọc lý tưởng.

Nếu như ta đã biết N mẫu rời rạc H(k) trên hàm đáp ứng tần số, tương đương với N mẫu ảnh qua phép biến đổi Fourier rời rạc của dãy đáp ứng xung h(n), ta hoàn toàn có thể xây dựng hàm đáp ứng tần số $H(e^{j\omega})$ bằng phép nội suy theo công thức:

$$H(e^{j\omega}) = \frac{1 - e^{-j\omega N}}{N} \sum_{k=1}^{N-1} \frac{X(k)}{1 - e^{-j\omega} e^{j\frac{2\pi k}{N}}}$$

Đương nhiên là các giá trị X(k) chính là các giá trị của $H(e^{j\omega})$ tại các mẫu rời rạc:

$$H(k) = H\left(e^{j\frac{2\pi}{N}k} \right), \text{ với } k = 0,1,..., N-1$$

Do $H(e^{j\omega})$ là đáp ứng tần số của một hệ thống đặc trưng bởi dãy đáp ứng xung đơn vị thực nên $H(e^{j\omega})$ có tính chất đối xứng Hermit, tâm đối xứng tại 0, đồng thời $H(e^{j\omega})$ tuần hoàn chu kỳ 2π hay H(k) tuần hoàn chu kỳ N. Do đó các mẫu rời rạc X(k) phải có tính chất:

$$H(k) = H^*(N-k) \text{ với } k = 1,..., N-1$$

Riêng mẫu ứng với k=0 là một ngoại lệ bởi nó là tâm đối xứng nếu xét trong chu kỳ tuần hoàn của $H(e^{j\omega})$ là [-π,π].

Nếu đáp ứng tần số được viết dưới dạng độ lớn và pha:

$$H(e^{j\omega}) = A(e^{j\omega}) e^{j\theta(\omega)}, \text{ với } A(e^{j\omega}),\ \theta(\omega) \text{ là các hàm thực}$$

Ảnh của h(n) qua phép biến đổi Fourier rời rạc cũng được viết dưới dạng độ lớn và pha:

$$H(k) = A(k) e^{j\theta(k)}, \text{ với } A(k),\ \theta(k) \text{ là các dãy thực}$$

thì độ lớn và pha của dãy H(k) sẽ được tính theo công thức:

$$A(k) = A\left(e^{j\frac{2\pi}{N}k} \right) \text{ và } \theta(k) = \theta\left(\frac{2\pi}{N} k \right)$$

Do $H(e^{j\omega})$ và H(k) đều có tính chất đối xứng Hermit nên:

$$A(k) = A(N-k) \text{ với } k = 1,..., N-1$$

và:

- đối với bộ lọc FIR loại 1: $\theta(k)=\begin{cases} -\frac{N-1}{2}\left[\frac{2\pi}{N}k\right] & ,k=1,...,\frac{N-1}{2} \\ +\frac{N-1}{2}\left[\frac{2\pi}{N}(N-k)\right] & ,k=\frac{N+1}{2},...,N-1 \end{cases}$

- đối với bộ lọc FIR loại 2: $\theta(k)=\begin{cases} -\frac{N-1}{2}\left[\frac{2\pi}{N}k\right] & ,k=1,...,\frac{N-2}{2} \\ +\frac{N-1}{2}\left[\frac{2\pi}{N}(N-k)\right] & ,k=\frac{N}{2},...,N-1 \end{cases}$

- đối với bộ lọc FIR loại 3:

$$\theta(k)=\begin{cases} \frac{\pi}{2}-\frac{N-1}{2}\left[\frac{2\pi}{N}k\right] & ,k=1,...,\frac{N-1}{2} \\ -\frac{\pi}{2}+\frac{N-1}{2}\left[\frac{2\pi}{N}(N-k)\right] & ,k=\frac{N+1}{2},...,N-1 \end{cases}$$

- đối với bộ lọc FIR loại 4: $\theta(k)=\begin{cases} \frac{\pi}{2}-\frac{N-1}{2}\left[\frac{2\pi}{N}k\right] & ,k=1,...,\frac{N-2}{2} \\ -\frac{\pi}{2}+\frac{N-1}{2}\left[\frac{2\pi}{N}(N-k)\right] & ,k=\frac{N}{2},...,N-1 \end{cases}$

Nếu coi hàm *sai số xấp xỉ* được tính bằng độ sai lệch giữa đáp ứng tần số của bộ lọc lý tưởng với đáp ứng tần số của bộ lọc thực tế , ta có các nhận xét sau:

- Hàm sai số xấp xỉ bằng không tại các tần số được lấy mẫu
- Hàm sai số xấp xỉ tại các tần số khác phụ thuộc vào mức độ dốc hay độ biến đổi đột ngột tại tần số đó. Tại tần số có đáp ứng càng dốc, ví dụ gần biên của dải thông và dải chắn, thì có hàm sai số xấp xỉ càng lớn.

Dãy đáp ứng xung của bộ lọc được suy ra từ biến đổi Fourier rời rạc ngược của dãy các mẫu X(k):

$$h(n)=IDFT[X(k)]$$

và hàm tìm biến đổi Fourier rời rạc ngược bằng thuật toán biến đổi Fourier nhanh có thể được áp dụng trong trường hợp này.

Nếu như chúng ta áp dụng kỹ thuật thô, tức là các mẫu được nhận giá trị bằng 1 tại dải thông và nhận giá trị bằng 0 tại dải chắn, hiện tượng gợn sóng, hay hiện tượng Gibb, ở gần rìa các dải là tương đối lớn, độ suy giảm dải chắn rất nhiều trường hợp là không đạt yêu cầu. Chúng ta lại có nhận xét là để đạt độ suy giảm dải chắn tối thiểu càng nhiều ta luôn phải đánh đổi lấy độ rộng dải chuyển tiếp lớn (giống như đã được phân tích ở phương pháp cửa sổ). Kỹ thuật của chúng ta có thể áp dụng là tạo ra một số mẫu ở dải chuyển tiếp có thể nhận các giá trị trung gian nằm giữa 0 và 1. Số mẫu ở dải chuyển tiếp

là không lớn, chỉ cần từ 1 đến 2 mẫu là đủ do trên thực tế dải chuyển tiếp là rất nhỏ so với dải thông và dải chắn.

Thay đổi các giá trị chuyển tiếp quá độ có thể dẫn đến kết quả tốt hơn, nói một cách khác độ suy giảm dải chắn tối thiểu lớn hơn. Bài toán đặt ra ở đây là việc phải tìm cách tối ưu hoá giá trị tại 1 hay 2 mẫu đó để đạt được độ suy giảm dải chắn tối thiểu lớn nhất tương đương với việc tối thiểu hoá bướu bên lớn nhất. Trong *tối ưu hoá*, bài toán này gọi là bài toán **minimax**, và MATLAB cũng cung cấp hàm này trong bộ công cụ Optimization Toolbox. Tuy nhiên trong phần thực hành này, các giá trị ở dải chuyển tiếp là cho trước.

c. Phương pháp lặp

Hai phương pháp đã được trình bày ở trên, phương pháp cửa sổ và phương pháp lấy mẫu tần số, tồn tại một số bất cập. Thứ nhất ta không thể định được chính xác các tần số cắt ω_p và ω_s. Thứ hai ta không thể ràng buộc điều kiện đồng thời điều kiện về độ gợn sóng δ_1 và δ_2 ở cả dải thông và dải chắn. Thứ ba là hàm sai số xấp xỉ phân bố không đều trên các dải và có xu hướng tăng lên khi đến gần dải chuyển tiếp. Phương pháp lặp dựa trên thuật toán tối ưu có thể giải quyết được các vấn đề trên.

Phương pháp này được các tài liệu đề cập đến với một số tên gọi khác nhau: Optimal (Optimum) Equiriple, Remez Exchange. Bản chất của phương pháp này là xuất phát từ một chiều dài dãy N cho trước, bằng thuật toán trao đổi Remez để tìm ra dãy đáp ứng xung sao cho cực đại của hàm sai số giữa đáp ứng tần số của bộ lọc lý tưởng và đáp ứng tần số của bộ lọc thực tế là nhỏ nhất. Nếu như hàm đáp ứng tần số ứng với dãy đáp ứng xung tìm được nói trên vẫn chưa thoả mãn điều kiện yêu cầu của thiết kế, giá trị N cần phải tăng. Quá trình này được lặp đi lặp lại đến khi tìm ra được bộ lọc thoả mãn các yêu cầu đã được đặt ra.

Dưới đây sẽ trình bày tóm tắt về mặt lý thuyết quá trình tối thiểu hoá sai số cực đại giữa đáp ứng tần số của bộ lọc thực tế và đáp ứng tần số cuat bộ lọc lý tưởng. Trước tiên, ta đưa hàm độ lớn của đáp ứng tần số của 4 loại bộ lọc FIR về dạng sau:

$$A\left(e^{j\omega}\right) = P(\omega)Q(\omega)$$

với P(ω) là hàm có dạng: $P(\omega) = \sum_{n=0}^{R} \alpha(n)\cos n\omega$

Bảng sau đây đưa ra giá trị R, các hàm P(ω) và Q(ω) cho 4 loại bộ lọc:

Loại bộ lọc	**Q(ω)**	**R**	**P(ω)**
FIR loại 1	1	$\frac{N-1}{2}$	$\sum_{n=0}^{R} \bar{a}(n)\cos n\omega$
FIR loại 1	$\cos\frac{\omega}{2}$	$\frac{N}{2}-1$	$\sum_{n=0}^{R} \bar{b}(n)\cos n\omega$
FIR loại 1	$\sin\omega$	$\frac{N-1}{2}-1$	$\sum_{n=0}^{R} \bar{c}(n)\cos n\omega$

FIR loại 1 $\sin\frac{\omega}{2}$ $\frac{N}{2}-1$ $\sum_{n=0}^{R}\bar{d}(n)\cos n\omega$

Hàm *sai số* giữa bộ lọc lý tưởng và bộ lọc thực tế được xây dựng như sau:

$$E(\omega)=\mathrm{W}(\omega)[\mathrm{A_d}(\omega)-\mathrm{A}(\omega)] \text{ với } \omega\in S=[0,\omega_p]\cup[\omega_s,\pi]$$

- Hàm E(ω) có miền xác định chỉ là phần dải thông và dải chắn, mà không xác định tại dải chuyển tiếp.
- Hàm W(ω) được gọi là hàm trọng số có tác dụng trải đều sai số giữa bộ lọc thực tế và bộ lọc lý tưởng trên cả dải thông và dải chắn.

Nếu ta lựa chọn hàm trọng số trong trường hợp $\delta_2>\delta_1$, với δ_1 và δ_2 lần lượt là độ độ gợn sóng của dải thông và dải chắn, là:

$$\mathrm{W}(\omega)=\begin{cases}\dfrac{\delta_2}{\delta_1} & \textit{ở dải thông}\\ 1 & \textit{ở dải chắn}\end{cases}$$

thì hàm sai số ở cả dải thông và dải chắn đều không vượt quá δ_2 ở cả dải thông và dải chắn. Điều này có nghĩa nếu như ta tối thiểu hoá cực đại của hàm sai số E(ω) là δ_2 ta tự động có luôn cực đại của sai số giữa bộ lọc thực tể và bộ lọc lý tưởng ở dải chắn là δ_2 và ở dải thông là δ_1.

Chúng ta nhận thấy là hàm Q(ω) ở 4 loại bộ lọc là khác nhau, để triệt tiêu hàm này, hàm sai số được biến đổi như sau:

$$E(\omega)=\mathrm{W}(\omega)[\mathrm{A_d}(\omega)-\mathrm{A}(\omega)]=\mathrm{W}(\omega)[\mathrm{A_d}(\omega)-\mathrm{P}(\omega)\mathrm{Q}(\omega)]=\mathrm{W}(\omega)\mathrm{Q}(\omega)\left[\frac{\mathrm{A_d}(\omega)}{\mathrm{Q}(\omega)}-\mathrm{P}(\omega)\right]$$

Ở đây nếu như định nghĩa các hàm trọng số biến dạng và hàm độ lớn của đáp ứng tần số bộ lọc lý tưởng biến dạng là:

$$\hat{\mathrm{W}}(\omega)=\mathrm{W}(\omega)\mathrm{Q}(\omega) \text{ và } \hat{\mathrm{A}}_\mathrm{d}(\omega)=\frac{\mathrm{A_d}(\omega)}{\mathrm{Q}(\omega)}$$

thì hàm sai số của cả 4 loại bộ lọc có cùng dạng chung:

$$E(\omega)=\hat{\mathrm{W}}(\omega)[\hat{\mathrm{A}}_\mathrm{d}(\omega)-\mathrm{P}(\omega)]$$

Bài toán Chebyshev đặt ra là: *Tìm các hệ số $\bar{a}(n)$, hoặc $\bar{b}(n)$, hoặc $\bar{c}(n)$, hoặc $\bar{d}(n)$ nhằm tối thiểu hoá cực đại của trị tuyệt đối hàm sai số trên dải thông và dải chắn, tức là tìm ra:*

$$\min_{\textit{trong tập các hệ số}}\left[\max_{\omega\in S}|E(\omega)|\right]$$

Định lý xoay chiều: *S là một khoảng đóng bất kỳ (còn gọi là đoạn vì nó chứa cả biên) trên đoạn $[0,\pi]$, giả sử $P(\omega)$ có dạng:*

$$P(\omega) = \sum_{n=0}^{R} \alpha(n) \cos n\omega$$

Hàm sai số $E(\omega)$ *được tính theo công thức:*

$$E(\omega) = \hat{W}(\omega)\left[\hat{A}_{\mathrm{d}}(\omega) - P(\omega)\right]$$

Điều kiện cần và đủ để $P(\omega)$ *duy nhất và xấp xỉ theo kiểu cực đại là nhỏ nhất, theo nghĩa gần đúng Chebyshev, so với* $\hat{\mathrm{A}}_{\mathrm{d}}(\omega)$ *trên S là:* ***Hàm sai số*** $E(\omega)$***phải có tối thiểu R+2 giá trị cực trị đổi dấu xen kẽ nhau trên S mà:***

$$E(\omega_i) = -E(\omega_{i-1}) = \pm \max_{\omega \in S} |E(\omega)|$$

với $\omega_0 < \omega_1 < \omega_2 < \ldots < \omega_{R+1}$

Định lý nói trên không chỉ ra cách thức để thu được hàm $P(\omega)$. Tuy nhiên nó chỉ ra rằng nghiệm đó tồn tại, không những thế nghiệm là duy nhất và điều kiện để biết đó là nghiệm khi hàm sai số $E(\omega)$ có ít nhất R+2 cực trị, các cực trị này có giá trị tuyệt đối bằng nhau, hai cực trị liên tiếp có một là cực đại và một là cực tiểu. Để tìm ra hàm $P(\omega)$, thuật toán trao đổi Remez được tiến hành như sau:

1. Chọn lấy R+2 điểm rời rạc, giả sử đó là các cực trị của hàm sai số
2. Trên cơ sở tại R+2 điểm rời rạc nói trên, hàm $E(\omega)$ luân phiên đổi dấu và có trị tuyệt đối bằng một giá trị δ nào đó, tính nội suy lại giá trị δ và hàm $P(\omega)$, từ đó tính ra hàm sai số $E(\omega)$, tính được giá trị cực trị thực của hàm sai số đó
3. Xem xét xem các giá trị rời rạc được chọn ban đầu có thực sự là các điểm mà hàm sai số $E(\omega)$ đạt cực trị và có trị tuyệt đối bằng nhau hay không. Nếu không, tìm các điểm tại đó $E(\omega)$ đạt cực trị.
4. Trong các điểm cực trị đó của $E(\omega)$ lấy ra R+2 điểm và quay về lặp lại từ bước 2.
5. Lặp lại các bước 2, 3, và 4 cho đến khi tập hợp các điểm rời rạc hội tụ
6. Từ tập các điểm rời rạc cuối cùng, tính ra hàm $P(\omega)$, từ đó tính ra các hệ số $\alpha(n)$.

