TÌM TẦN SỐ CƠ BẢN CỦA TÍN HIỆU TIẾNG NÓI

Hồ Ngọc Huy, Đào Việt Hoàng, Phan Văn Khải, Lê Minh Châu, Đặng Huỳnh Khánh Ly

Nhóm 1, lớp HP: 16.89

Điểm (dành cho GV ghi)	**Bảng phân công nhiệm vụ** (SV ghi càng cụ thể thì GV càng dễ đặt câu hỏi và cho điểm mỗi SV)		**Chữ ký của SV** (mỗi SV ký xác nhận trước khi nộp báo cáo)
	Hồ Ngọc Huy	Đọc tài liệu, cài đặt và trình bày thuật toán cửa sổ, tính F0 thủ công, tổng hợp slide	
	Đào Việt Hoàng	Đọc tài liệu, viết báo cáo thuật toán cửa sổ Hamming, trình bày số điểm fft	
	Phan Văn Khải	Đọc tài liệu, viết báo cáo, cài đặt và trình bày thuật toán tự động tính F0 trên miền thời gian dùng hàm tự tương quan. Phân công nhiệm vụ các thành viên	
	Lê Minh Châu	Đọc tài liệu, viết báo cáo, cài đặt và trình bày thuật toán tự động tính F0 trên miền tần số	
	Đặng Huỳnh Khánh Ly	Đọc tài liệu, viết báo cáo, cài đặt và trình bày thuật toán lọc trung vị, tổng hợp báo cáo	

Lời cam đoan: Chúng tôi, gồm các sinh viên có chữ ký ở trên, cam đoan rằng báo cáo này là do chúng tôi tự viết dựa trên các tài liệu tham khảo ghi rõ trong phần VII. Các số liệu thực nghiệm và mã nguồn chương trình nếu không chỉ dẫn nguồn tham khảo đều do chúng tôi tự làm. Nếu vi phạm thì chúng tôi xin chịu trách nhiệm và tuân theo xử lý của giáo viên hướng dẫn.

TÓM TẮT— Xác định tần số cơ bản F0 là một trong những bài toán quan trọng nhất trong xử lý tín hiệu số, đặc biệt là trong xử lý tín hiệu tiếng nói. Báo cáo này sẽ trình bày thuật toán xác định F0 trên miền thời gian và trên miền tần số, với sự tham gia của thuật toán cửa sổ Hamming và thuật toán lọc trung vị để cải thiện tính chính xác của thuật toán.

Từ khóa— Tần số cơ bản, hàm tự tương quan, biến đổi Fourier, cửa sổ Hamming, lọc trung vị

Mục lục

I. ĐẶT VẤN ĐỀ

Xử lý tiếng nói là một trong những chủ đề đang được nghiên cứu không những ở trong nước mà còn ở tầm quốc tế. Ứng dụng của việc xử lý tiếng nói rất rộng rãi: nhận dạng giọng, tái tạo hoặc tạo mới giọng nói của con người. Để làm được điều đó, một phân đoạn quan trọng là xác định tần số cơ bản của tín hiệu bởi tần số cơ bản có ý nghĩa đặc trưng cho tín hiệu số nói chung và tín hiệu giọng nói nói riêng.

Giọng nói của mỗi người sẽ có phần nhiễu và phần có tính chu kỳ. Chu kỳ của tín hiệu là thời gian mà tín hiệu đó lặp lại trên miền thời gian. Tần số cơ bản của một tín hiệu tuần hoàn bằng nghịch đảo của tín hiệu đó. Bài báo cáo này sẽ trình bày hai phương pháp tìm tần số cơ bản của một tín hiệu âm thanh: sử dụng hàm tự tương quan và sử dụng thuật toán FFT kết hợp với thuật toán hàm cửa sổ Hamming và thuật toán lọc trung vị để tối ưu kết quả đạt được

Bài báo cáo có bố cục như sau: Phần II trình bày Lý thuyết xử lý tín hiệu tiếng nói, các thuật toán tính tần số cơ bản trên miền tần số, trên miền thời gian, hàm cửa sổ và lọc trung vị. Phần III trình bày mã nguồn cài đặt thuật toán. Phần IV trình bày kết quả thực nghiệm. Phần V trình bày kết luận.

II. LÝ THUYẾT XỬ LÝ TÍN HIỆU TIẾNG NÓI VÀ CÁC THUẬT TOÁN TÍNH F0

Phần này trình bày các lý thuyết có liên quan đến vấn đề xử lý tín hiệu tiếng nói.

A. Hàm cửa sổ

1. Cơ sở lý thuyết:

Như ta đã biết , phép biến đổi Fourier rời rạc DFT tác động trên tín hiệu có độ dài hữu hạn, nên cần thiết phải hạn chế độ dài đối với các tín hiệu có độ dài vô hạn hoặc quá lớn để có thể nghiên cứu phổ của chúng. Vì vậy khi đi phân tích tín hiệu tiếng nói, ta phải xử lí tín hiệu thành từng đoạn hay từng khung (frame) có thời gian tương dối nhỏ (30ms).

Để làm điều này ta thường dùng hàm cửa sổ, tức là nhân tín hiệu X(n) với cửa sổ $W(n-n_0)$ để nhận được một đoạn $X_N(n)$ trong đoạn n_0 tới n_0+N-1 để phân tích.

$$x_N(n) = x(n).w(n-n_0) = \begin{cases} x(n) & n0 \le n \le n0+N-1 \\ 0 & với\ n\ còn\ lại \end{cases}$$

Việc lấy cửa sổ tín hiệu nhằm phân loại tín hiệu, cải thiện các đặc điểm nổi bật của tín hiệu trên miền tần số để phân tích.

Hàm cửa sổ có nhiều loại, tùy thuộc vào mục đích khi phân tích tín hiệu và sử dụng. Trong đó có 2 hàm cửa sổ được sử dụng nhiều nhất là hàm cửa sổ chữ nhật (Rectangular window) và hàm cửa sổ Hamming (Hamming window). Ngoài ra còn có hàm cửa sổ Hanning, Blackman, tam giác (Triangle window)….

a) Hàm cửa sổ Chữ nhật (Retangular Window)

Đây là dạng cửa sổ đơn giản nhất, ta chỉ việc nhân tín hiệu với tín hiệu chữ nhật để có được tín hiệu của x(n). Hàm cửa sổ chữ nhật thường được dùng trong các tín hiệu quang phổ hẹp vì không hàm mất các tín hiệu quang trọng.

Hàm cửa sổ chữ nhật: $w(n) = \begin{cases} 1 ; & 0 \le n \le N-1 \\ 0 ; & với\ n\ còn\ lại \end{cases}$

Trong matlab có cung cấp sẵn hàm tạo cửa sổ chữ nhật là: W = Rectwin(N), với N là chiều dài cửa sổ.

Hàm trả về một cửa sổ hình chữ nhật có chiều dài N trong vector cột w.

b) Hàm cửa sổ Hamming (Hamming Window)

Là hàm cửa sổ đặc biệt được sử dụng trong việc phân tích phổ để giảm hiện tượng rò phổ. Hàm cửa sổ Hamming thường được dùng nhiều hơn hàm cửa sổ chữ nhật trong các tín hiệu có phạm vi quang phổ rộng.

Dạng tổng quát của cửa sổ Hamming là :

$$W(n) = \begin{cases} 0{,}54 + 0{,}46\cos\left(\frac{2\pi n}{N-1}\right) & ; 0 \le n \le N-1 \\ 0 & ; với\ n\ còn\ lại \end{cases}$$

Trong matlab có cung cấp sẵn hàm tạo cửa sổ Hamming là: H = Hamming(N,'periodic'), với N là chiều dài cửa sổ, periodic là kiểu lấy mẫu theo chu kỳ.

Hàm trả về một cửa sổ dưới dạng vecto cột H.

B. Hàm tự tương quan

1. Cơ sở lý thuyết:

a. Khái niệm:

Việc phân tích tiếng nói trong miền thời gian tức là phân tích trực tiếp trên dạng sóng tín hiệu sau khi thực hiện việc lấy cửa sổ trong miền thời gian.

Bài báo cáo này sẽ trình bày thuật toán phân tích tiếng nói trong miền thời gian bằng cách sử dụng hàm tự tương quan.