Vòng lặp tiếp theo bao giờ cũng thu được R+2 điểm rời rạc tiến gần tới những cực trị của hàm $P(\omega)$ mà chúng ta mong muốn gần đúng với $\hat{\mathrm{A}}_{\mathrm{d}}(\omega)$ theo nghĩa Chebyshev hơn, và cuối cùng nó sẽ hội tụ về các điểm cực trị thực. Một vấn đề trong việc sử dụng chương trình máy tính để tìm ra nghiệm là máy tính không thể làm việc với các giá trị liên tục trên miền S mà chỉ có khả năng tính toán trên tập các giá trị rời rạc. Thực tế ta có thể thiết kế một lưới các giá trị rời rạc trên miền S rồi tính gần đúng trên đó.

Parks và McClellan đã đưa ra giải pháp sử dụng thuật toán Remez để tìm ra đáp ứng xung của bộ lọc tối ưu nhất, tức là gần đúng theo nghĩa Chebyshev đối với một bộ

lọc lý tưởng, cho giá trị N là chiều dài của dãy đáp ứng xung nào đó với các điều kiện ràng buộc về độ gợn sóng ở dải thông và dải chắn như sau:

1. Xác định loại bộ lọc, tính giá trị R và xây dựng các hàm $W(\omega)$, $Q(\omega)$
2. Xây dựng bài toán gần đúng bằng cách xác định các hàm $\hat{W}(\omega)$, $\hat{A}_{\mathrm{d}}(\omega)$
3. Sử dụng thuật toán trao đổi Remez để tìm ra hàm tối ưu $P(\omega)$
4. Tính các giá trị của dãy đáp ứng xung h(n)

Tất nhiên khi chọn giá trị N càng chuẩn thì kết quả là thu được bộ lọc có hàm đáp ứng tần số càng gần với yêu cầu bài toán. Nếu như với giá trị N nào đó mà chưa thoả mãn được yêu cầu cầu của bài toán thì ta phải tăng giá trị N đến khi nào thoả mãn các điều kiện ràng buộc cho δ_1 và δ_2 (hay A_s và R_p) thì thôi. Về mặt kinh nghiệm, một số tài liệu đưa ra công thức lựa chọn ban đầu cho chiều dài N của dãy đáp ứng xung là:

$$\hat{N} = \frac{-20\log\sqrt{\delta_1\delta_2} - 13}{14{,}6\Delta f} \qquad \text{,với } \Delta f = \frac{\omega_s - \omega_p}{2\pi}$$

Trong MATLAB, việc tìm ra dãy đáp ứng xung của bộ lọc tối ưu với giá trị N và hàm đáp ứng tần số lý tưởng cho trước được thực hiện bởi hàm **firpm**.

4. Một số lệnh và hàm của MATLAB

Phần này đưa ra danh mục các lệnh các hàm của MATLAB có thể sử dụng trong phần thí nghiệm này. Để biết cụ thể hơn về chức năng của hàm và cú pháp của lệnh gọi hàm, gõ lệnh **help** kèm theo tên của hàm tại cửa số lệnh của MATLAB.

`freqz`: trả về đáp ứng tần số của một hệ thống tại một số hữu hạn các điểm rời rạc trên vòng tròn đơn vị khi biết hàm truyền đạt của nó

`sin, cos, sinc`: trả về các hàm toán học thể hiện các công thức lượng giác

`boxcar, bartlett, hanning, hamming, blackman, kaiser`: trả về các hàm cửa sổ với chiều dài cho trước

`firpm`: thực hiện công việc tìm ra dãy đáp ứng xung của bộ lọc tối ưu theo nghĩa Chebyshev bằng thuật toán của Parks và McClellan.

Ngoài ra, bộ công cụ ***Signal Processing Toolbox*** của MATLAB đã cung cấp rất nhiều hàm hữu ích để thiết kế bộ lọc như **fir1**, **fir2**, **firls**... Thêm vào nữa bộ công cụ này cũng cung cấp một gói phần mềm thiết kế sẵn với tên **FDATool** giúp các kỹ sư giảm thời gian rất nhiều trong thiết kế các hệ số của bộ lọc. Tuy nhiên bộ công cụ ***Signal Processing Toolbox*** có thể không được cung cấp kèm theo phiên bản phần mềm MATLAB dành cho sinh viên thực hành. Đồng thời, để giúp sinh viên hiểu rõ thêm về bản chất toán học trong quá trình thiết kế bộ lọc, phần thực hành này yêu cầu tiến hành từng bước thiết kế các bộ lọc rất cụ thể. Chỉ có một hàm thiết kế bộ lọc có sẵn trong bộ công cụ nên được dùng là **firpm** bởi xây dựng lại một chương trình phần mềm theo thuật toán của Parks và McClellan dùng thuật toán trao đổi Remez là khá phức tạp.

5. Các bước thực hành

2.1. Tạo các hàm thể hiện độ lớn của đáp ứng tần số các bộ lọc FIR loại 1 và FIR loại 2 từ dãy đáp ứng xung của chúng theo chương trình mẫu bằng cách gõ các dòng lệnh cho ở 2 bảng dưới đây vào cửa sổ soạn thảo (Editor) và ghi lại theo các tên tệp lần lượt là **Hr_Type1.m** và **Hr_Type2.m**:

```
function [Hr,w,a,L] = Hr_Type1(h)
% Tinh ham do lon cua dap ung tan so Hr(w)
% bo loc FIR loai 1
% ------------------------------------------------
% [Hr,w,a,L] = Hr_Type1(h)
% Hr = Do lon
%  w = Vector tan so trong khoang [0 pi]
%  a = Cac he so cua bo loc FIR loai 1
%  L = Bac cua bo loc
%  h = Dap ung xung cua bo loc FIR loai 1
%
M  = length(h);
L  = (M-1)/2;
a  = [h(L+1) 2*h(L:-1:1)];
n  = [0:1:L];
w  = [0:1:500]'*pi/500;
Hr = cos(w*n)*a';
```

Hàm độ lớn của đáp ứng tần số bộ lọc FIR loại 1:

Hàm độ lớn của đáp ứng tần số bộ lọc FIR loại 2:

2.2. Viết chương trình tính hàm độ lớn của đáp ứng tần số bộ lọc FIR loại 3 và bộ lọc FIR loại 4 với các tham số đầu vào và đầu ra được nhập theo các câu lệnh:

>> `[Hr,w,c,L] = Hr_Type3(h)` -> cho bộ lọc FIR loại 3

>> `[Hr,w,d,L] = Hr_Type4(h)` -> cho bộ lọc FIR loại 4

2.3. Cho bộ lọc FIR với đáp ứng xung như sau:

```
function [Hr,w,b,L] = Hr_Type2(h)
% Tinh ham do lon cua dap ung tan so Hr(w)
% bo loc FIR loai 2
% ------------------------------------------------
% [Hr,w,a,L] = Hr_Type2(h)
% Hr = Do lon
%  w = Vector tan so tron khoang [0 pi]
%  b = Cac he so cua bo loc FIR loai 2
%  L = Bac cua bo loc
%  h = Dap ung xung cua bo loc FIR loai 2
%
M  = length(h);
L  = M/2;
b  = 2*h(L:-1:1);
n  = [1:1:L]; n = n-0.5;
w  = [0:1:500]'*pi/500;
Hr = cos(w*n)*b';
```

$$h(n) = \left\{ \underset{\uparrow}{4},1,-1,-2,5,6,5,-1,-2,1,-4 \right\}$$

a. Xác định loại của bộ lọc.

Tính và biểu diễn trên đồ thị:

b. Dãy đáp ứng xung của bộ lọc

c. Các hệ số của bộ lọc

d. Hàm độ lớn của đáp ứng tần số

e. Phân bố điểm cực và điểm không

theo chương trình mẫu bằng cách gõ các dòng lệnh cho ở bảng dưới đây vào cửa số soạn thảo (Editor) và ghi lại theo tên tệp là **Solution_2_3.m**

```
h = [-4,1,-1,-2,5,6,5,-2,-1,1,-4];
M = length(h); n =0:M-1;
[Hr,w,a,L] = Hr_Type1(h);
a, L
amax = max(a)+1; amin = min(a)-1;
%
subplot(2,2,1); stem(n,h);
axis([-1,2*L+1,amin,amax]);
title('Impulse Response');
xlabel('n'); ylabel('h(n)');
%
subplot(2,2,3); stem(0:L,a);
axis([-1,2*L+1,amin,amax]);
title('a(n) coefficients');
xlabel('n'); ylabel('a(n)');
%
subplot(2,2,2); plot(w/pi,Hr); grid;
title('Type-1 Amplitude Response');
xlabel('frequency in pi units'); ylabel('Hr');
%
subplot(2,2,4); zplane(h,1);
```

Gõ lệnh **Solution_2_3** tại cửa sổ lệnh của MATLAB để chạy kịch bản nói trên và xem các đồ thị.

2.4. Cho bộ lọc FIR với đáp ứng xung như sau:

$$h(n) = \left\{ \underset{\uparrow}{4},1,-1,-2,5,6,-6,-5,1,2,-1,4 \right\}$$

a. Xác định loại của bộ lọc.

Viết chương trình tính và biểu diễn trên đồ thị:

b. Dãy đáp ứng xung của bộ lọc

c. Các hệ số của bộ lọc

d. Hàm độ lớn của đáp ứng tần số

Phân bố điểm cực và điểm không

2.5. Tạo hàm thể hiện độ dãy đáp ứng xung của bộ lọc thông thấp lý tưởng từ các tham số đầu vào là tần số cắt ω_c và chiều dài đáp ứng xung M theo chương trình mẫu bằng cách gõ các dòng lệnh cho ở bảng dưới đây vào cửa số soạn thảo (Editor) và ghi lại theo tên tệp là **ideal_lp.m**:

```
function hd = ideal_lp(wc,M)
% Ideal LowPass filter computation
% --------------------------------
% [hd] = ideal_lp(wc,M)
% hd = ideal impluse response between 0 to M-1
% wc = cutoff frequency in radians
%  M = length of the ideal filter
%
alpha = (M-1)/2;
n = [0:1:(M-1)];
m = n - alpha + eps;
hd = sin(wc*m) ./ (pi*m);
```

2.6. Hàm **freqz** của MATLAB trả về đáp ứng tần số của một hệ thống số khi biết trước hệ số của đa thức tử số và đa thức mẫu số theo z^{-1} của hàm truyền đạt H(z). Trong nhiều trường hợp, để thuận tiện ta cần tìm thêm các thông số: hàm độ lớn của đáp ứng tần số, hàm pha của đáp ứng tần số, hàm trễ nhóm, thể hiện độ lớn theo thang decibels. Tạo hàm tính đáp ứng tần số có tên sau **freqz_m** nhằm tính các thông số trên theo chương trình mẫu bằng cách gõ các dòng lệnh cho ở bảng dưới đây vào cửa số soạn thảo (Editor) và ghi lại theo tên tệp là **freqz_m.m**:

```
function [db,mag,pha,grd,w] = freqz_m(b,a)
% Phien ban sua doi cua ham freqz
% -------------------------------------
%  db = Do lon tuong doi theo dB tren doan tu 0 den pi
% mag = Do lon tuyet doi tren doan tu 0 den pi
% pha = Dap ung pha tren doan tu 0 den pi
% grd = Tre nhom tren doan tu 0 den pi
%   w = Cac mau tan so doan tu 0 den pi
%   b = Cac he so da thuc tu so cua H(z) (voi FIR: b=h)
%   a = Cac he so da thuc mau so cua H(z)(voi FIR: a=[1])
%
[H,w] = freqz(b,a,1000,'whole');
H = (H(1:1:501))';
w = (w(1:1:501))';
mag = abs(H);
db = 20*log10((mag+eps)/max(mag));
pha = angle(H);
grd = grpdelay(b,a,w);
```

2.7. Thiết kế bộ lọc thông thấp theo phương pháp cửa số với các tham số đầu vào như sau:

$$\omega_p = 0{,}2\pi\,, \qquad R_p = 0{,}25dB$$

$$\omega_s = 0{,}3\pi\,, \qquad A_s = 50dB$$

Với điều kiện đã cho của bài toán, dựa trên bảng tham số của các cửa số đã cho ở phần tóm tắt lý thuyết, cửa sổ Hamming và cửa sổ Blackman là có thể thoả mãn yêu cầu về độ suy giảm dải chắn hơn 50dB. Tuy nhiên ta thể tính trước các tham số theo độ gợn dải thông. Phần thiết kế ví dụ dưới đây lựa chọn cửa số Hamming. Tính và biểu diễn trên đồ thị:

a. Dãy đáp ứng xung của bộ lọc lý tưởng
b. Dãy hàm cửa sổ
c. Hàm độ lớn tuyệt đối của đáp ứng tần số
d. Hàm độ lớn tương đối tính theo dB của đáp ứng tần số

theo chương trình mẫu bằng cách gõ các dòng lệnh cho ở bảng dưới đây vào cửa số soạn thảo (Editor) và ghi lại theo tên tệp là **Solution_2_7.m**

```
wp = 0.2*pi; ws =0.3*pi;
tr_width = ws - wp;
M = ceil(6.6*pi/tr_width) + 1
n = [0:1:M-1];
wc = (ws+wp)/2;
hd = ideal_lp(wc,M);
w_ham = (hamming(M))';
h = hd .* w_ham;
[db,mag,pha,grd,w] = freqz_m(h,[1]);
delta_w = 2*pi/1000;
Rp = -(min(db(1:1:wp/delta_w+1)))
As = -round(max(db(ws/delta_w+1:1:501)))
%plot
subplot(2,2,1); stem(n,hd);
axis([0,M-1,-0.1,0.3]);
title('Ideal Impulse Response');
xlabel('n'); ylabel('hd(n)');
%
subplot(2,2,2); stem(n,w_ham);
axis([0,M-1,0,1.1]);
title('Hamming Window');
xlabel('n'); ylabel('w(n)');
%
subplot(2,2,3); stem(n,h);
axis([0,M-1,-0.1,0.3]);
title('Actual Impulse Response');
xlabel('n'); ylabel('h(n)');
```

```
%
subplot(2,2,4); plot(w/pi,db); grid;
axis([0,1,-100,10]);
title('Magnitude Response in dB');
xlabel('frequency in pi units'); ylabel('Decibels');
```

Gõ lệnh **Solution_2_7** tại cửa sổ lệnh của MATLAB để chạy kịch bản nói trên và xem các kết quả hiển thị ở cửa số lệnh (Command Window) và trên đồ thị.

2.8. Thiết kế bộ lọc thông dải theo phương pháp cửa số với các tham số đầu vào như sau:

$$\omega_{s1} = 0{,}2\pi\,, \quad A_s = 60dB$$

$$\omega_{p1} = 0{,}35\pi\,, \quad R_p = 1dB$$

$$\omega_{p2} = 0{,}65\pi\,, \quad R_p = 1dB$$

$$\omega_{s2} = 0{,}8\pi\,, \quad A_s = 60dB$$

Về mặt lý thuyết, chúng ta thấy rằng đáp ứng xung của bộ lọc thông dải lý tưởng là hiệu đáp ứng xung của hai bộ lọc thông thấp lý tưởng. Dùng hàm **ideal_lp** đã được viết ở trên là có thể tính được đáp ứng xung của bộ lọc thông dải lý tưởng. Tần số cắt có thể lấy là trung bình cộng của các tần số cắt dải thông và tần số cắt dải chắn.

Cửa số Blackman là có thể thoả mãn yêu cầu của bài toán đối với độ suy giảm dải chắn. Tham số độ rộng dải chuyển tiếp để tính chiều dài dãy đáp ứng xung (hay bậc của bộ lọc) có thể lựa chọn là giá trị nhỏ nhất của độ rộng hai dải chuyển tiếp, dải chuyển tiếp từ dải chắn lên dải thông $[\omega_{s1},\omega_{p1}]$ và dải chuyển tiếp từ dải thông xuống dải chắn $[\omega_{p2},\omega_{s2}]$.

Viết chương trình tính và biểu diễn trên đồ thị:

a. Dãy đáp ứng xung của bộ lọc lý tưởng
b. Dãy hàm cửa sổ
c. Hàm độ lớn tuyệt đối của đáp ứng tần số
d. Hàm độ lớn tương đối tính theo dB của đáp ứng tần số

2.9. Thiết kế bộ lọc thông thấp theo phương pháp lấy mẫu tần số với các tham số đầu vào như sau:

$$\omega_p = 0{,}2\pi\,, \quad R_p = 0{,}25dB$$

$$\omega_s = 0{,}3\pi\,, \quad A_s = 50dB$$

Giả sử rằng ta chọn đáp ứng xung có chiều dài 60 tương đương với lấy 60 mẫu tần số trong khoảng $[0,2\pi)$. Dải thông có độ rộng là $0,2\pi$ tương đương với 7 mẫu nhận giá trị 1. Giả sử tiếp rằng quá trình tối ưu hoá chỉ ra nên chọn dải chuyển tiếp 2 mẫu nhận các giá trị $T_1 = 0{,}5925$ và $T_2 = 0{,}1099$. Vậy dãy mẫu các tần số được cho như sau:

$$H(k)=\left\{1,1,1,1,1,1,1,1,T_1,T_2,\underbrace{0,...,0}_{43\ mẫu\ 0},T_2,T_1,1,1\,1,1,1,1\right\}$$

Tính và biểu diễn trên đồ thị:

a. Dãy các mẫu tần số

b. Dãy đáp ứng xung của bộ lọc thực tế

c. Hàm độ lớn tuyệt đối của đáp ứng tần số

d. Hàm độ lớn tương đối tính theo dB của đáp ứng tần số

theo chương trình mẫu bằng cách gõ các dòng lệnh cho ở bảng dưới đây vào cửa số soạn thảo (Editor) và ghi lại theo tên tệp là **Solution_2_9.m**

```
M = 60; alpha = (M-1)/2; l = 0:M-1; wl = (2*pi/M)*l;
Hrs = [ones(1,7),0.5925,0.1099,zeros(1,43),
0.1099,0.5925,ones(1,6)];
% Day dap ung tan so mau ly tuong
Hdr = [1,1,0,0]; wdl = [0,0.2,0.3,1];
% Dap ung tan so ly tuong de bieu dien do thi
k1 = 0:floor((M-1)/2); k2 = floor((M-1)/2)+1:M-1;
angH = [-alpha*(2*pi)/M*k1, alpha*(2*pi)/M*(M-k2)];
H = Hrs.*exp(j*angH);
h = real(ifft(H,M));
[db,mag,pha,grd,w] = freqz_m(h,1);
[Hr,ww,a,L] = Hr_Type2(h);
%plot

subplot(2,2,1); plot(wl(1:31)/pi,Hrs(1:31),'o',wdl,Hdr);
axis([0,1,-0.1,1.1]);
title('Frequency Samples: M=40, T2 = 0.5925, T1 =
0.1099');
xlabel('frequency in pi units'); ylabel('Hr(k)');
%
subplot(2,2,2); stem(l,h);
axis([-1,M,-0.1,0.3]);
title('Impulse Response');
xlabel('n'); ylabel('h(n)');
%
subplot(2,2,3); plot(ww/pi,Hr,wl(1:31)/pi,Hrs(1:31),'o');
axis([0,1,-0.2,1.2]);
title('Amplitude Response');
xlabel('frequency in pi units'); ylabel('Hr(w)');
%
subplot(2,2,4); plot(w/pi,db);
axis([0,1,-100,10]); grid
title('Magnitude Response');
xlabel('frequency in pi units'); ylabel('Decibels');
```

Gõ lệnh **Solution_2_9** tại cửa sổ lệnh của MATLAB để chạy kịch bản nói trên và xem các kết quả hiển thị ở cửa số lệnh (Command Window) và trên đồ thị.

2.10. Thiết kế bộ lọc thông dải theo phương pháp lấy mẫu tần số với các tham số đầu vào như sau:

$$\omega_{s1} = 0{,}2\pi\,, \quad A_s = 60dB$$

$$\omega_{p1} = 0{,}35\pi\,, \quad R_p = 1dB$$

$$\omega_{p2} = 0{,}65\pi\,, \quad R_p = 1dB$$

$$\omega_{s2} = 0{,}8\pi\,, \quad A_s = 60dB$$

Giả sử rằng ta chọn đáp ứng xung có chiều dài 40 tương đương với lấy 40 mẫu tần số trong khoảng [0,2π). Dải thông có độ rộng là 0,3π tương đương với 7 mẫu nhận giá trị 1. Giả sử tiếp rằng quá trình tối ưu hoá chỉ ra nên chọn trên cả 2 dải chuyển tiếp 2 mẫu nhận các giá trị $T_1 = 0{,}109021$ và $T_2 = 0{,}59417456$. Vậy dãy mẫu các tần số được cho như sau:

$$H(k) = \left\{ \underbrace{0,\ldots,0}_{5\ mẫu\ 0}, T_1, T_2, \underbrace{1,\ldots,1}_{7\ mẫu\ 1}, T_2, T_1, \underbrace{0,\ldots,0}_{9\ mẫu\ 0}, T_1, T_2, \underbrace{1,\ldots,1}_{7\ mẫu\ 1}, T_2, T_1, \underbrace{0,\ldots,0}_{4\ mẫu\ 0} \right\}$$

Viết chương trình tính và biểu diễn trên đồ thị:

a. Dãy các mẫu tần số

b. Dãy đáp ứng xung của bộ lọc thực tế

c. Hàm độ lớn tuyệt đối của đáp ứng tần số

d. Hàm độ lớn tương đối tính theo dB của đáp ứng tần số

2.11. Thiết kế bộ lọc thông thấp theo phương pháp lặp (thuật toán của Parks và McClellan) với các tham số đầu vào như sau:

$$\omega_p = 0{,}2\pi\,, \quad R_p = 0{,}25dB$$

$$\omega_s = 0{,}3\pi\,, \quad A_s = 50dB$$

Trước tiên xuất phát từ độ dài của dãy đáp ứng M theo công thức

$$M = \frac{-20\log\sqrt{\delta_1\delta_2} - 13}{14{,}6\Delta f} \qquad \text{,với } \Delta f = \frac{\omega_s - \omega_p}{2\pi}$$

Lặp công việc tìm bộ lọc tối ưu theo nghĩa Chebyshev và tăng M sau mỗi lần lặp để tìm ra bộ lọc thoả mãn yêu cầu thiết kế, sau đó tính và biểu diễn trên đồ thị:

a. Dãy đáp ứng xung của bộ lọc thực tế

b. Hàm độ lớn tuyệt đối của đáp ứng tần số

c. Hàm độ lớn tương đối tính theo dB của đáp ứng tần số

d. Hàm sai số $E(\omega)$

theo chương trình mẫu bằng cách gõ các dòng lệnh cho ở bảng dưới đây vào cửa số soạn thảo (Editor) và ghi lại theo tên tệp là **Solution_2_11.m**

```
wp = 0.2*pi; ws =0.3*pi; Rp = 0.25; As = 50;
delta_w = 2*pi/1000;
wsi = ws/delta_w+1;
delta1 = (10^(Rp/20)-1)/(10^(Rp/20)+1);
delta2 = (1+delta1)*(10^(-As/20));
deltaH = max(delta1,delta2);
deltaL = min(delta1,delta2);
weights = [delta2/delta1 1];
deltaf = (ws-wp)/(2*pi);
M = ceil((-20*log10(sqrt(delta1*delta2))-
13)/(14.6*deltaf)+1)
f = [0 wp/pi ws/pi 1];
m = [1 1 0 0];
h = firpm(M-1,f,m,weights);
[db,mag,pha,grd,w] = freqz_m(h,[1]);
Asd = -max(db(wsi:1:501))
while Asd<As
    M = M+1
    [h,ERR,RES] = firpm(M-1,f,m,weights);
    [db,mag,pha,grd,w] = freqz_m(h,[1]);
    Asd = -max(db(wsi:1:501))
end
%plot
n = [0:1:M-1];
subplot(2,2,1); stem(n,h);
axis([0,M-1,-0.1,0.3]);
title('Impulse Response');
xlabel('n'); ylabel('h(n)');
%
subplot(2,2,2); plot(w/pi,db); grid;
axis([0,1,-80,10]);
title('Magnitude Response in dB');
xlabel('frequency in pi units'); ylabel('Decibels');
%
subplot(2,2,3); plot(w/pi,mag); grid;
axis([0,1,-0.2,1.2]);
title('Magnitude Response');
xlabel('frequency in pi units'); ylabel('Hr(w)');
%
subplot(2,2,4); plot(RES.fgrid,RES.error); grid;
axis([0,1,-0.0150,0.0150]);
title('Error Response');
xlabel('frequency in pi units'); ylabel('Er(w)');
```

Gõ lệnh **Solution_2_11** tại cửa sổ lệnh của MATLAB để chạy kịch bản nói trên và xem các kết quả hiển thị ở cửa số lệnh (Command Window) và trên đồ thị.

2.12. Thiết kế bộ lọc thông dải theo phương pháp lặp (thuật toán của Parks và McClellan) với các tham số đầu vào như sau:

$$\omega_{s1} = 0{,}2\pi\,, \quad A_s = 60dB$$

$$\omega_{p1} = 0{,}35\pi\,, \quad R_p = 1dB$$

$$\omega_{p2} = 0{,}65\pi\,, \quad R_p = 1dB$$

$$\omega_{s2} = 0{,}8\pi\,, \quad A_s = 60dB$$

Viết chương trình lặp công việc tìm bộ lọc tối ưu theo nghĩa Chebyshev và tăng M sau mỗi lần lặp để tìm ra bộ lọc thoả mãn yêu cầu thiết kế, sau đó tính và biểu diễn trên đồ thị:

a. Dãy đáp ứng xung của bộ lọc thực tế
b. Hàm độ lớn tuyệt đối của đáp ứng tần số
c. Hàm độ lớn tương đối tính theo dB của đáp ứng tần số
d. Hàm sai số $E(\omega)$

6. Mở rộng

Viết các chương trình mô phỏng thiết kế các bộ lọc FIR khác bao gồm: bộ lọc thông cao, bộ lọc thông dải, bộ lọc chắn dải, bộ vi phân, bộ biến đổi Hilbert theo cả 3 phương pháp.

B. THIẾT KẾ BỘ LỌC SỐ CÓ ĐÁP ỨNG XUNG CHIỀU DÀI VÔ HẠN (BỘ LỌC SỐ IIR)

1. Yêu cầu trước khi làm thí nghiệm

Sinh viên nắm vững các kiến thức về thiết kế bộ lọc tương tự và các định dạng của bộ lọc tương tự, cụ thể:

- Biến đổi Laplace, hàm truyền đạt của hệ thống tương tự
- Các định dạng mẫu (prototype) của bộ lọc tương tự: Butterworth, Chebyshev I, Chebyshev II, Elliptic.

Sinh viên nắm vững các phương pháp chuyển đổi từ hệ thống tương tự sang hệ thống số bao gồm:

- Phương pháp bất biến xung
- Phương pháp biến đổi song tuyến
- Phương pháp tương đương vi phân
- Phương pháp biến đổi Z thích ứng.

Đồng thời sinh viên cũng cần nắm vững các kiến thức trong phép ánh xạ hàm biến phức để chuyển đổi băng tần số.

2. Mục đích của phần thí nghiệm

Sinh viên dùng MATLAB mô phỏng các nội dung sau:

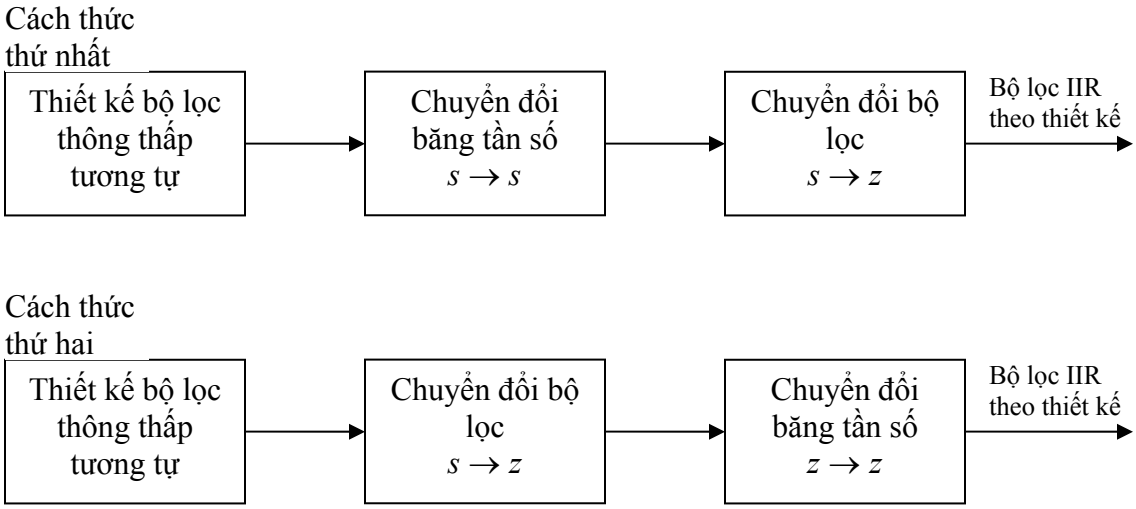
- Thiết kế bộ lọc thông thấp tương tự theo định dạng Chebyshev I
- Chuyển đổi bộ lọc thông thấp tương tự sang bộ lọc thông thấp số
- Chuyển đổi bộ lọc thông thấp sang bộ lọc thông dải.

3. Tóm tắt lý thuyết

Bộ lọc số FIR có ưu điểm nổi bật là pha tuyến tính. Nói một cách khác, bộ lọc FIR pha tuyến tính đảm bảo được cùng một độ trễ với các nhóm tần số, mỗi nhóm là một tập hợp các tần số lân cận nào đó. Thực nghiệm cho thấy tai người về phần nào đó có khả năng nhận biết được trễ nhóm của tín hiệu âm thanh. Bộ lọc có đáp ứng xung chiều dài vô hạn, hay bộ lọc số IIR, không đảm bảo được tính chất này.