Hàm tự tương quan là một trong các cách thức so sánh mức độ giống nhau giữa tín hiệu cần phân tích và tín hiệu đã được dịch đi một khoảng thời gian của chính nó. Do đó, hàm tự tương quan thường được sử dụng như một công cụ để xác định tính chu kỳ của tín hiệu và nó cũng là cơ sở cho nhiều phương pháp phân tích phổ khác.

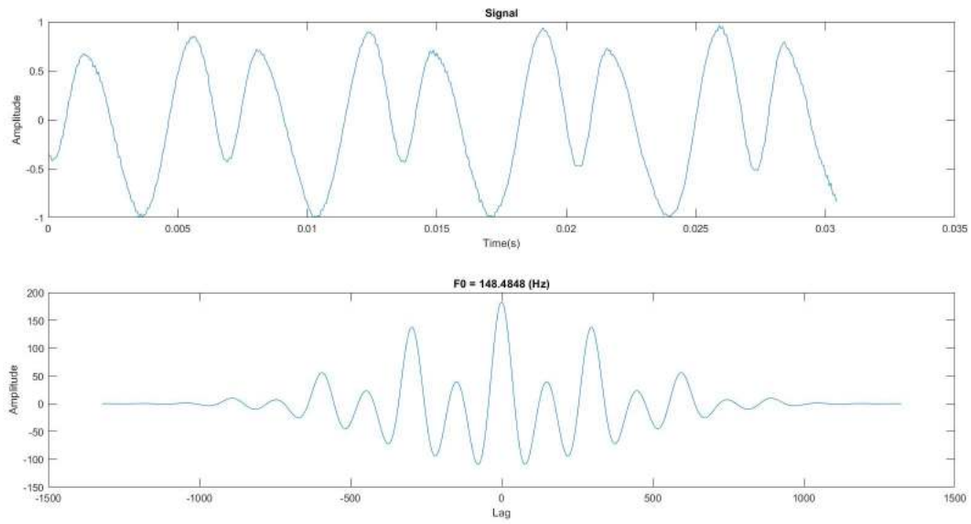
b. Công thức:

$$R(k) = \sum_{n=1}^{N-1-k} x(n)x(n+k) \;\; k = 0,1,\dots,K$$

Hàm tự tương quan có những tính chất quan trọng sau:

- Là hàm chẵn R(k) = R(-k)
- R(k) đạt giá trị cực đại tại 0: R(k) ≤ R(0) với mọi k
- Giá trị R(0) chính bằng năng lượng của tín hiệu

$$R(0) = \sum_{m=-\infty}^{\infty} x^2(m)$$



Hình 1. Độ trễ khi áp dụng hàm XCORR trong Matlab

Dựa vào các tính chất trên ta có nhận xét: Hàm tự tương quan R(k) sẽ đạt giá trị cực đại tương ứng tại các điểm 0, ±T, ±2T,… là bội của chu kỳ cơ bản của tín hiệu và bằng giá trị năng lượng của tín hiệu, các điểm cực đại được gọi là các đỉnh (peak). Như vậy bài toán xác định chu kỳ cơ bản của tín hiệu tiếng nói sẽ đưa về việc xác định chu kỳ của hàm tự tương quan. Khi đã có chu kỳ cơ bản, ta chỉ cần nghịch đảo giá trị đó để ra tần số cơ bản của tín hiệu tiếng nói cần phân tích.

2. Cài đặt thuật toán:

a. Ý tưởng thuật toán:

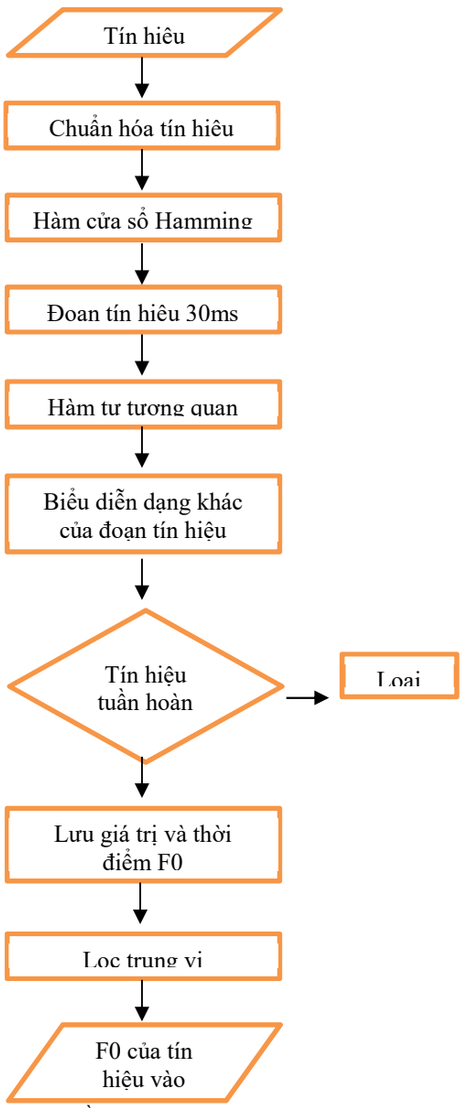
Từ tín hiệu đầu vào đã được chuẩn hóa và qua hàm cửa sổ Hamming, ta sẽ xây dựng thuật toán tìm tần số cơ bản F0:

- B1: Chia nhỏ tín hiệu đầu vào thành những khung cửa sổ 30ms
- B2: Sử dụng hàm tự tương quan để xác định chu kỳ cơ bản của khung 30ms:
  - Dùng hàm xcorr trong thư viện Matlab để biểu diễn dạng khác của tín hiệu
  - Xác định các đỉnh đạt giá trị cực đại
  - Hiệu giữa 2 đỉnh đạt giá trị cực đại liên tiếp nhau là chu kỳ của tín hiệu 30ms
  - Nghịch đảo giá trị chu kỳ, ta được giá trị tần số của tín hiệu 30ms
- B3: Xác định tính tuần hoàn của tín hiệu 30ms
  - Vì hàm tự tương quan R(k) sẽ đạt giá trị cực đại tương ứng tại các điểm 0, ±T, ±2T,… là bội của chu kỳ cơ bản của tín hiệu nên ta sẽ xác định từ 3 đỉnh đạt giá trị liên tiếp nhau trở lên để xác định tính tuần hoàn
  - So sánh các hiệu giữa 2 đỉnh liên tiếp trong những đỉnh đang xét
    Nếu các hiệu bằng nhau hoặc chênh lệch với sai số nhỏ thì kết luận tín hiệu là tuần hoàn
    Nếu các hiệu không bằng nhau thì kết luận tín hiệu là không tuần hoàn.
- B4: Lưu các giá trị và thời điểm tính F0 của các khung 30ms
- B5: Sử dụng lọc trung vị để lọc các giá trị F0 đã tính được
- B6: Hiển thị kết quả F0 của tín hiệu đầu vào.

b. Thư viện Matlab:

- Thuật toán sử dụng hàm xcorr và hàm find của thư viện Matlab

c. Lưu đồ thuật toán:



Hình 2. Sơ đồ thuật toán hàm Tự tương quan

3. Hạn chế và hướng khắc phục:

- Từ Hình 1, có thể nhìn thấy năng lượng giảm dần theo thời gian, nên việc so sánh các chu kỳ liên tiếp nhau trong khung tín hiệu 30ms gặp khó khăn
  - ⇨ Để việc tính toán tần số cơ bản bớt phức tạp, thuật toán được sử dụng sẽ chỉ xét ít nhất 3 điểm có giá trị cực đại về biên độ và sẽ đặt một biến sai số (error) để thuận tiện cho việc kiểm tra tính tuần hoàn.
- Tần số cơ bản F0 tính được có nhiều giá trị bất thường
  - ⇨ Sử dụng hàm lọc trung vị để làm trơn tín hiệu

C. Phép biến đổi Fourier

1. Cơ sở lí thuyết:

a. Biến đổi Fourier rời rạc:

Phép biến đổi Fourier rời rạc (DFT), là phiên bản của biến đổi Fourier cho các tín hiệu thời gian rời rạc. Đầu vào của biến đổi này là một chuỗi hữu hạn các số thực hoặc số phức, làm biến đổi này là một công cụ lý tưởng để xử lý thông tin trên các máy tính. Đặc biệt, biến đổi này được sử dụng rộng rãi trong xử lý tín hiệu và các ngành liên quan đến phân tích

tần số chứa trong một tín hiệu, để giải phương trình đạo hàm riêng, và để làm các phép như tích chập. Biến đổi này có thể được tính nhanh bởi thuật toán biến đổi Fourier nhanh (FFT).
Công thức DFT-N điểm:

$$X[k] = X(e^{j\omega}) = \sum_{n=0}^{N-1} x(n)e^{-\frac{jk2\pi n}{N}} \quad (\text{với } \omega = \frac{k2\pi}{N} \text{ và } 0 \le k \le N-1) \ (*)$$

b. Biến đổi Fourier nhanh:

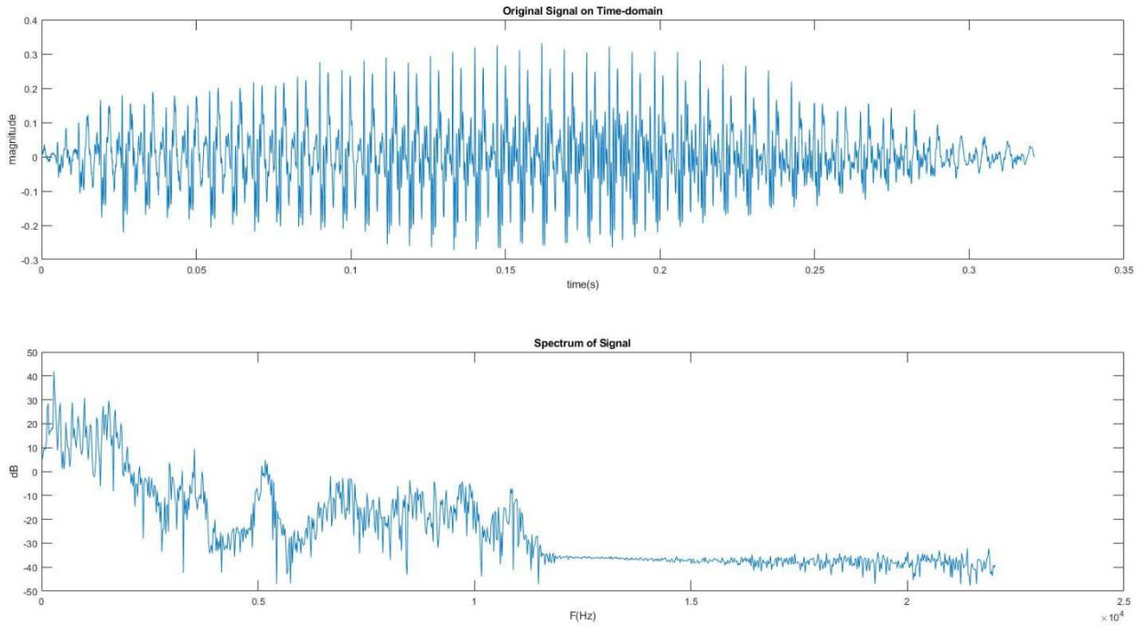
Thuật toán DFT giúp việc tính toán DFT nhanh và gọn hơn.Để đánh giá hiệu quả của thuật toán, ta sử dụng số phép tính nhân và cộng phức. Số phép nhân và cộng phức liên quan trực tiếp đến tốc độ tính toán khi thuật toán được thực hiện trên các máy tính hay là các bộ xử lý chuyên dụng.

Thuật giải FFT chỉ áp dụng cho trường hợp $N = 2^s$, $s \in N$. Vì N chẵn, nên tổng (*) có thể phân tích thành hai tổng:

$$X[k] = \sum_{n\ lẻ} x(n)e^{-\frac{jk2\pi k}{N}} + \sum_{n\ chẵn} x(n)e^{-\frac{jk2\pi k}{N}}$$
$$= \sum_{m=0}^{\frac{N}{2}-1} x(2m)e^{-\frac{jk4\pi k}{N}} + \sum_{m=0}^{\frac{N}{2}-1} x(2m+1)e^{-\frac{jk4\pi k}{N}}e^{-\frac{jk2\pi k}{N}} = X_e[k] + e^{-\frac{jk2\pi k}{N}}X_0[k]$$

Xe[k] và Xo[k] lần lượt là biến đổi Fourier của hai dãy {x[2m] | m = 0, 1, 2, ..., N/2 -1 } và {x[2m+1] | m = 0, 1,2, ..., N/2 -1 }. Có nghĩa là mỗi một Xe[k] và Xo[k] được phân tích thành tổng của hai phép biến đổi Fourier rời rạc của N/2 điểm. Tiếp tục quá trình trên cho đến khi ta được phép biến đổi Fourier của 2 điểm. Ngoài ra, do tính tuần hoàn của chu kỳ N/2 nên chỉ cần tính Xe[k] và Xo[k] với $N/2 \le k \le N-1$ và $0 \le k \le N/2 - 1$.

Bằng thuật toán FFT, cần $\frac{N}{2}log_2 N$ phép nhân phức thay cho $(N-1)^2$ phép nhân phức và $Nlog_2 N$ phép cộng phức thay vì $N(N-1)$. Do đó, tính trực tiếp từ định nghĩa DFT (*) đòi hỏi $O(N^2)$ phép tính, FFT sẽ giúp tính cùng kết quả đó trong O(N*logN) phép tính.



Hình 3. Độ trễ khi áp dụng hàm XCORR trong Matlab

2. Cài đặt thuật toán:

a. Ý tưởng thuật toán:

Tín hiệu đầu vào, sau khi phân tích phổ sử dụng hàm FFT, có ra phổ có dạng vạch, khoảng cách giữa cách vạch là bội số của f: f, 2f, 3f,… Để tìm tần số cơ bản, ta thực hiện các bước sau:

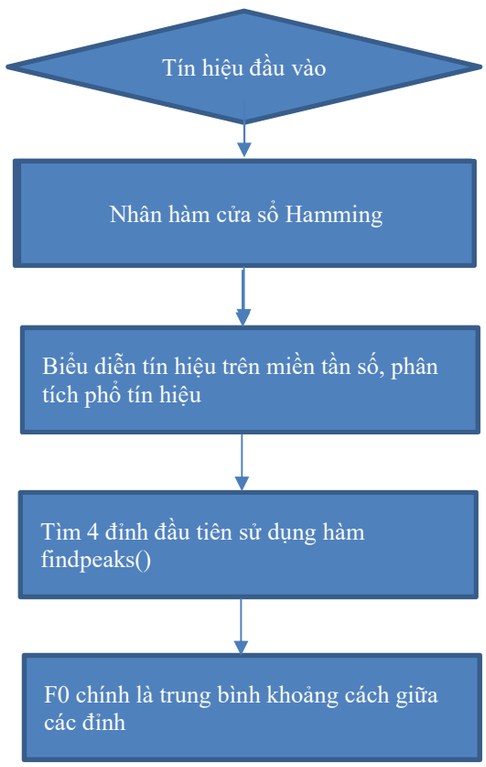
- B1: Chuẩn hóa tín hiệu, nhân tín hiệu với hàm cửa sổ Hamming
- B2: Chia tín hiệu thành từng đoạn 30ms

- B3: Sử dụng hàm FFT, phân tích phổ tính hiệu
- B4: Tìm các đỉnh của phổ bằng hàm findpeaks()
  - Tìm 4 đỉnh đầu tiên
  - Sử dụng các điều kiện để lọc các đỉnh không mong muốn (nhiễu)
  - Nếu không thỏa điều kiện/tìm không đủ số đỉnh => tín hiệu không tuần hoàn
  - Trung bình khoảng cách các đỉnh là tần số cơ sở của đoạn tín hiệu 30ms
- B5: Lưu các giá trị F0 tại các đoạn 30ms thành một mảng
- B6: Lọc trung vị mảng giá trị F0
- B7: Tính trung bình các giá trị F0, cho ra tần số cơ bản của tính hiệu đầu vào

b. Thư viện Matlab:

Thuật toán sử dụng hàm fft() và findpeaks() của Matlab.

c. Sơ đồ khối thuật toán



Hình 4. Xác định F0 trên miền tần số

c. Hạn chế và hướng khắc phục:

- Hiện tượng rò phổ
  => hàm cửa sổ Hamming.
- Xác định được các khung lựa chọn là khung của tín hiệu hay nhiễu
  => Dựa trên biên độ của tín hiệu

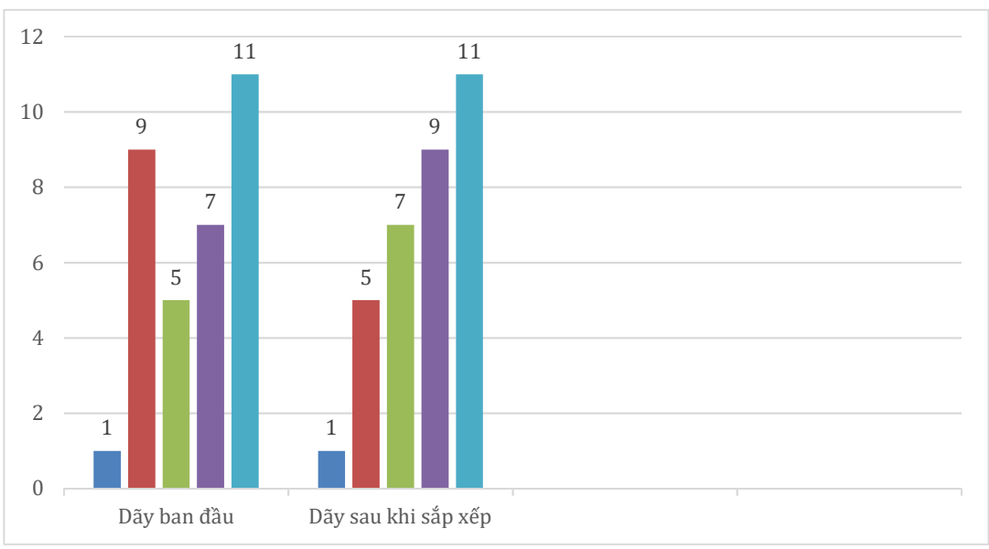
D. Lọc trung vị

1.Cơ sở lí thuyết:

1. Số trung vị là gì ?

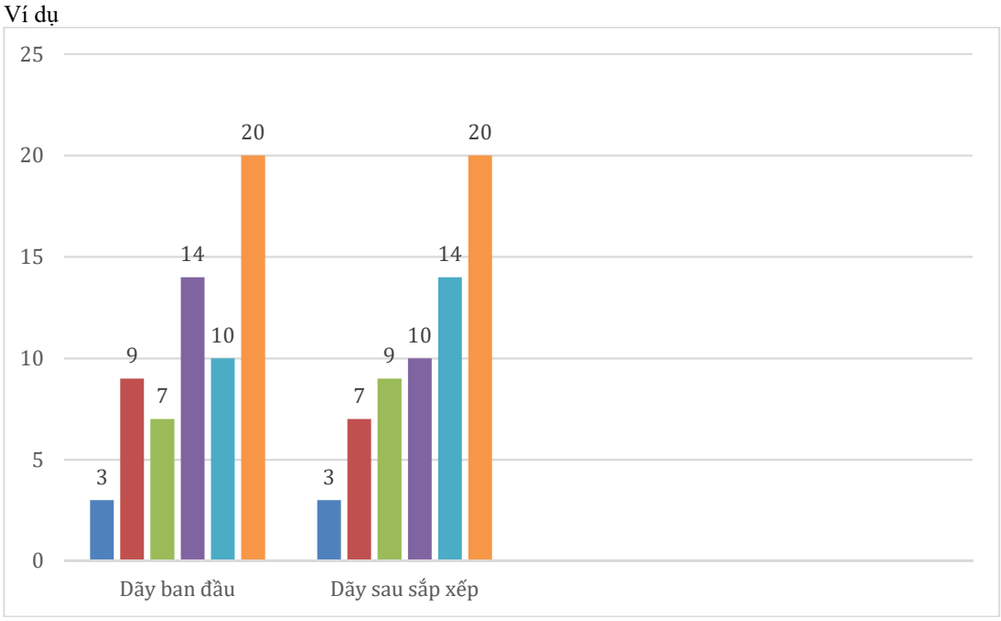
Số trung vị chính là số tại vị trí chính giữa của dãy sau khi sắp xếp theo thứ tự tăng dần.
Giả sử dãy gồm N phần tử, ta xét 2 trường hợp sau khi dãy được sắp xếp :
+ Nếu N lẻ :
Số trung vị là số tại vị trí (N+1) / 2.
Ví dụ:

⇨ 7 là số trung vị.



Hình 5. Lọc trung vị với N lẻ

+ Nếu N chẵn:
Số trung vị là trung bình cộng của 2 số tại vị trí [N / 2 và (N / 2 +1)].
Ví dụ



Hình 6. Lọc trung vị với N chẵn

➔ Số trung vị bằng $\frac{9+10}{2} = 9$.

2. Thuật toán lọc trung vị :

a.Vì sao chọn lọc trung vị ?

Mục đích : làm mịn tín hiệu sau khi tự tương quan trên miền tần số (loại bỏ những điểm bất thường).

***b*. Hàm có sẵn trong MATLAB?**

Trong MATLAB, ta sử dụng hàm medfilt1(x,n) với x là tín hiệu vào và n là độ dài khung lọc.

***c*. Mô tả thuật toán tự code:**

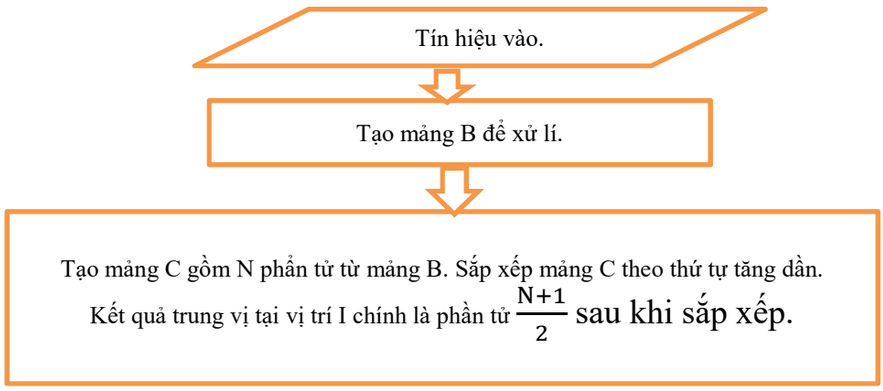
Trong bài thực hành này, ta chỉ xét với trường hợp N lẻ, tức là độ dài khung lọc nhập vào là một số lẻ.

Bước 1:
Gọi tín hiệu vào là mảng A = x, độ dài của mảng là M = length(x),
N là độ dài khung lọc.
Tính số phần tử ta cần thêm vào đầu mảng A và cuối mảng A. Số phần tử cần thêm vào được tính theo công thức : $tmp = \frac{N-1}{2}$.

Bước 2:
Tạo được mảng mới B có số phần tử là M + 2*tmp.
Mảng B gồm tmp phần tử đầu và tmp phần tử cuối có giá trị bằng 0.
Đoạn từ tmp+1 đến tmp+M có giá trị là các phần tử của mảng A, các giá trị này được tính theo công thức : B[i] = A [i - tmp].