Trong những trường hợp pha tuyến tính không phải là yêu cầu bắt buộc trong thiết kế thì việc lựa chọn bộ lọc FIR hay bộ lọc IIR đều được chấp nhận. Tuy nhiên, bộ lọc IIR thường được lựa chọn hơn vì một số lý do. Thứ nhất, nếu có cùng yêu cầu về độ suy giảm thì bộ lọc IIR đơn giản hơn nhiều so với bộ lọc FIR dẫn đến số phép tính để thực hiện trong bộ lọc IIR ít hơn bộ lọc FIR và các phần tử nhớ trong kết cấu của bộ lọc IIR sẽ ít hơn bộ lọc FIR. Thứ hai, bộ lọc IIR được thiết kế thông qua việc chuyển đổi các thiết kế của bộ lọc tương tự sang bộ lọc số và rất may mắn là các bảng thông số trong thiết kế bộ lọc số có thể tra được trong rất nhiều các tài liệu.

Bộ lọc số IIR, trên nguyên tắc là chuyển đổi từ thiết kế của bộ lọc tương tự thông qua một trong một số phương pháp chuyển đổi bộ lọc. Các phương pháp chuyển đổi sẽ được trình bày tóm tắt trong phần này. Một mặt, các định dạng có sẵn và các bảng tra cho chúng chỉ áp dụng đối với bộ lọc thông thấp. Do đó, để có được kết quả cuối cùng là bộ lọc số với các loại khác, ví dụ bộ lọc thông dải, quá trình thiết kế cần có một bước để thực hiện việc chuyển đổi băng tần số theo1 trong 2 cách tiếp cận: hoặc chuyển đổi băng tần số tương tự, hoặc chuyển đổi băng tần số số. Nói chung con đường để đi đến một thiết kế bộ lọc số IIR có 2 cách thức được đưa ra ở hình vẽ dưới đây.



Trong phạm vi phần thí nghiệm này, chúng ta sẽ tiến hành các bước theo cách tiếp cận thứ hai bao gồm các bước:

- Thiết kế bộ lọc thông thấp tương tự
- Chuyển đổi từ bộ lọc thông thấp tương tự sang bộ lọc thông thấp số
- Chuyển đổi băng tần số để thu được các bộ lọc khác từ bộ lọc thông thấp.

1. Thiết kế bộ lọc tương tự

Các yêu cầu thiết kế cho bộ lọc tương tự dựa trên điều kiện giới hạn các chỉ tiêu kỹ thuật cho hàm bình phương biên độ của hàm đáp ứng tần số. Giống như khi chúng ta quan tâm tới các hệ thống số, hàm truyền đạt $H_a(s)$ của một hệ thống tuyến tính bất biến tương tự là biến đổi Laplace hàm đáp ứng xung $h_a(t)$ của hệ thống và $H_a(s)$ chính bằng tỷ số giữa biến đổi Laplace của tín hiệu đầu ra với biến đổi Laplace của tín hiệu đầu vào. Hàm đáp ứng tần số $H_a(j\Omega)$ của một hệ thống tuyến tính bất biến là hàm thể hiện đáp ứng của hệ thống với đầu vào là các giá trị tần số khác nhau. Do đó, hàm đáp ứng tần số là biến đổi Fourier hàm đáp ứng xung $h_a(t)$ của hệ thống và hàm $H_a(j\Omega)$ cũng chính là hàm hệ thống $H_a(s)$ đánh giá trên trục ảo.

Đối với các hệ thống thực hiện được về mặt vật lý, đáp ứng xung bao giờ cũng là một hàm thực. Do đó, hàm truyền đạt luôn đối xứng qua trục thực. Mặt khác, một hệ thống tương tự là nhân quả và ổn định nếu và chỉ nếu tất cả các điểm cực của hàm truyền đạt nằm ở nửa bên trái của mặt phẳng s hoặc cùng lắm là nằm ở gốc toạ độ. Khi ta xét đến hàm bình phương biên độ, hay bình phương môđun, của hàm hệ truyền đạt, các điểm cực của hàm số này sẽ phân bố trên tất cả các góc phần tư của mặt phẳng S. Lúc này, việc xét đáp ứng tần số của hệ thống cũng thuận tiện và tất cả các điểm cực nằm bên trái mặt phẳng S của hàm $|H_a(s)|^2$.

Yêu cầu về chỉ tiêu kỹ thuật của bộ lọc thông thấp tương tự thường được cho dưới dạng các tham số tuyệt đối như sau:

$$\frac{1}{1+\varepsilon^2} \le \left|H_a\left(j\Omega\right)\right|^2 \le 1, \quad \left|\Omega\right| \le \Omega_p$$

$$0 \le \left|H_a(j\Omega)\right|^2 \le \frac{1}{A^2}, \quad |\Omega| \le \Omega_p$$

với Ω_p và Ω_s lần lượt là các tần số cắt dải thông và tần số cắt dải chắn tính theo đơn vị rad/s, ε là tham số gợn sóng và A là tham số suy giảm.

Mối quan hệ giữa ε và A với R_p và A_s hay δ_1 và δ_2 như sau:

$$R_p = -10\log\frac{1}{1+\varepsilon^2} \Rightarrow \varepsilon = \sqrt{10^{\frac{R_p}{10}} - 1}$$

$$A_s = -10\log\frac{1}{A^2} \Rightarrow A = \sqrt{10^{\frac{A_s}{10}}}$$

$$\frac{1-\delta_1}{1+\delta_1} = \sqrt{\frac{1}{1+\varepsilon^2}} \Rightarrow \varepsilon = \frac{2\sqrt{\delta_1}}{1-\delta_1}$$

$$\frac{\delta_2}{1+\delta_1} = \frac{1}{A} \Rightarrow A = \frac{1+\delta_1}{\delta_2}$$

Tất cả các định dạng bộ lọc đều dựa trên nguyên tắc lựa chọn hàm đáp ứng tần số của bộ lọc thực tế xấp xỉ với đáp ứng tần số của bộ lọc lý tưởng và điểm cực của hàm đáp ứng tần số của bộ lọc thực tế được phân bố sao cho hệ thống là nhân quả và ổn định. Các hàm số bình phương biên độ thường được lựa chọn có dạng gợn sóng vừa phải trong khoảng từ 0 đến tần số cắt và giảm mạnh khi vượt ra ngoài tần số cắt đồng thời có xu hướng giảm về đến 0. Điều này tương đương với hàm gợn sóng bị chặn trong khoảng từ -1 đến 1 (với 1 là tần số đã được chuẩn hoá bởi tần số cắt Ω_c) và tăng nhanh khi vượt ra ngoài 1. Có 4 định dạng cơ bản thường được vận dụng trong quá trình thiết kế bộ lọc tương tự là: bộ lọc Butterworth, bộ lọc Chebyshev-1, bộ lọc Chebyshev-2 và bộ lọc Elliptic.

a. Bộ lọc thông thấp Butterworth:

Hàm bình phương biên độ của đáp ứng tần số bộ lọc Butterworth bậc N được cho bởi phương trình:

$$\left|H_a(j\Omega)\right|^2 = \frac{1}{1+\left(\frac{\Omega}{\Omega_c}\right)^{2N}}$$ với Ω_c là tần số cắt

Các điểm cực của hàm bình phương biên độ của hàm truyền đạt $|H_a(s)|^2$ là :

$$s_{pk} = \Omega_c e^{j\frac{\pi}{2N}(2k+1+N)}$$

Dẫn đến các điểm cực của hàm truyền đạt $H_a(s)$ là N điểm nằm trên nửa đường tròn tâm O bán kính Ω_c ở nửa bên trái mặt phẳng S và N điểm này đối xứng qua trục thực.

Giá trị thích hợp của bậc bộ lọc thông thấp Butterworth được tính theo công thức sau:

$$N = \left\lceil \frac{\log\left[\left(10^{\frac{R_p}{10}} - 1\right)\left(10^{\frac{A_s}{10}} - 1\right)\right]}{2\log\left(\Omega_s / \Omega_p\right)} \right\rceil$$

(giá trị nguyên nhỏ nhất lớn hơn hoặc bằng biểu thức trong dấu $\lceil \ \rceil$)

b. Bộ lọc thông thấp Chebyshev-1:

Hàm bình phương biên độ của đáp ứng tần số bộ lọc Chebyshev-1 bậc N được cho bởi phương trình:

$$\left|H_a\left(j\Omega\right)\right|^2 = \frac{1}{1 + \varepsilon^2 T_N{}^2\left(\frac{\Omega}{\Omega_c}\right)}$$

với Ω_c là tần số cắt, ε là tham số gợn sóng dải thông, $T_N(x)$ là đa thức Chebyshev bậc N được cho bởi công thức:

$$T_N\left(x\right) = \begin{cases} \cos\left[N\cos^{-1}\left(x\right)\right] & -1 \le x \le 1 \\ \cosh\left[N\cosh^{-1}\left(x\right)\right] & \left|x\right| > 1 \end{cases}$$

Các điểm cực của hàm bình phương biên độ của hàm truyền đạt $|H_a(s)|^2$ có dạng:

$$s_{pk} = \sigma_k + j\Omega_k$$

$$\sigma_k = \left(a\Omega_c\right)\cos\left[\frac{\pi}{2} + \frac{\left(2k+1\right)\pi}{2N}\right]$$

$$\Omega_k = \left(b\Omega_c\right)\sin\left[\frac{\pi}{2} + \frac{\left(2k+1\right)\pi}{2N}\right]$$

với $a = \frac{1}{2}\left(\sqrt[N]{\alpha} - \sqrt[N]{1/\alpha}\right)$, $b = \frac{1}{2}\left(\sqrt[N]{\alpha} + \sqrt[N]{1/\alpha}\right)$, và $\alpha = \frac{1}{\varepsilon} + \sqrt{1 + \frac{1}{\varepsilon^2}}$

Hay 2N điểm cực của hàm bình phương biên độ hàm thống phân bố đều trên một đường ellip có các bán kính là ($a\Omega_c$) và ($b\Omega_c$). Dẫn đến các điểm cực của hàm truyền đạt $H_a(s)$ là N điểm nằm trên nửa đường elip ở nửa bên trái mặt phẳng S và N điểm này đối xứng qua trục thực.

Giá trị thích hợp của bậc bộ lọc thông thấp Chebyshev-1 được tính theo công thức sau:

$$N = \left\lceil \frac{\log\left(g + \sqrt{g^2 - 1}\right)}{\log\left(\Omega_r + \sqrt{\Omega_r{}^2 - 1}\right)} \right\rceil$$

với $g = \sqrt{\left(A^2 - 1\right) / \varepsilon^2}$ và $\Omega_r = \frac{\Omega_s}{\Omega_p}$

c. Bộ lọc thông thấp Chebyshev-2:

Hàm bình phương biên độ của đáp ứng tần số bộ lọc Chebyshev-2 bậc N được cho bởi phương trình:

$$\left|H_a\left(j\Omega\right)\right|^2 = \frac{1}{1 + \left[\varepsilon^2 T_N{}^2\left(\frac{\Omega}{\Omega_c}\right)\right]^{-1}}$$

với Ω_c là tần số cắt, ε là tham số gợn sóng dải thông, $T_N(x)$ là đa thức Chebyshev bậc N.

Giá trị thích hợp của bậc bộ lọc thông thấp Chebyshev-2 đươc tính giống theo công thức đã cho với bộ lọc Chebyshev-1.

d. Bộ lọc thông thấp Elliptic:

Hàm bình phương biên độ của đáp ứng tần số bộ lọc Elliptic bậc N được cho bởi phương trình:

$$\left|H_a\left(j\Omega\right)\right|^2 = \frac{1}{1 + \varepsilon^2 U_N{}^2\left(\frac{\Omega}{\Omega_c}\right)}$$

với Ω_c là tần số cắt, ε là tham số gợn sóng dải thông, $U_N(x)$ là đa thức elliptic Jacobian bậc N.

Giá trị thích hợp của bậc bộ lọc thông thấp Elliptic theo các tài liệu được tính theo công thức sau:

$$N = \left\lceil \frac{K(k)}{K\left(\sqrt{1 - k^2}\right)} \frac{K\left(\sqrt{1 - k_1{}^2}\right)}{K(k_1)} \right\rceil$$

với $k = \frac{\Omega_p}{\Omega_s}$ và $k_1 = \varepsilon / \sqrt{\left(A^2 - 1\right)}$

K(x) là hàm số cho bởi biểu thức tích phân $K(x) = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - x^2 \sin^2\theta}}$

MATLAB cung cấp hàm **ellipke** để tính gần đúng biểu thức của K(x) cho bởi phương trình trên.

2. Chuyển đổi bộ lọc:

Trên nguyên tắc, việc chuyển đổi bộ lọc tập trung vào nghiên cứu các phép biến hình, hay ánh xạ, để chuyển đổi mặt phẳng s về mặt phẳng z. Trên lý thuyết có một số phương pháp chuyển đổi sau đây.

a. Phương pháp bất biến xung (Impulse Invariance Transformation):

Bản chất phương pháp bất biến xung là phép biến hình sao cho dãy đáp ứng xung của bộ lọc số chính là hàm đáp ứng xung của bộ lọc tương tự được lấy mẫu ở các điểm rời rạc. Phép biến hình cho ta công thức đổi biến:

$$z = e^{sT}$$

với: z là biến số độc lập của hàm H(z) trên miền Z

s là biến số độc lập của hàm $H_a(s)$ trên miền S

T là chu kỳ lấy mẫu của hàm đáp ứng xung hệ thống tương tự

Mối quan hệ giữa hàm truyền đạt H(z) ở miền Z và hàm truyền đạt $H_a(s)$ ở miền s được cho bởi công thức sau:

$$H(z) = \frac{1}{T}\sum_{k=-\infty}^{\infty} H_a\left(s - j\frac{2\pi}{T}k\right)$$

- Các nửa sọc ngang dài vô hạn có bề rộng $\frac{2\pi}{T}$ và nằm ở nửa bên trái mặt phẳng S được ánh xạ vào bên trong đường tròn đơn vị trên mặt phẳng Z theo nguyên tắc nhiều - một.
- Bởi phép biến hình ánh xạ toàn bộ nửa mặt phẳng bên trái của mặt phẳng S vào bên trong đường tròn đơn vị của mặt phẳng Z nên nó bảo toàn tính ổn định của hệ thống (dựa trên phân bố của các điểm cực).
- Nếu như bộ lọc tương tự là thông thấp lý tưởng và chu kỳ lấy mẫu đủ nhỏ để:

$$H_a(j\Omega) = H_a(j\omega T) = 0, \forall|\Omega| \geq \frac{\pi}{T} \quad \text{thì} \quad H\left(e^{j\omega}\right) = \frac{1}{T}H_a\left(\frac{j\omega}{T}\right), |\omega| \leq \pi$$

thì không có hiện tương chồng phổ (aliasing). Tuy nhiên bộ lọc thông thấp thực tế không thể có phổ hữu hạn nên hiện tượng chồng phổ gây ra bởi phép biến hình vẫn xảy ra.

b. Phương pháp biến đổi song song tuyến (Bilinear Transformation)

Bản chất của phép biến đổi song tuyến là phép biến hình dựa trên nguyên tắc đưa phương trình vi phân tuyến tính hệ số hằng đặc trưng cho một hệ thống tương tự về gần đúng một phương trình sai phân tuyến tính hệ số hằng, mà phương trình sau có thể đặc trưng cho một hệ thống số. Phép biến hình cho ta công thức đổi biến:

$$s = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}} \Rightarrow z = \frac{1+sT/2}{1-sT/2}$$

- Phép biến hình này ánh xạ toàn bộ nửa bên trái mặt phẳng S vào bên trong đường tròn đơn vị trên mặt phẳng Z trên nguyên tắc một - một nên nó bảo toàn tính ổn định của hệ thống.
- Mặt khác nguyên tắc ánh xạ một - một từ mặt phẳng S đến mặt phẳng Z cho phép hoàn toàn không xảy ra hiện tượng chồng phổ.

c. Phương pháp tương đương vi phân (Approximation of Derivatives Transformation)

Phương pháp này dựa trên việc thiết lập một sự tương ứng giữa định nghĩa của vi phân và định nghĩa của sai phân. Phép biến hình cho ta công thức đổi biến:

$$s = \frac{1-z^{-1}}{T} \Rightarrow z = \frac{1}{1-sT}$$

- Phép biến hình ánh xạ toàn bộ nửa bên trái mặt phẳng S vào bên trong đường tròn tâm (½,0) bán kính R = ½.
- Phép biến hình ánh xạ toàn bộ nửa bên trái mặt phẳng S vào bên trong đường tròn đơn vị trên mặt phẳng Z nên nó bảo toàn tính ổn định của hệ thống.
- Tuy nhiên tập hợp các điểm cực của hệ thống bị co lại trong một phạm vi nhỏ nên có thể dẫn tới hiện tượng cộng hưởng ở phạm vi tần số nào đó.

d. Phương pháp biến đổi Z thích ứng (Matched-Z Transformation)

Phương pháp này dựa trên nguyên tắc ánh xạ trực tiếp các điểm cực và điểm không của hàm truyền đạt hệ thống tương tự thành các điểm cực và điểm không của hàm truyền đạt hệ thống số. Giả sử hàm truyền đạt của hệ thống tương tự có dạng:

$$H_a(s) = C\frac{\prod_{r=1}^{M}(s-s_{0r})}{\prod_{k=1}^{M}(s-s_{pk})}$$

thì phép biến hình biến đổi các phần tử $(s-a)$ trở thành $1-e^{aT}z^{-1}$ thu được hàm truyền đạt của hệ thống số

$$H(z) = C\frac{\prod_{r=1}^{M}\left(1-e^{s_{0r}T}z^{-1}\right)}{\prod_{k=1}^{M}\left(1-e^{s_{pk}T}z^{-1}\right)}$$

Với phương pháp này phải chọn chu kỳ lấy mẫu T đủ nhỏ để các điểm cực và điểm không phân bố một cách thích hợp trên mặt phẳng Z, tránh hiện tượng chồng phổ, từ đó đảm bảo được đáp ứng tần số của bộ lọc số gần giống với đáp ứng tần số của bộ lọc tương tự.

3. Chuyển đổi băng tần số

Việc chuyển đổi băng tần số, xuất phát từ bộ lọc thông thấp có tần số cắt ω_c', được đưa ra theo các công thức ánh xạ ở bảng dưới đây:

Loại chuyển đổi	**Công thức chuyển đổi**	**Các tham số**
Thông thấp	$z^{-1} \to \dfrac{z^{-1}-\alpha}{1-\alpha z^{-1}}$	ω_c : Tần số cắt của bộ lọc mới $\alpha = \dfrac{\sin\left(\dfrac{\omega_c^{'}-\omega_c}{2}\right)}{\sin\left(\dfrac{\omega_c^{'}+\omega_c}{2}\right)}$
Thông cao	$z^{-1} \to \dfrac{z^{-1}+\alpha}{1+\alpha z^{-1}}$	ω_c : Tần số cắt của bộ lọc mới $\alpha = -\dfrac{\cos\left(\dfrac{\omega_c^{'}-\omega_c}{2}\right)}{\cos\left(\dfrac{\omega_c^{'}+\omega_c}{2}\right)}$
Thông dải	$z^{-1} \to \dfrac{z^{-2}-2\beta\dfrac{k}{k+1}z^{-1}+\dfrac{k-1}{k+1}}{\dfrac{k-1}{k+1}z^{-2}-2\beta\dfrac{k}{k+1}z^{-1}+1}$	ω_{c1} : Tần số cắt thấp của bộ lọc mới ω_{c2} : Tần số cắt cao của bộ lọc mới $\beta = \dfrac{\cos\left(\dfrac{\omega_{c2}+\omega_{c1}}{2}\right)}{\cos\left(\dfrac{\omega_{c2}-\omega_{c1}}{2}\right)}$ $k = \tan\left(\dfrac{\omega_c^{'}}{2}\right)\cot\left(\dfrac{\omega_{c2}-\omega_{c1}}{2}\right)$
Chắn dải	$z^{-1} \to \dfrac{z^{-2}-2\beta\dfrac{k}{k+1}z^{-1}+\dfrac{k-1}{k+1}}{\dfrac{k-1}{k+1}z^{-2}-2\beta\dfrac{k}{k+1}z^{-1}+1}$	ω_{c1} : Tần số cắt thấp của bộ lọc mới ω_{c2} : Tần số cắt cao của bộ lọc mới $\beta = \dfrac{\cos\left(\dfrac{\omega_{c2}+\omega_{c1}}{2}\right)}{\cos\left(\dfrac{\omega_{c2}-\omega_{c1}}{2}\right)}$ $k = \tan\left(\dfrac{\omega_c^{'}}{2}\right)\tan\left(\dfrac{\omega_{c2}-\omega_{c1}}{2}\right)$

4. Một số lệnh và hàm của MATLAB

Phần này đưa ra danh mục các lệnh các hàm của MATLAB có thể sử dụng trong phần thí nghiệm này. Để biết cụ thể hơn về chức năng của hàm và cú pháp của lệnh gọi hàm, gõ lệnh **help** kèm theo tên của hàm tại cửa số lệnh của MATLAB.

`freqs`: trả về đáp ứng tần số của một hệ thống tương tự khi biết hàm truyền đạt được cho dưới dạng phân thức hữu tỷ của nó

`impluse`: trả về đáp ứng xung của một hệ thống tương tự khi biết hàm truyền đạt được cho dưới dạng phân thức hữu tỷ của nó

`buttap, cheb1ap, cheb2ap, ellipap`: trả về các điểm không, điểm cực, và độ lợi trong thiết kế của một hàm truyền đạt bộ lọc thông thấp bậc N, tần số cắt đã được chuẩn hoá bằng 1 với các định dạng lần lượt là Butterworth, Chebyshev-I, Chebyshev-II, và Elliptic

`impinvar, bilinear`: trả về các hệ số của đa thức tử số và đa thức mẫu số hàm truyền đạt của hệ thống số xuất phát từ hệ thông tương tự qua các phương pháp chuyển đổi lần lượt là phương pháp bất biến xung và phương pháp biến đổi song tuyến

`butter, cheby1, cheby2, ellip`: trực tiếp trả về các hệ số của đa thức tử số và đa thức mẫu số hàm truyền đạt của bộ lọc số dựa trên tham số đầu vào là các tần số cắt, phương pháp chuyển đổi được sử dụng trong các hàm này là phương pháp biến đổi song tuyến

Ngoại trừ hai hàm đầu tiên, các hàm còn lại đều nằm trong bộ công cụ ***Signal Processing Toolbox***. Chúng là các hàm được tạo sau này dựa trên các hàm nguyên thể ban đầu của MATLAB nhằm phục vụ tính toán và mô phỏng trong ***Xử lý số tín hiệu***. Ngoài ra, còn một loạt các hàm khác mạnh hơn trong bộ công cụ này có thể hỗ trợ thiết kế dựa trên các chỉ tiêu kỹ thuật đầu vào và đưa ra kết quả với bậc của bộ lọc N được lựa chọn tối ưu. Trong phạm vi các bài thí nghiệm nhằm mục đích hiểu rõ cơ chế thiết kế bộ lọc, không cần thiết phải sử dụng các hàm đó mà có thể từng bước tự thiết kế và thực hiện chúng.

5. Các bước thực hành

2.13. Hàm **freqs** của MATLAB trả về đáp ứng tần số của một hệ thống tương tự khi biết trước hệ số của đa thức tử số và đa thức mẫu số của hàm truyền đạt $H_a(s)$. Trong nhiều trường hợp, để thuận tiện ta cần tìm thêm các thông số: hàm độ lớn của đáp ứng tần số, hàm pha của đáp ứng tần số, hàm trễ nhóm, thể hiện độ lớn theo thang decibels. Tạo hàm tính đáp ứng tần số có tên sau **freqs_m** nhằm tính các thông số trên theo chương trình mẫu bằng cách gõ các dòng lệnh cho ở bảng dưới đây vào cửa số soạn thảo (Editor) và ghi lại theo tên tệp là **freqs_m.m**:

```
function [db,mag,pha,w] = freqs_m(b,a,wmax);
% Modified version of freqs subrountine
% ------------------------------------
% [db,mag,pha,w] = freqs_m(b,a,wmax)
%   db = Do lon tuong doi theo dB tren doan tu 0 den wmax
%  mag = Do lon tuyet doi tren doan tu 0 den wmax
%  pha = Dap ung pha tren doan tu 0 den wmax
%   w = Cac mau tan so tren doan tu 0 den wmax
%   b = Cac he so da thuc tu so cua Ha(s)
%   a = Cac he so da thuc mau so cua Ha(s)
% wmax = Tan so cuc dai theo don vi rad/sec tren doan
% tan so mong muon tim dap ung tan so
%
w = [0:1:500]*wmax/500;
H = freqs(b,a,w);
mag = abs(H);
db = 20*log10((mag+eps)/max(mag));
pha = angle(H);
```

2.14. Hàm **cheb1ap** trả về danh sách các điểm không, điểm cực và độ lợi của hàm truyền đạt cho thiết kế bộ lọc dạng Chebyshev I, tần số cắt đã được chuẩn hoá. Tạo hàm có tên sau **u_chb1ap** nhằm trả về hệ số của các đa thức tử số và đa thức mẫu số của hàm truyền đạt cho thiết kế bộ lọc dạng Chebyshev I có tần số cắt tuỳ ý theo chương trình mẫu bằng cách gõ các dòng lệnh cho ở bảng dưới đây vào cửa sổ soạn thảo (Editor) và ghi lại theo tên tệp là **u_chb1ap.m**:

```
function [b,a] = u_chb1ap(N,Rp,Omegac)
% Bo loc thong thap dang Chebyshev-1
% tan so cat khong duoc chuan hoa
% ------------------------------------------------
% [b,a] = u_chb1ap(N,Rp,Omegac)
%      b = cac he so da thuc tu so cua Ha(s)
%      a = cac he so da thuc mau so cua Ha(s)
%      N = Bac cua bo loc Chebyshev-I
%     Rp = Do gon dai thong theo don vi dB; Rp > 0
% Omegac = tan so cat theo don vi radians/sec
%
```

```
[z,p,k] = cheb1ap(N,Rp);
a = real(poly(p));
aNn = a(N+1);
p = p*Omegac;
a = real(poly(p));
aNu = a(N+1);
k = k*aNu/aNn;
B = real(poly(z));
b0 = k;
b = k*B;
```

2.15. Hàm số mô tả ở trên trả về hàm truyền đạt với bậc N cho trước. Bậc của bộ lọc có thể lựa chọn cho phù hợp tối ưu với các chỉ tiêu kỹ thuật yêu cầu đầu vào. Tạo hàm trả về thiết kế bộ lọc thông thấp tương tự, định dạng Chebyshev có bậc tối ưu theo chương trình mẫu bằng cách gõ các dòng lệnh cho ở bảng dưới đây vào cửa số soạn thảo (Editor) và ghi lại theo tên tệp là **afd_chb1.m**:

```
function [b,a] = afd_chb1(Wp,Ws,Rp,As)
% Analog Lowpass Filter Design: Chebyshev-1
% ----------------------------------------
% [b,a] = afd_chb1(Wp,Ws,Rp,As)
%  b = cac he so da thuc tu so cua Ha(s)
%  a = cac he so da thuc mau so cua Ha(s)
% Wp = tan so cat dai thong theo don vi rad/sec; Wp >0
% Ws = tan so cat dai chan theo don vi rad/sec; Ws>Wp >0
% Rp = Do gon dai thong theo don vi dB; (Rp > 0)
% As = Do suy giam dai chan theo don vi +dB; (Ap > 0)
%
if Wp <= 0
    error('Tan so cat dai thong phai lon hon 0')
end
if Ws <= Wp
    error('Tan so cat dai thong phai lon hon tan so cat
dai chan')
end
if (Rp <= 0) | (As<0)
    error('Do gon dai thong va/hoac Do suy giam dai chan
phai lon hon 0')
end
%
```

```
ep = sqrt(10^(Rp/10)-1);
A = 10^(As/20);
OmegaC = Wp;
OmegaR = Ws/Wp;
g = sqrt(A*A-1)/ep;
N = ceil(log10(g+sqrt(g*g-
1))/log10(OmegaR+sqrt(OmegaR*OmegaR-1)));
fprintf('\n*** Bac cua bo loc Chebyshev-1 = %2.0f\n',N);
[b,a] = u_chb1ap(N,Rp,OmegaC);
```

2.16. Thiết kế bộ lọc thông thấp tương tự, định dạng Chebyshev-I, cửa số với các tham số đầu vào như sau:

$$\omega_p = 0,2\pi\,, \qquad R_p = 1dB$$

$$\omega_s = 0,3\pi\,, \qquad A_s = 16dB$$

Viết chương trình tính và biểu diễn trên đồ thị:

a. Độ lớn của đáp ứng tần số

b. Hàm đáp ứng pha của bộ lọc

c. Hàm độ lớn tương đối tính theo dB của đáp ứng tần số

d. Hàm đáp ứng xung của bộ lọc tương tự

2.17. Chuyển đổi bộ lọc với các tham số đã cho ở phần 2.16 sang bộ lọc số bằng phương pháp biến đổi song tuyến. Hàm **bilinear** cho phép thực hiện việc chuyển đổi này. Tính và biểu diễn trên đồ thị:

a. Độ lớn của đáp ứng tần số

b. Hàm đáp ứng pha của bộ lọc

c. Hàm độ lớn tương đối tính theo dB của đáp ứng tần số

d. Trễ nhóm theo tần số.

theo chương trình mẫu bằng cách gõ các dòng lệnh cho ở bảng dưới đây vào cửa số soạn thảo (Editor) và ghi lại theo tên tệp là **Solution_2_17.m**

```
% Chi tieu ky thuat cua bo loc so:
wp =0.2*pi;                    % digital Passband freq in
Hz
ws =0.3*pi;                    % digital Stopband freq in
Hz
Rp = 1;                        % Passband ripple in dB
As = 15;                       % Stopband attenuation in
dB
% Chi tieu ky thuat cua bo loc tuong tu: Anh xa nguoc
T = 1; Fs =1/T;                % Dat T=1
OmegaP = (2/T)*tan(wp/2);
OmegaS = (2/T)*tan(ws/2);
% Tinh toan bo loc tuong tu:
[cs, ds] = afd_chb1(OmegaP,OmegaS,Rp,As);
% Bien doi song tuyen:
[b,a] = bilinear(cs,ds,Fs);
%
[db,mag,pha,grd,w] = freqz_m(b,a);
%plot
figure(37); clf;
%
subplot(2,2,1); plot(w/pi,mag);
axis([0,1,0,1.2]); grid
title('Amplitude Response');
xlabel('frequency in pi units'); ylabel('|Hr(w)|');
%
subplot(2,2,3); plot(w/pi,db);
axis([0,1,-30,10]); grid
title('Magnitude Response');
xlabel('frequency in pi units'); ylabel('Decibels');
%
subplot(2,2,2); plot(w/pi,pha/pi);
axis([0,1,-1,1]); grid
title('Phase Response');
xlabel('frequency in pi units'); ylabel('Angle(Hr(w))');
%
subplot(2,2,4); plot(w/pi,grd);
axis([0,1,0,15]); grid
title('Group Delay');
xlabel('frequency in pi units'); ylabel('Samples');
```

Gõ lệnh **Solution_2_17** tại cửa sổ lệnh của MATLAB để chạy kịch bản nói trên và xem các kết quả hiển thị ở cửa số lệnh (Command Window) và trên đồ thị.