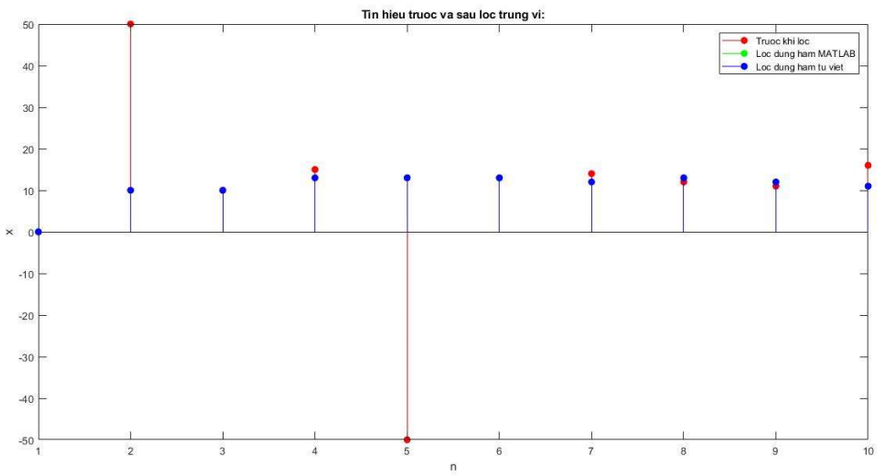
Bước 3:
+ Cho một vòng lặp i chạy từ phần tử thứ nhất đến phần tử M của mảng A, với mỗi i lần lượt tạo ra một mảng C gồm N phần tử lấy từ mảng B vừa tạo ở trên.
+ Sắp xếp mảng C theo thứ tự tăng dần. Kết quả cần tìm chính là phần tử tại vị trí $\frac{N+1}{2}$.

Sơ đồ thuật toán:



Hình 7. Lọc trung vị với N lẻ

a. Đánh giá thuật toán?
Thuật toán hoạt động ổn định, thử với N truyền vào nhỏ hơn 50, so sánh không khác nhiều với việc dùng hàm có sẵn của MATLAB. Sử dụng hàm sort() có sẵn của MATLAB



Hình 8. Đánh giá thuật toán

3.Áp dụng vào việc xử lí bài toán:

a. Vấn đề đặt ra:

Tín hiệu tiếng nói sau khi tính F0 (được rời rạc hóa) bằng hàm tự tương quan hoặc trên miền tần số vẫn còn tồn tại những điểm bất thường do tạp âm gây ra.

b. Giải quyết vấn đề:

Kết quả : Loại bỏ được những điểm có biên độ bất thường trong dải tần số F0.

c. Yếu tố ảnh hưởng đến hiệu quả lọc:

Cần phải quan tâm đến việc chọn độ dài khung lọc phù hợp.

➔ Sai số toàn phương RMSE của tín hiệu là một trong các tiêu chí quan trọng để xác định N nên chọn để lọc. N nào có sai số RMSE càng thấp thì nên chọn bậc đó.

d. Khảo sát kết quả thực nghiệm thuật toán lọc trung vị:

Xét các tần số của từng nguyên âm cơ bản /a/, /e/, /i/, /o/ và /u/ trước khi lọc (sau khi được tính bằng hàm tự tương quan) và sau khi lọc trung vị với N khác nhau.

Bảng 1. Bảng số liệu so sánh F0 (được tính trên miền thời gian) của các nguyên âm trước và sau khi lọc trung vị.

Tín hiệu	Trước khi lọc	Lọc N = 3	Lọc N = 5	Lọc N = 7
/a/	F0 mean = 137.6232(Hz) F0 RMSE = 2.987(Hz)	F0 mean = 137.6572(Hz) F0 RMSE = 2.9556 (Hz)	F0 mean = 137.6572(Hz) F0 RMSE = 2.9556(Hz)	F0 mean = 137.8635(Hz) F0 RMSE = 2.8036(Hz)
/e/	F0 mean = 135.1001(Hz) F0 RMSE = 4.8102(Hz)	F0 mean = 135.1632(Hz) F0 RMSE = 4.6786(Hz)	F0 mean = 135.2737(Hz) F0 RMSE = 4.1718(Hz)	F0 mean = 135.6897(Hz) F0 RMSE = 3.8224(Hz)
/i/	F0 mean = 130.2465 (Hz) F0 RMSE = 6.3254 (Hz)	F0 mean = 130.1367 (Hz) F0 RMSE = 6.1397 (Hz)	F0 mean = 129.3647 (Hz) F0 RMSE = 5.5697 (Hz)	F0 mean = 129.5678 (Hz) F0 RMSE = 5.9842 (Hz)
/o/	F0 mean = 132.1709(Hz) F0 RMSE = 5.6782(Hz)	F0 mean = 132.4265(Hz) F0 RMSE = 5.362(Hz)	F0 mean = 132.3858(Hz) F0 RMSE = 5.3023(Hz)	F0 mean = 132.5422(Hz) F0 RMSE = 5.0504(Hz)
/u/	F0 mean = 132.8496(Hz) F0 RMSE = 7.7934(Hz)	F0 mean = 132.972(Hz) F0 RMSE = 7.5815(Hz)	F0 mean = 133.674(Hz) F0 RMSE = 6.5154(Hz)	F0 mean = 134.1749(Hz) F0 RMSE = 5.8805(Hz)

Xét các kết quả tần số của từng nguyên âm cơ bản /a/, /e/, /i/, /o/ và /u/ trước khi lọc (sau khi được tính trên miền tần số bằng phép biến đổi FFT) và sau khi lọc trung vị với N khác nhau.

Bảng 2. Bảng số liệu so sánh F0 (được tính trên miền tần số) của các nguyên âm trước và sau khi lọc trung vị.

Tín hiệu	Trước khi lọc	Lọc N = 3	Lọc N = 5	Lọc N = 7
/a/	F0 mean = 163.2935(Hz) F0 RMSE = 23.1408(Hz)	F0 mean = 163.6523(Hz) F0 RMSE = 23.1281 (Hz)	F0 mean = 163.6523(Hz) F0 RMSE = 23.1281(Hz)	F0 mean = 162.2168(Hz) F0 RMSE = 22.2566(Hz)
/e/	F0 mean = 151.2451(Hz) F0 RMSE = 16.2272(Hz)	F0 mean = 151.2451(Hz) F0 RMSE = 16.2272(Hz)	F0 mean = 151.2451(Hz) F0 RMSE = 16.2272(Hz)	F0 mean = 152.7832(Hz) F0 RMSE = 14.2571(Hz)
/i/	F0 mean = 171.0693 (Hz) F0 RMSE = 23.0901 (Hz)	F0 mean = 170.6092 (Hz) F0 RMSE = 22.0092 (Hz)	F0 mean = 171.7135 (Hz) F0 RMSE = 21.3808 (Hz)	F0 mean = 175.5784 (Hz) F0 RMSE = 24.1587 (Hz)
/o/	F0 mean = 157.9102(Hz) F0 RMSE = 28.469(Hz)	F0 mean = 157.8169(Hz) F0 RMSE = 28.364(Hz)	F0 mean = 157.7632(Hz) F0 RMSE =28.132 (Hz)	F0 mean = 157.8942(Hz) F0 RMSE = 28..2504(Hz)
/u/	F0 mean = 156.3721(Hz) F0 RMSE = 14.5031(Hz)	F0 mean = 156.8848(Hz) F0 RMSE = 12.3903(Hz)	F0 mean = 157.3975(Hz) F0 RMSE = 10.8987(Hz)	F0 mean = 157.9102(Hz) F0 RMSE = 10.6528(Hz)

Nhận xét:

+ Với N nhỏ (N = 3) : Tín hiệu không khác nhiều so với tín hiệu trước khi lọc, vẫn còn tồn tại nhiều điểm bất thường.

+ Với N lớn : N càng lớn thì tín hiệu càng mịn tuy nhiên tín hiệu bị làm trơn quá, một vài thông tin quan trọng đã bị loại bỏ, khác xa với tin hiệu ban đầu.

Lọc trung vị làm việc tốt tại N = 5 và N =7. Với N = 5 thì các tín hiệu vẫn còn một vài giá trị bất thường nhưng tại vài vị trí khác nhau của tín hiệu cũng giữ được độ chính xác. Khi N = 7 thì hầu như các giá trị bất thường không còn, nhưng giá trị tại một vài vị trí đã bị thay thế thông tin.