2.18. Thực hiện yêu cầu của câu 2.17 theo phương pháp bất biến xung, dùng hàm **impinvar** của MATLAB. So sánh kết quả thu được với câu trên.

2.19. Tạo hàm thực hiện việc chuyển đổi băng tần số, trả về hàm truyền đạt của bộ lọc mới với tham số đầu vào là hàm truyền đạt của bộ lọc thông thấp, hàm đa thức thể hiện phép đổi biến số độc lập theo chương trình mẫu bằng cách gõ các dòng lệnh cho ở bảng vào đây trong cửa sổ soạn thảo (Editor) và ghi lại theo tên tệp là **zmapping.m**:

```
% function [bz,az] = zmapping(bZ,aZ,Nz,Dz)
% Chuyen doi bang tan so tu mien Z sang mien z
% ------------------------------------------------------
% [bz,az] = zmapping(bZ,aZ,Nz,Dz)
% perform:
%         b(z)   b(Z)|
%         ---- = ----|    N(z)
%         a(z)   a(Z)|Z = ----
%                        D(z)
%
bzord = (length(bZ)-1)*(length(Nz)-1);
azord = (length(aZ)-1)*(length(Dz)-1);

bz = zeros(1,bzord+1);
for k = 0:bzord
    pln = [1];
    for l = 0:k-1
        pln = conv(pln,Nz);
    end
    pld = [1];
    for l = 0:bzord-k-1
        pld = conv(pld,Dz);
    end
    bz = bz+bZ(k+1)*conv(pln,pld);
end
%
az = zeros(1,azord+1);
for k = 0:azord
    pln = [1];
    for l = 0:k-1
        pln = conv(pln,Nz);
    end
    pld = [1];
    for l = 0:azord-k-1
        pld = conv(pld,Dz);
    end
    az = az+aZ(k+1)*conv(pln,pld);
end
%
az1 = az(1); az = az/az1; bz=bz/az1;
```

2.20. Viết chương trình chuyển đổi từ bộ lọc thông thấp theo thiết kế của câu 2.17 sang bộ lọc thông cao có tần số cắt $\omega_c=0,6\pi$. Tính và biểu diễn trên đồ thị:

a. Độ lớn của đáp ứng tần số

b. Hàm đáp ứng pha của bộ lọc

c. Hàm độ lớn tương đối tính theo dB của đáp ứng tần số

d. Trễ nhóm theo tần số.

6. Mở rộng

1. Viết chương trình mô phỏng thiết kế các bộ lọc thông thấp tương tự theo các định dạng khác bao gồm: Butterworth, Chebyshev-2, Elliptic.
2. Viết chương trình mô phỏng thiết kế việc chuyển đổi từ mộ lọc thông thấp số sang các bộ lọc số khác bao gồm: chuyển đổi thông thấp – thông thấp, chuyển đổi thông thấp – thông dải, và chuyển đổi thông thấp – chắn dải.

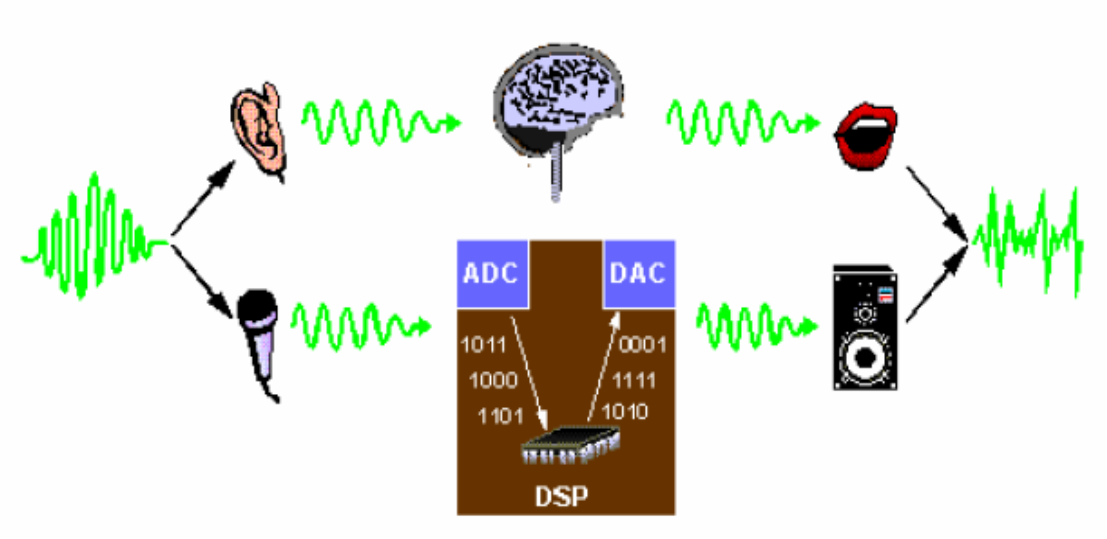
BÀI 3. GIỚI THIỆU VỀ DIGITAL SIGNAL PROCESSOR

1. Mục đích:

Kết thúc bài thí nghiệm này, sinh viên có thể giải thích sự khác nhau giữa một bộ xử lý tín hiệu số (DSP) và một bộ xử lý mục đích chung. Xa hơn một bước, sinh viên có thể làm quen với quá trình thiết kế cho các chương trình cho DSP.

2. Cơ sở lý thuyết.

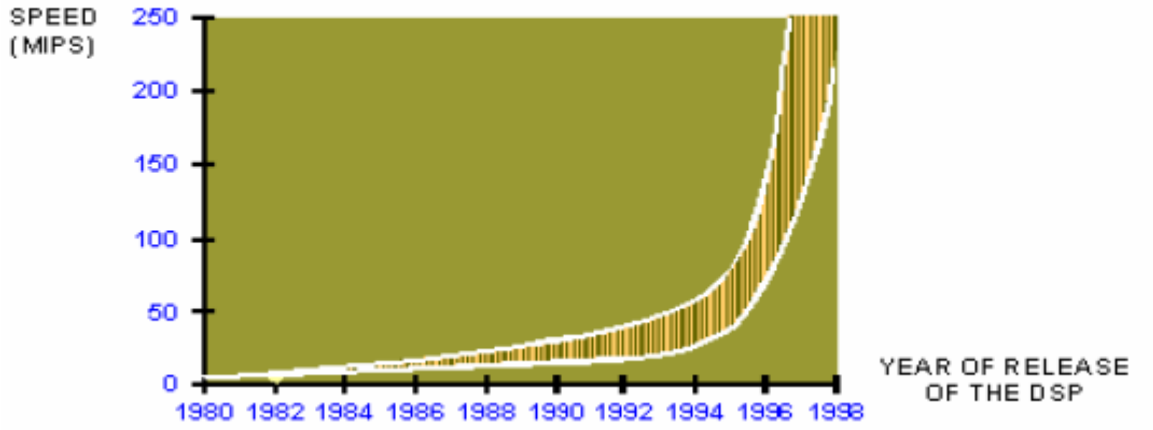
Bộ xử lý tín hiệu số (Digital Signal Processor - DSP) là một bộ phận xử lý mạnh và rất nhanh, nó có thể điều khiển quá trình phân tích tín hiệu trong thời gian thực. Bởi các phần tử khoá cho các mạch logic được thiết kế chuyên dụng cho các phép toán nhân và cộng nên thời gian tính toán trong các DSP nói chung thường nhanh hơn so với các bộ vi xử lý khác.



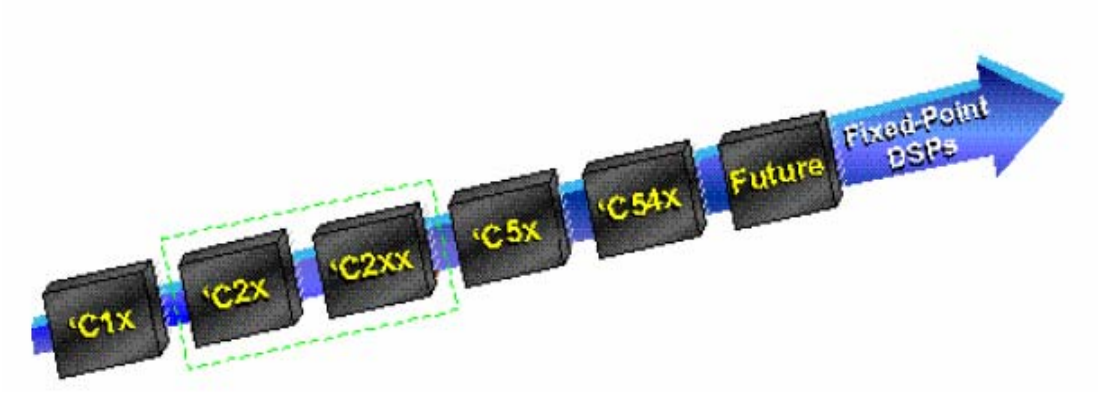
Các bộ xử lý tín hiệu số được đặc trưng bởi:

- Các cấu trúc chuyên môn hoá cho phép chúng thực hiện các lệnh mới một cách nhanh chóng và hiểu quả
- Các chỉ thị nhận nhanh
- Một số rút gọn các lệnh làm cho quá trình lập trình DSP đơn giản hơn

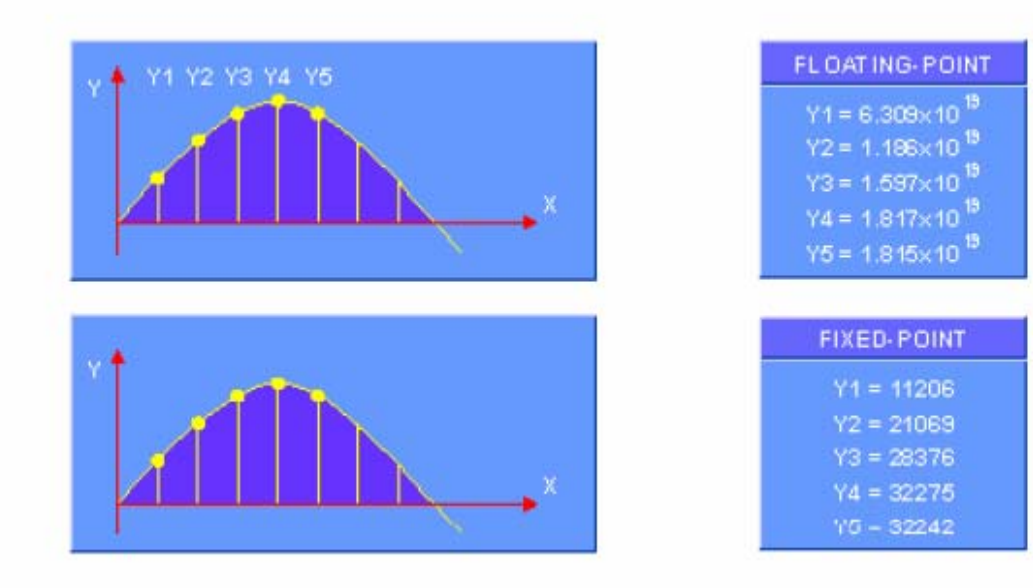
Các DSP đã làm cuộc cách mạng trong công nghệ điện tử viễn thông. DSP có thể coi như trái tim trong hàng loạt các thiết bị hiện đại như điện thoại di động, các thiết bị nhận dạng và tổng hợp tiếng nói, bộ chơi DVD (Digital Versatile), và các thiết bị an toàn mức cao. Không những vậy, rất nhiều ứng dụng ngày nay đã được tích hợp DSP như là trung tâm điều khiển của hệ thống bao gồm các bộ điều khiển đĩa cứng, các hệ thống treo xe ô tô, trong các mạng xử lý tín hiệu ảnh y tế, và các hệ thống radar.



DSP bắt đầu xuất hiện vào cuối những năm 1970 và vào đầu năm 1980 với DSP1 của Bell Lab, 2920 của Inlel, uPD7720 của NEC. Vào năm 1982, Texas Instrument đã đưa ra TMS32010, thành viên đầu tiên của họ DSP dấu phẩy tĩnh 16 bit. DSP này có tốc độ tính toán là 8MIPS. Các bước nhảy vọt liên tiếp xuất hiện. Cụ thể là vào năm 1998, các DSP sử dụng xử lý song song đã đạt tới tốc độ tính toán 1600MIPS.



Trong hệ thống thí nghiệm Lab-Volt DIGITAL SIGNAI PROCESSOR, loại DSP được sử dụng là Texas Instrument TMS320C50. Đây là loại DSP thế hệ thứ ba với thiết kế bên trong dựa trên DSP thế hệ thứ nhất TMS320C10.



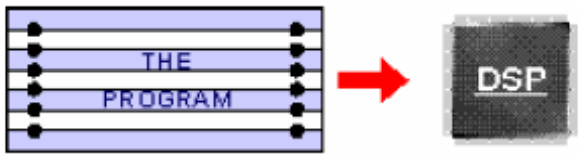
Cũng vào năm 1982, các bộ xử lý dấu phẩy động đầu tiên đã được sản xuất bởi Hitachi. Khuôn dạng số này tăng đáng kể khoảng tính toán động của DSP. Hai năm sau NEC đã đưa ra các DSP 32 bit dấu phẩy động đầu tiên có tốc độ tính toán 6,6MIPS.

Nói chung, các tín hiệu của thế giới thực (ví dụ: âm thanh, radar) được xử lý tốt hơn bằng các DSP dấu phẩy động. Các tín hiệu được xây dựng (ví dụ như: viễn thông, ảnh và điều khiển) nói chung được xử lý tốt hơn bằng các DSP dấu phẩy tĩnh .

Trên thế giới, xu thế phát triển các sản phẩm dựa trên DSP tăng nhanh vì:

- Chúng cho phép xử lý phức tạp hơn các mạng tương tự.
- Chúng cung cấp tính năng xử lý tín hiệu lặp đi lặp lại.
- Mã nguồn có thể dễ dàng được sửa đổi và việc cập nhật. Nói một cách khác, thay đổi thiết kế của nó là mềm dẻo hơn.

- Chúng thường được cho giá thành phát triển thấp hơn các thiết kế tương tự với các bậc tính năng tương đương.



Một hệ thống muốn vận hành cần phải thông qua sự chỉ thị từ một phần mềm được lập trình từ trước. Phần mềm bao gồm một tập các chỉ dẫn, hay còn gọi là các lệnh, để bảo cho hệ thống biết sẽ làm các công việc gì một cách tuần tự và hệ thống cần thao tác thế nào một khi có một điều kiện đã được dự đoán trước xảy ra.. Chương trình này được lưu trữ như mã máy bên trong DSP.

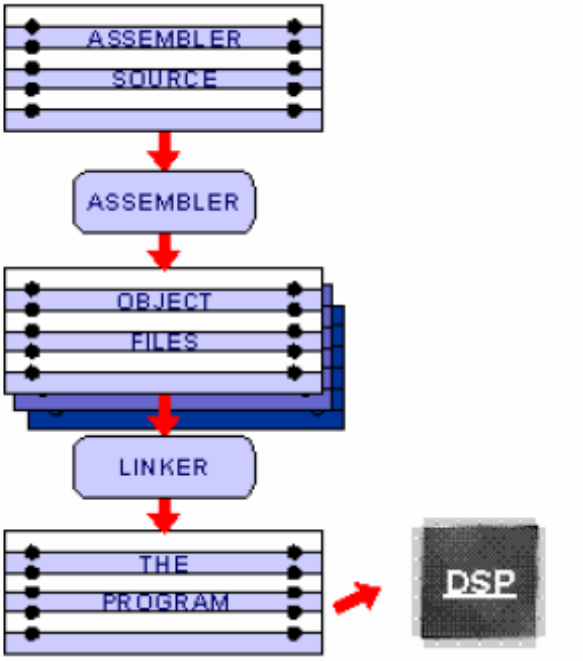
Hỏi: Lựa chọn nào trong các lựa chọn dưới đây là một lệnh nằm trong chương trình?

a. ADD #214, 4

b. F9E7h

c. 1011,1110 0001 0110

d. Tất cả các lựa chọn trên



Xây dựng một chương trình DSP mà đơn thuần từ mã máy là không khả thi. Vì lý do này, ngôn ngữ assembler (hợp ngữ) được phát triển để viết chương trình cho DSP. Đây là ngôn ngữ lập trình mà các chỉ thị của nó ở dạng gợi nhớ là biểu tượng và thường tương ứng một – một với các chỉ thị máy.

Bộ dịch (assembler) và bộ liên kết (linker) được sử dụng để dịch chương trình được viết bằng hợp ngữ thành các mã máy của DSP. Assembler dịch tệp chương trình thành tệp đích, các tệp này sau đó được liên kết với nhau (link) để tạo ra tệp mã máy vận hành bên trong DSP.

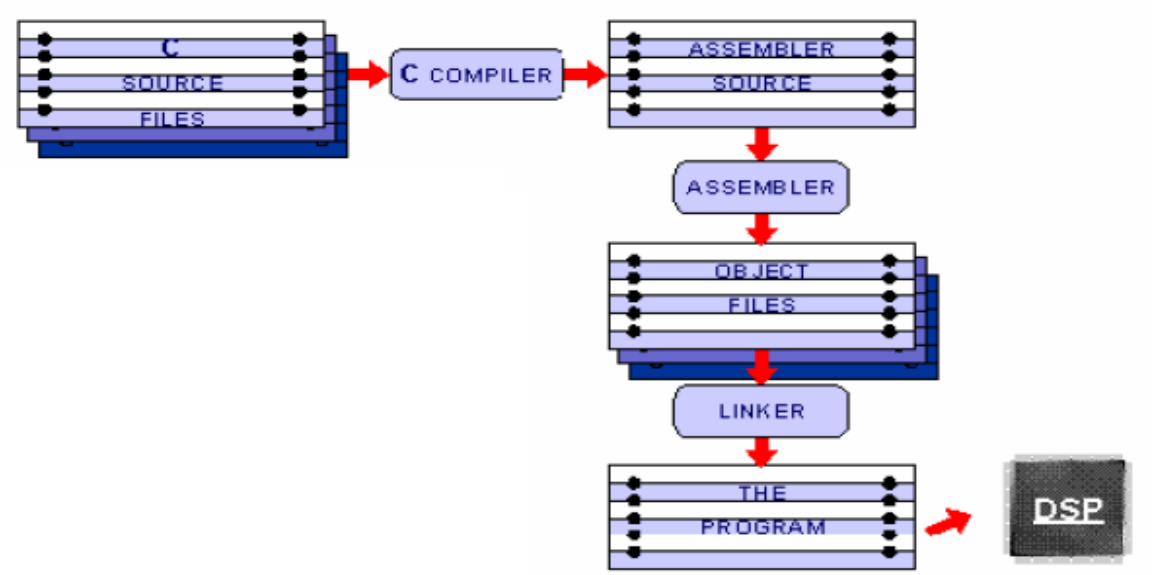
Hỏi: Sự lựa chọn nào trong các câu lệnh dưới đây được viết bằng hợp ngữ?

a. IF (i.NE.27) THEN (omega=2*sin(x))

b. 982Eh

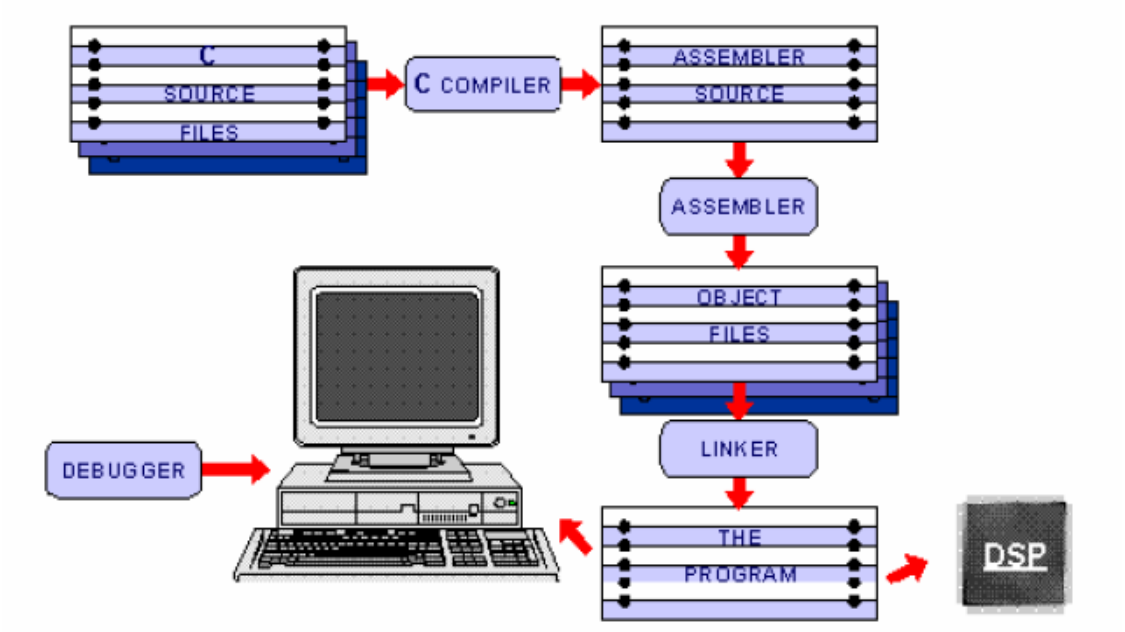
c. 1011 1110 0001 0110

d. DMOV *, AR1



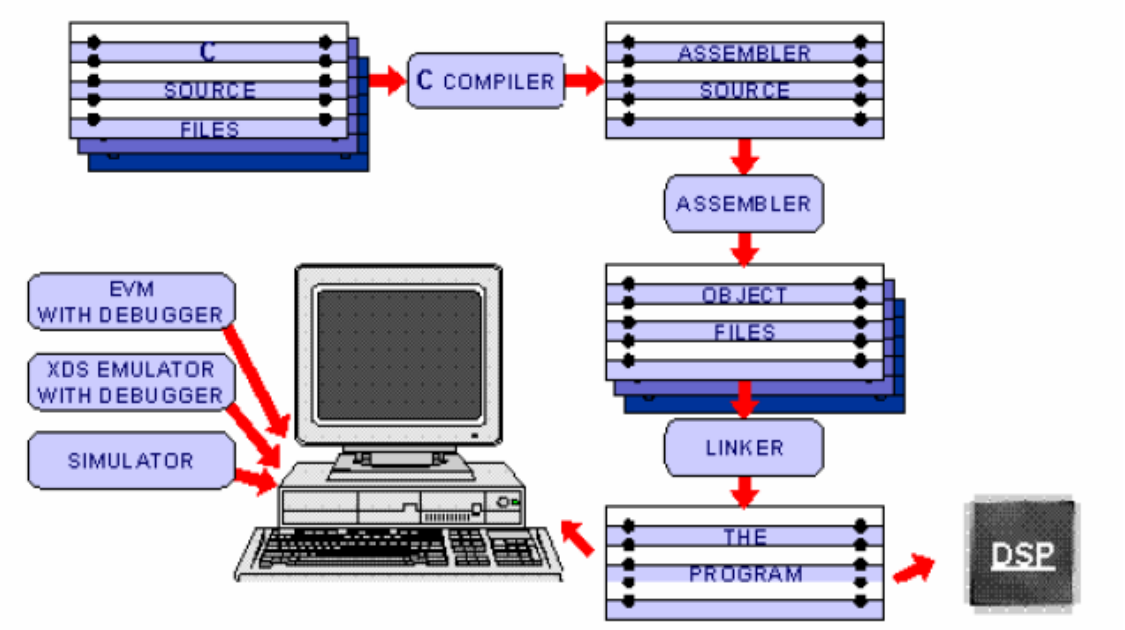
Ngôn ngữ C là ngôn ngữ bậc cao được sử dụng ngày càng nhiều để lập trình các DSP phức tạp hoặc thực thi các thuật toán có độ phức tạp cao. Lập trình bằng C đơn giản hoá thiết kế của các ứng dụng DSP vì người lập trình không còn bị giới hạn bởi tập chỉ thị nhỏ của các ngôn ngữ bậc thấp (như hợp ngữ).

Bộ biên dịch (compiler) C được sử dụng để dịch các mã nguồn C thành các mã hợp ngữ DSP thích hợp.



Phần cuối của lập trình bao gồm việc kiểm tra lỗi chương trình và làm thay đổi cho đến khi thực hiện tốt chức năng mong muốn. Quá trình cuối cùng trong chuỗi các quá trình phát triển một phần mềm thường được gọi là gỡ rối (debugging). Chương trình giúp cho việc gỡ rối phần mềm được gọi là bộ gỡ rối (debugger).

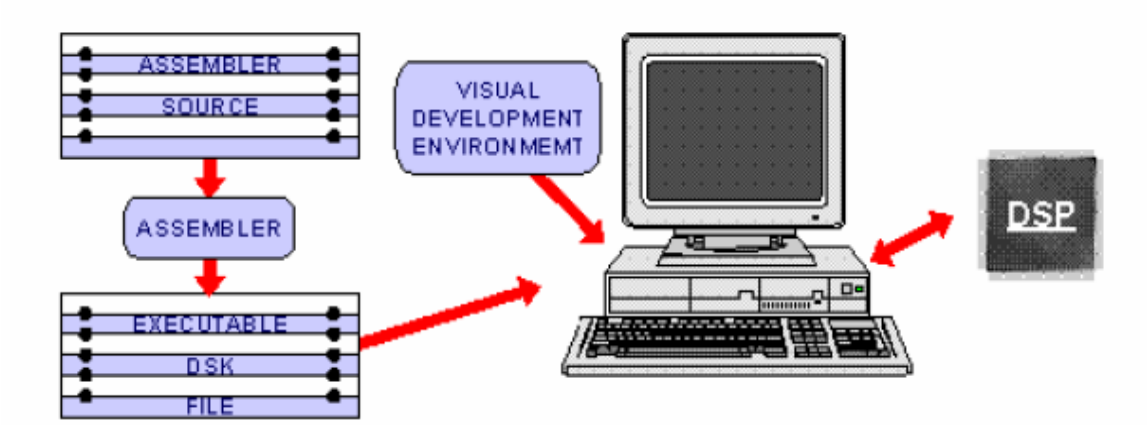
Một bộ gỡ rối cho phép người lập chương trình khả năng phân tích vấn đề kết hợp với các chương trình DSP của họ. Điều này được thực hiện trước khi gỡ rối được sử dụng với DSP mà ta làm thí nghiệm. C5x Visual Development Environment (C5x VDE) là bộ gỡ rối được sử dụng với DSP mà chúng ta làm thí nghiệm.



Những người phát triển hệ thống DSP hiếm khi gỡ rối một DSP mà không sử dụng một bộ gỡ rối hay debugger. Vì vậy, họ thường sử dụng EVMs, emulators và simulators để trợ giúp cho việc này.

Bộ DSP được sử dụng với bộ mạch là một bộ phận của module TM320C5x DSK (Digital Signal Processing Kit). Khi sử dụng EVMs, emulators và simulators, người phát triển có thể thay đổi trong quá trình phát triển mô hình của DSP dang được thí nghiệm

Một khi đã hoạt động được, thử nghiệm cuối cùng của chương trình này được cài đặt trên hệ thống DSP.



Các chương trình được bao gồm và sử dụng trong Digital Signal Processor được viết bằng hợp ngữ. Hợp ngữ được sử dụng như một đặc trưng của TM320C5x EVMs, nó đã cộng thêm các chỉ thị trong nó, và được gọi là các chỉ thị DSK.

3. Yêu cầu thiết bị

Để hoàn thành được các bài tập sau đây, ta cần:

- FACET base unit.
- Bọ mạch DIGITAL SIGNAL PROCESSOR.
- Chương trình C5x VDE.
- Các tệp chương trình (dsk) và hợp ngữ (asm) 1_1, Exl_2
- Máy hiện sóng
- Đồng hồ đo điện đa chức năng

BÀI 4. LÀM QUEN VỚI BỘ THÍ NGHIỆM LABVOLT - DSP

1. Mục đích

Kết thúc bài này, sinh viên được làm quen với vị trí và chức năng của mỗi linh kiện khác nhau trong hệ thống DSP

2. Thảo luận

Bo mạch có hai vùng chức năng: vùng chứa các phụ kiện của bo mạch và vùng chứa DSP và ngoại vi của nó.

Vùng chứa các phụ kiện của bo mạch bao gồm:

- DOWER SUPPLY với AUXILIARY POWER INPUT
- DC SOURCE.
- MICROPHONE PRE-AMPLIFIEF
- AUDIO AMPLIFIER

Chức năng:

- Khối mạch POWER SUPPLY cung cấp một nguồn DC đã được chỉnh lưu và lọc cho toàn bộ bọ mạch. Bo mạch có thể được vận hành theo hai cách khác nhau : hoặc điện áp vào của Power Supply có thể được nhận từ Lab-BoIl FACET base Unit hoặc có thể được nhận từ các kết nối ± 15V ngoài được tìm thấy trên khối AUXILIARY POWER INPUT.
- Khối DC SOURCE cung cấp một điện áp DC thay đổi và phụ thuộc vào vị trí của chiết áp, giữa -3,5V de và + 3,5Vdc. Khối DC SOURCE có thể được dùng nguồn của một tín hiệu tham chiến đầu vào cho chương trình chạy trên DSP.
- Khối MICROPIIONE PRE-AMPLIFIER được sử dụng để điều chỉnh một tín hiệu micro thành một mức thích hợp với đầu vào của DSP. Chiết áp GAIN thay đổi mức ra giữa một giá trị thấp và một giá trị cao.
- Để có thể nghe thấy tín hiệu từ ANALOG OUTPUT, được định vị trên khối CODEC, khối AUDIO AMPLIER được sử dụng.

Vùng chức năng thứ hai của bọ mạch là DSP và các ngoại vi của nó bao gồm:

- DSP
- CODEC
- I/O INTERFACE
- INTERRUPTS
- AUXILIARY I/O
- SERIAL PORT.

DSP được coi như là trái tim của hệ thống xử lý tín hiệu số.