Kết luận:

a) Đối với tần số cơ bản được tính trên miền thời gian

Nguyên âm /a/ : nên chọn độ dài khung lọc N = 7
Nguyên âm /e/ : nên chọn độ dài khung lọc N = 7
Nguyên âm /i/ : nên chọn độ dài khung lọc N = 5
Nguyên âm /o/ : nên chọn độ dài khung lọc N = 7
Nguyên âm /u/ : nên chọn độ dài khung lọc N = 7
Vậy , nên chọn N = 7 để có được tin hiệu sau lọc đẹp nhất.

b) Đối với tần số cơ bản được tính trên miền tần số :

Nguyên âm /a/ : nên chọn độ dài khung lọc N = 7
Nguyên âm /e/ : nên chọn độ dài khung lọc N = 7
Nguyên âm /i/ : nên chọn độ dài khung lọc N = 5
Nguyên âm /o/ : nên chọn độ dài khung lọc N = 5
Nguyên âm /u/ : nên chọn độ dài khung lọc N = 7
Vậy , nên chọn N = 7 để có được tin hiệu sau lọc đẹp nhất.

III. CÀI ĐẶT CÁC THUẬT TOÁN

A. Chuẩn bị tín hiệu âm thanh

Mỗi thành viên thu âm tín hiệu của 5 nguyên âm (/a/, /e/, /i/, /o/, /u/) trong môi trường yên tĩnh bằng chính giọng nói của mình, dùng tính năng "record" của 1 phần mềm xử lý âm thanh với định dạng .wav. Tần số lấy mẫu Fs được thiết lập ở 44,1 kHz, độ phân giải 16 bit/mẫu, số kênh là 1 (mono).

B. Phân tích tín hiệu thủ công

Mỗi thành viên đo chu kỳ cơ bản T_0 theo đơn vị 1/10 milisecond của đoạn tín hiệu của nguyên âm bằng cách đo thủ công 1 chu kỳ nào đó (chọn ở trung tâm của nguyên âm) của sóng tín hiệu (waveform), dùng Audacity. Tính nghịch đảo của T_0 để thu được ước lượng của tần số cơ bản F_0, sau đó tính trung bình của 5 giá trị F_0 thu được.

Trình bày mã nguồn cài đặt các thuật toán (copy & paste mã nguồn từ Editor của IDE) kèm theo chú thích (comment) từng khối code theo các sơ đồ khối mô tả trong phần II.

C. Hàm tự tương quan

```
function [F0, timeF0, lengthF0] = AutoCorrelationFunction(y,Fs)
frame_time = 0.03;
frame_length = Fs*frame_time;
lengthF0 = 1;
% Sử dụng thuật toán tự tương quan
for i = 1 : length(y)/frame_length
        % Xác định từng khung 30ms của tín hiệu đầu vào
        area = (i-1)*frame_length + 1 : i*frame_length;
        frame_current = y(area);
        % Xác định biên độ cực đại của khung 30ms
        max_amplitude = max(frame_current);
        % Sử dụng hàm XCORR trong Matlab để tính tự tương quan
        [r,lags] = xcorr(frame_current,frame_current);
        % Vì R(k) là hàm chẵn nên chỉ cần tính toán trong khoảng (0;+∞)
        r = r(frame_length : end);
        lags = lags(frame_length : end);
        % Cho những tín hiệu có biên độ âm thành giá trị 0 để thuận tiện cho việc xác định F0
        % Sử dụng hàm FIND trong Matlab
        r(find(r < 0)) = 0;
        % Xác định các đỉnh cực đại liền kề nhau
         j = 1;
        for k = 1 : 3
                [value_peak,location_peak] = max(r(1 : end));
                location(j)= location_peak;
                end_of_peak_center = find(r(location(j) + 1:end) == 0 , 1);
                r(1 : location(j) + end_of_peak_center) = 0;
```

```
            j = j + 1;
        end
        % Xác định tính tuần hoàn của tín hiệu 30ms
        for j = 1 : 2
            T(j) = location(j+1) - location(j);
        end
        error = 5;
        if (T(1) < T(2) + error) || (T(1) == T(2)) || (T(2) < T(1) + error)
            fundamental_period = abs(T(1))/Fs;
            fundamental_frequency = 1/fundamental_period;
            % Kiểm tra F0 có thuộc dải tần số phù hợp (75-400 Hz)
            if (max_amplitude > 0.05) && (fundamental_frequency < 401) &&
        (fundamental_frequency > 74)
                F0(lengthF0) = fundamental_frequency;
            else
                F0(lengthF0) = NaN;
            end
        end
        timeF0(lengthF0) = i*frame_time;
        lengthF0 = lengthF0 + 1;
end
lengthF0 = lengthF0 - 1;
```

D. Tìm F0 trên miền tần số:

```
function [F0, timeF, lengthF0] = auto_f_frequency(y,Fs)

frame_time = 0.03;
frame_length = frame_time .* Fs; % Độ dài mẫu tín hiệu

for i = 1:length(y)/frame_length
   area = (i-1) * frame_length + 1:i*frame_length
   frame = y(area); %Hàm cửa sổ chữ nhật

   n = 2048; %Số điểm lấy mẫu của hàm FFT
   y_fft=abs(fft(frame,n)); % dùng hàm FFT, phân tích phổ biên độ
   y_fft=y_fft(1:round((length(y_fft)/2))); %Lấy phần phổ thực
   y_fft=10.*log(y_fft); % chuyên sang dB
   F=0:Fs/n:Fs/2-1; %Vecto tan so;

   [pks,locs]=findpeaks(y_fft,'MinPeakDistance',75/(Fs/n),'MinPeakProminence',10,'MinPeakHeight',5);
%tTìm các đỉnh trên phổ, sử dụng các điều kiện chọn được theo kinh nghiệm
   m = 3; %Số đỉnh đầu tiên cần lấy
   if (length(locs)<=m)
      F0(i)= NaN; %Nếu không đủ số điểm thì loại
   else
      locs = locs(1:m+1); %Lấy m đỉnh đầu tiên
      for j=1:m
         locs(j)=(locs(j+1)-locs(j))*(Fs/n); %Tính khoảng cách giữa các đỉnh
      end
      locs=locs(1:m);
      if (mean(locs)<=400) %Chặn tần số 75 <= f <= 400
         F0(i) = mean(locs); %Tính trung bình khoảng cách các đỉnh, tìm được F0 trên mẫu tín hiệu
      else
```

```
            F0(i) = NaN; %Tần số quá cao =>loại
        end
    end
    timeF(i) = i*frame_time; %Lưu vị trí trên miền thời gian của F0 vừa tính được
end
lengthF0=length(F0);
end
```

E. Lọc trung vị:

```
function [new_signal] = MedianFilter(y,N)
% Tín hiệu vào y, độ dài khung lọc là N. Kết quả trả về [new_signal].
    a = y;                  % Khởi tạo giá trị cho mảng a .
    M = length(a);          % M là độ dài của mảng a.
    tmp = ceil((N - 1) / 2); % tmp là số phần tử cần thêm vào đầu và cuối mảng a.
    M1 = M + N - 1;         % Số phần tử của mảng mới.
    b = 1 : M1;             % Tạo trục thời gian cho mảng b có độ dài M1.
    d = 1 : M;              % Tạo trục thời gian cho mảng d có độ dài M.
    % Thực hiện việc tạo mảng b.
    for i = 1 : tmp         % Vòng lặp thực hiện việc thêm tmp phần tử 0 vào đầu mảng b.
        b(i) = 0;
    end

    for i = tmp + 1 : tmp + M       % Vòng lặp chạy từ tmp+1 đến tmp+M .
        b(i) = a(i - tmp);          % Tạo ra mảng b có các giá trị từ mảng a.
    end

    for i = M + tmp + 1 : M1        % Vòng lặp thực hiện việc thêm tmp phần tử 0 vào cuối mảng b.
        b(i) = 0;
    end

    tmp2 = 1;                       %  Gán giá trị tmp2 ban đầu là 1.
    tmp1 = (N - 1) / 2 + 1;         % Giá trị cần lấy sau khi sắp xếp mảng.

    for i = 1 : M                   %  Vòng lặp chạy từ 1 đến M.
        ptu = 0;                    % Gán giá trị ptu ban đầu là 0.
        for j = tmp2 : tmp2 + N - 1 % Vòng lặp chạy từ tmp2 đến tmp2+N-1.
            ptu = ptu + 1;          % Đếm số lượng phần tử.
        end
        c = 1 : ptu;                %  Tạo trục thời gian cho mảng c.
        ptu = 0;                    % Gán giá trị ptu ban đầu là 0.
        for j = tmp2 : tmp2 + N - 1 % Vòng lặp chạy từ tmp2 đến tmp2+N-1.
            ptu = ptu + 1;          % Tăng giá trị ptu lên 1.
            c(ptu) = b(j);          % Gán giá trị c tại vị trí phần tử bằng giá trị của b tại vị trí j.
        end

        tmp2 = tmp2 +1;             % Dịch giá trị, tăng giá trị tmp2 lên 1 đơn vị.
        c = sort(c);                % Hàm có sẵn của MATLAB, sắp xếp mảng c theo thứ tự tăng dần.
        d(i) = c (tmp1);            % Kết quả cần tìm tại vị trí I là giá trị c tại phần tử tmp1.
    end

    new_signal = d;                 %  Trả về kết quả là tín hiệu sau khi lọc.
end
```

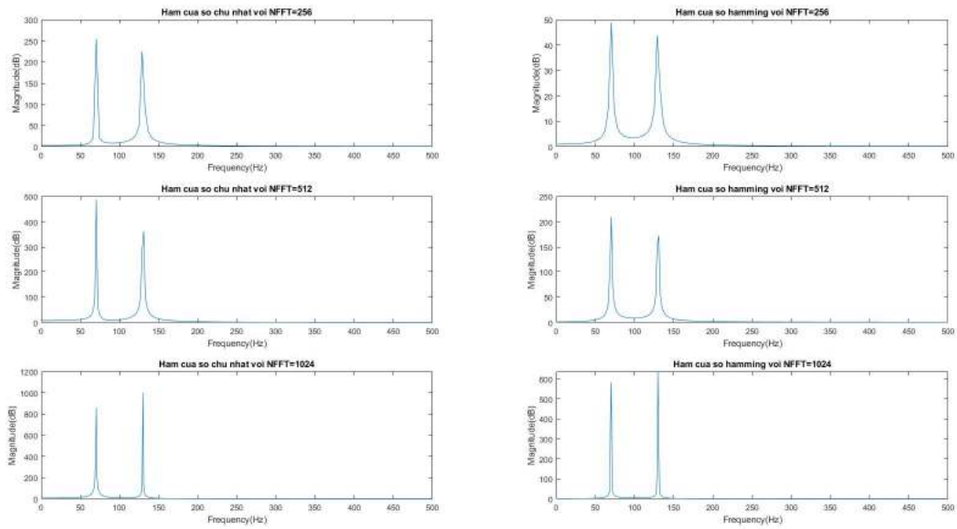
IV. KẾT QUẢ THỰC NGHIỆM

Mô tả dữ liệu dùng để đánh giá độ chính xác của các thuật toán, đưa ra các đánh giá định tính và định lượng, so sánh các thuật toán đã cài đặt với nhau và với các cài đặt (hoặc thuật toán) khác.

A. Hình vẽ

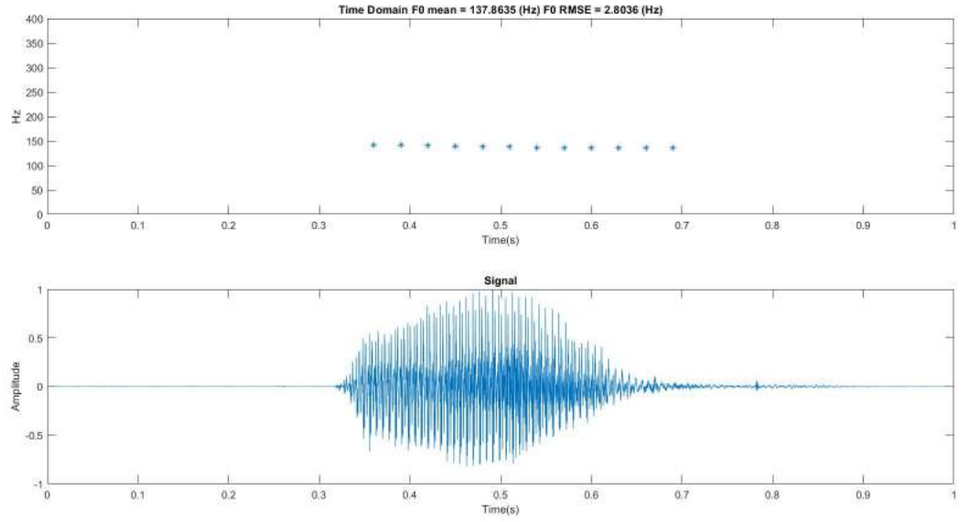
Hình vẽ trong bài viết được đánh số bắt đầu từ 1, được canh lề Justified, lời chú thích được viết dưới hình vẽ với kích thước font chữ là 9pt như Hình 1.

i) Khảo sát ảnh hưởng của hàm cửa sổ và số điểm tính FFT khi phân tích phổ tín hiệu hình sin

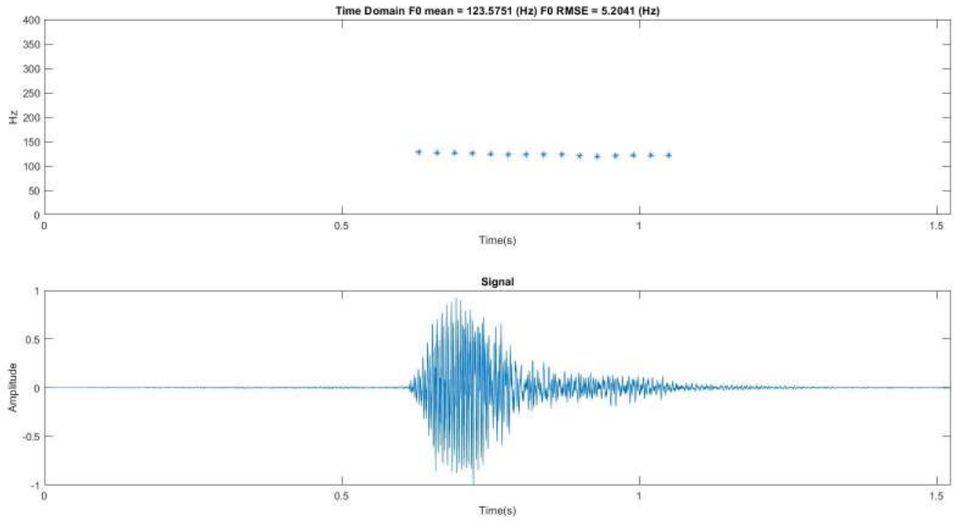


Hình 1. Kết quả khảo sát ảnh hưởng của hàm cửa sổ và số điểm tính FFT

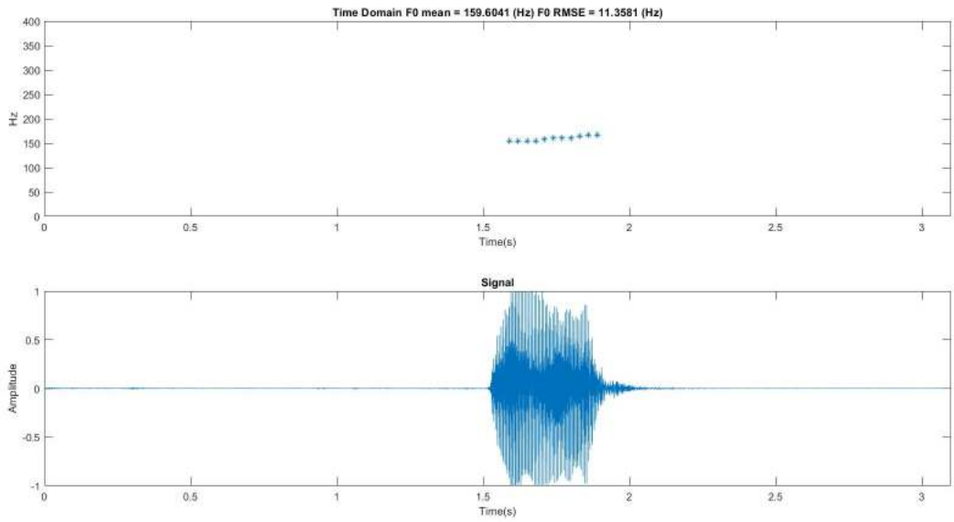
ii) Hàm tự tương quan:



Hình 2. Kết quả tính F0 trong trường hợp tốt nhất

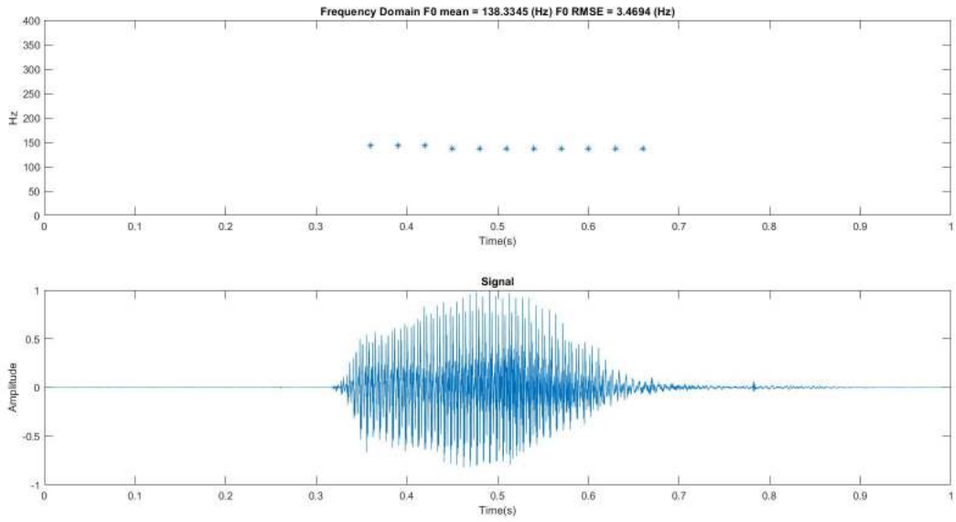


Hình 3. Kết quả tính F0 trong trường hợp trung bình

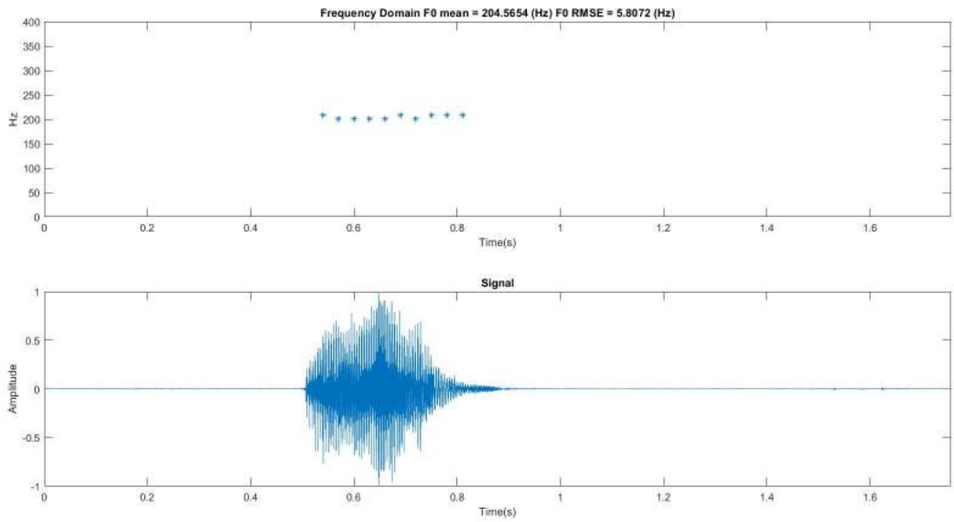


Hình 4. Kết quả tính F0 trong trường hợp xấu nhất

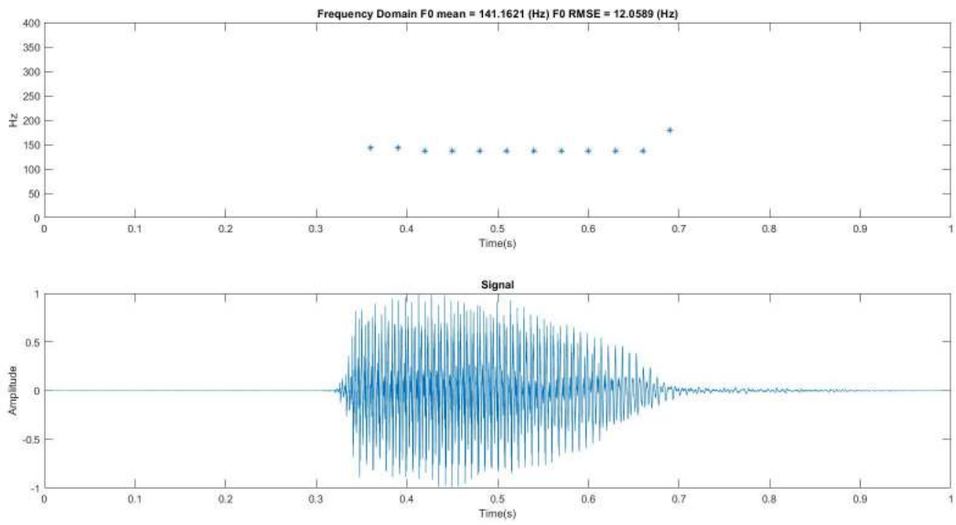
iii) Trên miền tần số



Hình 5. Kết quả tính F0 trong trường hợp tốt nhất



Hình 6. Kết quả tính F0 trong trường hợp trung bình



Hình 7. Kết quả tính F0 trong trường hợp xấu nhất

V. KẾT LUẬN

1.Kết quả đã đạt được:

Hiểu và hoàn thành tương đối hoàn chỉnh việc cài đặt thuật toán tự tương quan để tìm tần số cơ bản của tín hiệu tiếng nói trên miền thời gian và thuật toán dùng phép biến đổi Fourier nhanh trong Matlab để tìm tần số cơ bản của tín hiệu trên miền tần số. Sử dụng thuật toán lọc trung vị để làm trơn tín hiệu sau khi đã tiến hành tiền xử lí. Sự chênh lệch giữa các kết quả tính tần số cơ bản của mỗi nguyên âm càng cao cho thấy tầm quan trọng của lọc trung vị trong việc làm giảm sai số.

2.Hướng phát triển trong tương lai:

Tiến hành cải tiến thuật toán tìm tần số cơ bản để cải thiện độ chính xác của việc tính toán tần số cơ bản trên tín hiệu tiếng nói của con người. Ngoài ra, trong tương lai sẽ tiến hành thực nghiệm các thuật toán để xử lí tín hiệu vào là một câu nói dài, thay vì chỉ là một nguyên âm ngắn như đã làm.

VI. NHỮNG ĐIỀU ĐÃ HỌC ĐƯỢC

1.Về kiến thức:

Tích lũy thêm nhiều kiến thức bổ ích trong lĩnh vực xử lí tín hiệu, cụ thể là xử lí tín hiệu tiếng nói.

2.Về kĩ năng:

Thông qua bài báo cáo, mỗi thành viên đã học được cách làm việc nhóm, giao tiếp trong nhóm, phân công công việc và giúp đỡ nhau khi làm việc. Ngoài ra, mỗi thành viên còn tích lũy được kinh nghiệm về việc trình bày báo cáo, tìm nguồn tài liệu và lọc được những thông tin bổ ích từ tài liệu giáo viên đã giao cũng như nguồn tài nguyên trên Internet. Thêm vào đó, khả năng đọc hiểu Tiếng Anh và phân công thời gian làm việc để hoàn thành đúng hạn deadline cũng được nâng cao.

VII. TÀI LIỆU THAM KHẢO

[1] Link: https://en.wikipedia.org/wiki/Window_function
[2] Link: https://en.wikipedia.org/wiki/Fourier_transform
[3] Link: https://en.wikipedia.org/wiki/Biến_đổi_Fourier_nhanh

[4] (1.1),(1.2) textbook “Prentice Hall - Digital Processing Of Speech Signals_1978.pdf” phần 4.2 (*Bai Giang XLTN_HV CNBCVT trang 39-40),

(chú ý chỉ đưa vào các tài liệu có trích dẫn [1], [2], … trong báo cáo)