- Khối DSP chứa một vi mạch DSP TM320C50 trong một chíp 132 chân dán trên bề mặt (surface mount). Nó có thể đạt tới tốc độ thực hiện 50MIPS. Có nhiều lại DSP chúng có thể thay đổi về các tốc độ chu trình. Tuy nhiện, tốc độ được giới hạn bởi các ràng buộc của hệ thống bên trong vi mạch. DSP có thể sử dụng một bộ tạo dao động bên trong để thiết lập đồng hồ hoặc cũng có thể sử dụng bộ tạo dao dông ngoài. DSP được dùng trên bọ mạch thí nghiệm được đặt cấu hình để sử dụng bộ tạo dao động ngoài.
- Khối OSCILATOR được đặt trên bọ mạch cung cấp cho nó một tín hiệu tham chiến 40 MHz. DSP chia tín hiệu này để tạo ra tín hiệu bên trong 20Mhz (tần số tín hiệu chủ) mà nó sử dụng để tính toán thời gian các chu trình chỉ thị của nó.
- Khối CODEC thường được cấu thành bởi các linh kiện sau:
  - một đầu vào GAIN lập trình được
  - một ANTI-ALISING FILTER (bộ lọc chống trùm phổ)
  - một bộ biến đổi tương tự - số
  - một bộ biến đổi số - tương tự
  - một POST-GILER (bộ lọc sau)
- Khối I/O INTERFACE là một phương tiện để hiển thị và nạp và thông tin chương trình. Chuyển mạch DIP8 có chức năng đưa 8 bit vào cấu hình DSP. Phụ thuộc vào chương trình đang được sử dụng, thông tin có thể được xử lý theo nhiều cách khác nhau. Các bộ hiển thị LED 7 thanh được sử dụng để đưa ra thông tin chương trình cho người sử dụng DSP. Như hầu hết các bộ vi xử lý, các DSP đều có khả năng điều khiển ngắt. Hai nút có thể được sử dụng như các thiết bị vào của người sử dụng cho một chương trình. Khi một trong các nút nhấn được nhấn thì một ngắt được sinh ra bên trong DSP và mã chương trình kết hợp với nó được thực hiện.
- Vùng AUXILARY I/0 đã được cộng thêm vào cho mục đích giám sát tín hiệu và để và để làm nguyên mẫu cho các bài tập DSP thêm vào được thực hiện trên bo mạch. Các đầu của khối AUXILARY I/O có thể được sử dụng để giao tiếp DSP với một mạch ngoài. Mạch ngoài này có thể được cấp nguồn bởi đầu 10 chân đặt trên khối AUXILARY I/O. Vùng AUXLIIARY I/O có ba cổng:
  - Các điểm kết nối ± 5Vdc và ± 5Vdc có sẵn để sử dụng trên đầu phải có 10 chân, chúng có thể được sử dụng để cấp nguồn cho một mạch ngoài. Các bộ cung cấp của bo mạch có cùng điểm đặt.
  - Đầu trái của 8 chân LSB (được đánh nhãn từ D0 đến D7) của bus dữ liệu của DSP ngoài, và bao gồm 4 đường địa chỉ được tiền mã hoá (được đánh nhãn từ PA0# đến PA3#).
  - Đầu giữa có các phần vào/ra (I/O) bao gồm:

- chọn dữ liệu (DS#), chương trình (PS#), khoảng vào/ra (IS#)
- đầu ra bộ định thời
- chọn đầu (RD#) và cho ghi (WE#) cho các thiết bị ngoài
- chọn đọc/ghi (R/W#) cho các truy nhập ngoài.
- tín hiệu báo cho biết đã nhận được ngắt (IACK#)
- đầu vào ngắt ngoài (INT4#)
- chọn hướng (DIR) và chọn chíp (CS#) để điều khiển việc truyền dữ liệu ngoài.

DSP trên bo mạch được lập trình để thành vai trò server đối với máy tính trong vai trò client. Để bộ DSP hoạt động, bo mạch SERIAL, PORT phải được nối với một trong các cổng nối tiếp của máy tính của bạn.

Chú ý: Nếu máy tính chủ không có một kết nối tiếp thứ hai thì vào thời điểm thích hợp trong tiến trình thực hiện bài tập sinh viên có thể tháo kết nối tiếp của Base Unit và dùng nó để nối bo mạch SERIAL PORT với máy tính

C5x VDE (C5x Visual Davelopment Environment) quản lý việc bắt tay giữa bo mạch và máy tính. Nó điều khiển tất cả các đầu vào và đầu ra từ bộ nhớ của DSP cổng nối tiếp. Một khi kết nối liên lạc giữa máy tính của bạn và bo DSp được thiết lập, C5x VDE có thể được sử dụng để nạp một chương trình vào DSP.

3. Tiến trình thí nghiệm

Giới thiệu bo mạch: Trong phần này, bạn sẽ làm quen với m ột số các linh kiện và khối mạch trên bọ mạch DIGIAL SIGNAL PROCESSOR.

1. Định vị trên bo mạch DIGITAL SIGNAL PROCESSOR tất cả các thiết bị đầu cuối chung. Dùng một điện trở kể để kiểm tra các thiết bị đầu cuối được nối với nhau hay chưa.

   Hỏi: Tất cả các thiết bị đầu cuối đã được nối với nhau?

   □ Có □ không

2. Bật nguồn cung cấp cho bọ mạch DIGITAL SIGNAL PROCESSOR.
3. Dùng một volt kế để kiểm tra điện áp một chiều bằng cách thay đổi chiết áp của DC SOURCE từ giá trị nhỏ nhất cho tới giá trị lớn nhất của nó. Đo điện áp DC tại đầu ra của DC source

Điện áp DC nhỏ nhất (VDC min) và điện áp DC lớn nhất (VDC max) đưa ra từ DC source?

VDC min = ……V

VDC max = ……V

4. Thực hiện các kết nối với DIGITAL SIGNAL PROCESSOR

Chú ý: Nếu chất lượng audio từ loa không tốt, có thể dùng tai nghe kèm theo bo mạch. Nối tai nghe vào đầu cắm tai nghe được đặt trên khối mạch AUDIO AMPLIFIER.

5. Nói vào micro, xem xét sự thay đổi của âm thanh phát ra trong khi cùng thực hiện thay đổi chiết áp của MICROPHONE PRE-AMPLIFIER và của AUDIO AMPLIFIER
6. Tháo toàn bộ các kết nối hiện có trên bọ mạch.

Làm quen với bọ mạch dùng một chương trình DSP: Trong mục này, C5x VDE sẽ được dùng để nạp và chạy một chương trình bên trong DSP

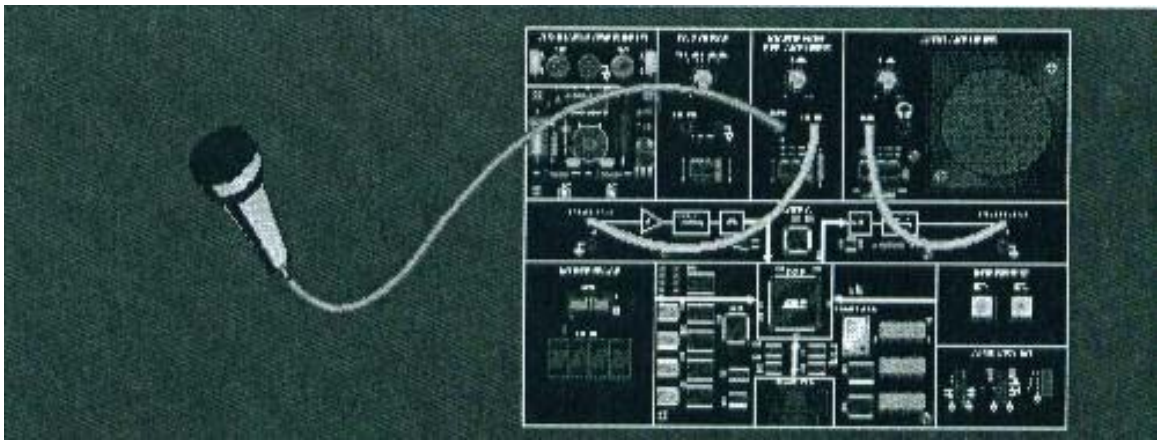
Chú ý: Trước khi sử dụng C5x VDE, hãy chắc chắn rằng nguồn của bọ mạch được bật và kết nối nối tiếp là hiện có giữa máy tính và khối mạch DIGITAL SIGNAL PROCESSOR được đánh nhãn SERIAL PORT.

7. Mở chương trình C5x VDE:
8. Dùng lệnh Load Program trong menu File để nạp chương trình ex1_1.dsk vào DSP.

Hỏi: Hai cửa sổ nào đang được mở trong C5x VDE?

a. C5x Registers và Peripheral Registers.
b. Dis-Assembly và Periphearal Registers.
c. C5x Registers và Dis-Assembly.
d. Peripheral Registers và File Selection

9. Kết nối bo mạch như hình vẽ . Điều này cho phép chương trình ex1_1.dsk vận hành đúng đắn



Chú ý: Dùng tai nghe nếu cần thiết.

10. Thực hiện lện RUN trên thanh công cụ của C5x VDE.
11. Quan sát những gì đọc ra được hiển thị bên trong khối mạch I/O INTERFACE.

Điều chỉnh chuyển mạch DIP (tất cả các bit đều ở vị trí 0) sao cho hiển thị đọc được là **0000**.

12. Nhấn nút thứ nhất INT# trên bo mạch INTERRUPTS để chuyển tới DSP các giá trị được nhập vào thông qua chuyển mạch DIP.
13. Dùng micro, cho một tín hiệu (giọng nói) vào DSP

    Chú ý: Điều chỉnh các chiết áp GAIN của MICROPHONE PRE-AMPLIFIER và của AUDIO AMPLIFIER để cải thiện âm thanh đầu ra.
14. Lưu ý rằng trong khi đang nói vào micro, các chấm trên màn hình của khối mạch I/O INTERFACE bật sáng.
15. Điều chỉnh chuyển mạch DIP sao cho màn hình I/O INTERFACE đọc được là **0015**.
16. Truyền giá trị của chuyển mạch DIP vào DSP bằng cách nhấn nút nhấn INT#.
17. Quan sát kết quả của sự thay đổi của xử lý tín hiệu trong âm thanh của giọng nói.
18. Lặp lại các bước từ 15 đến 17 cho mỗi một giá trị được hiển thị trên I/O INTERFACE sau đây: **0031**, **0063**, **0127**, **0255**

    Nhớ nhấn nút INT # sau khi đặt chuyển mạch DIP tới một giá trị mới.

    Hỏi: Sự lựa chọn nào sau đây là mô tả đúng đắn nhất về chương trình ex1_1.dsk được nạp vào DSP?

    a. Đây là một bộ ghi tiếng nói

    b. Đây là hệ điều hành Base Unit

    c. Đây là một máy phát chức năng

    d. Đây là một máy phát tiếng vọng.

    Hỏi: Con số được hiển thị trên I/O INTERFACE tỉ lệ với cái gì?

    a. Thời gian trễ (theo ms) giữa các tiếng vọng liên tiếp

    b. Số các tiếng vọng được tạo ra

    c. Thời gian cần dùng (theo ms) để sinh ra các tiếng vọng cho một âm thanh

    d. Số các mẫu phải lấy trên tín hiệu ra trong một giây
19. Thực hiện lệnh Halt trên thanh công cụ của C5x VDE. Đóng C5x VDE.

4. Kết luận

- DIGITAL SIGNAL PROCESSOR có hai vùng: vùng các phụ kiện của bo mạch và vùng DSP với các ngoại vi.
- Bo mạch được chia thành các khối mạch riêng rẽ.
- Trước khi một chương trình DSP có thể được nạp hoặc sử dụng, nguồn cung cấp của DIGITAL SIGNAL PROCESSOR phải được bật lên và kết nối nối tiếp giữa khối mạch SERIAL PORT và máy tính phải được thực hiện
- Các khối mạch CODEC, I/O INTERFACE, INTERRUPT và AUXILIARY I/O có thể chỉ được áp dụng bởi người sử dụng nếu chương trình nạp vào DSP đòi hỏi việc sử dụng chúng.

5. Câu hỏi ôn tập

1. Trước khi bo mạch DIGITAL SIGNAL PROCESSOR sẵn sàng để sử dụng có một số bước bắt buộc cần phải theo. Mệnh đề nào sau đây là bước cần thiết phải thực hiện trước khi sử dụng bo mạch ?
    a. Chắc chắn rằng các chuyển mạch của I/O INTERFACE đều ở vị trí 0
    b. Chắc chắn rằng kết nối nối tiếp là hiện có giữa máy tính chủ và khối mạch DIGITAL, SIGNAL PROCESSOR được đánh nhãn SERIAL PORT.
    c. Chắc chắn rằng nguồn cung cấp của bo mạch được bật
    d. Các mệnh đề b và c.
2. Khoảng điện áp DC mà chiết áp cho nguồn DC điều chỉnh được là bao nhiêu?
    a. –3,3V đến +3,6V
    b. –3,0V đến + 3,0V
    c. –3,5V đến + 3,5V
    d. Không có mệnh đề nào trong các mệnh đề trên là đúng.
3. Chân nào trong số các chân sau đây được đặt trên đầu giữa của bo mạch AUXILIARY I/O ?
    a. 4 đường địa chỉ tiền mã hoá (được đánh nhãn từ PA0# đến PA3#)
    b. TOUT, IACK #, INT4#, và RD#
    c. DS#, D0, D1, và D2
    d. CS#, INT4#, DS#, và PA1#
4. DSP TMS320C50 trên bo mạch DIGITAL SIGNAL PROCESSOR sử dụng đồng hồ hệ thống có tần số là bao nhiêu (nhắc lại rằng đây là đồng hồ đặt tốc độ tính toán cho DSP)?
    a. DSP dùng bộ tạo dao động bên trong 20MHZ
    b. DSP dùng bộ tạo dao động bên ngoài 40MHZ
    c. Thông qua kết nối nối tiếp, DSP dùng bộ tạo dao động bên trong 33.3MHz CODEC
    d. Thông qua kết nối bo mạch SERIAL PORT, DSP dùng bộ dao động trong của máy tính chủ.
5. Linh kiện nào trong các linh kiện sau đây thường được tìm thấy trong CODEC
    a. Một bộ lọc chống trùm phổ
    b. Một bộ biến đổi tương tự – số
    c. Một bộ biến đổi số – tương tự
    d. Tất cả các bộ nói trên.

TÀI LIỆU THAM KHẢO:

- [1] Ingle V., Proakis J., *Digital Signal Processing using MATLAB v.4,* 1997, PWS Publishing Company, ISBN 0-534-93805-1.
- [2] Nguyễn Quốc Trung, *Xử lý tín hiệu và lọc số, Tập 1 và Tập 2*, Nhà xuất bản KHKT 2004.
- [3] Proakis J., Manolakis D., *Digital Signal Processing, Principles, Algorithms, and Applications*, Third Edition, 1996, Prentice-Hall International Inc., ISBN 0-13-394336-9.
- [4] Rorabaurg Britton, *Digital Filter Designer's Handbook Featuring C Routines*, McGraw-Hill Education –Europe, 1993, ISBN 0-07-911166-1
- [5] Lab-Volt Ltd, *Digital Signal Processors, Student Workbook*, First Edition, 2000, ISBN 2-89289-473-5.

LINKS

- MATLAB website: www.mathworks.com
- LABVOLT website: www.labvolt.com
- Texas Instrument website: www.ti.com