**TÌM TẦN SỐ CƠ BẢN CỦA TÍN HIỆU TRÊN MIỀN THỜI GIAN DÙNG HÀM TỰ TƯƠNG QUAN VÀ TRÊN MIỀN TẦN SỐ, KẾT HỢP LỌC TRUNG VỊ.**

## Nhóm số 2:

## Hồ Gia Khánh

## Đinh Quang Duy

## Hồ Trọng Hiền

## Nguyễn Xuân Lực.

**Lớp sinh hoạt: 16T3**

**Lớp HP:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Điểm** | **Bảng phân chia nhiệm vụ** | | **Chữ ký sinh viên** |
|  | Hồ Gia Khánh | Đọc tài liệu, cài đặt thuật toán, viết báo cáo, làm slide và thuyết trình về hàm lọc trung vị. |  |
|  | Đinh Quang Duy | Đọc tài liệu, cài đặt thuật toán, viết báo cáo, làm slide và thuyết trình về hàm tự tương quan. |  |
|  | Nguyễn Xuân Lực | Đọc tài liệu phân tích phổ, tìm F0 trên miền tần số, làm slide và thuyết trình về F0 trên miền tần số. |  |
|  | Hồ Trọng Hiền  (nhóm trưởng) | -- |  |

***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****— Tìm tần số cơ bản của tín hiệu là bài toán cần thiết trong xử lý tín hiệu âm thanh, đặc biệt là tín hiệu tiếng nói. Bài thực hành này thực hiện việc xác định tần số cơ bản của tín hiệu tiếng nói trên miền thời gian dùng hàm tự tương quan và trên miền tần số. Để làm trơn kết quả tần số cơ bản tìm được, lọc trung vị được sử dụng. Các thử nghiệm* *với tín hiệu của 5 nguyên âm (/a/, /e/, /i/, /o/, /u/) cho thấy sai số trung bình của thuật toán hàm tự tương quan có là 123Hz. Kết quả thực nghiệm cũng cho thấy ta có thể xác định được tần số cơ bản của tín hiệu tiếng nói trên miền thời gian và miền tần số. Tuy nhiên, việc phát triển một thuật toán để tính chính xác tần số cơ bản trên miền tần số là không dễ vì thường gặp những lỗi như rò phổ và việc thay đổi tần số thường xuyên trên tín hiệu có nhiễu. Phép lấy cửa sổ*

***Từ khóa****— Tính/đo tần số cơ bản, tính/đo chu kỳ pitch, hàm tự tương quan, lọc trung vị.*

## I. ĐẶT VẤN ĐỀ

Xử lý tiếng nói từ khi xuất hiện đã có một vai trò rất quan trọng trong cuộc sống của chúng ta. Cùng với sự phát triển của khoa học kỹ thuật, nhu cầu xử lý tiếng nói của con người ngày càng tăng cao. Xử lý tiếng nói có ứng dụng về mọi mặt, về cơ bản có ứng dụng như nhận dạng tiếng nói, người nói, tăng chất lượng giọng nói tổng hợp tiếng nói. Để làm được điều đó, việc xác định tần số cơ bản là rất quan trọng. Có nhiều phương pháp khác nhau như AMDF, LPC, xử lý đồng hình, tự tương quan, … để xác định được tần số cơ bản.

Tần số cơ bản (còn gọi là F0 hoặc cao độ) của một tín hiệu tuần hoàn bằng nghịch đảo của chu kỳ cơ bản của tín hiệu đó . Chu kỳ cơ bản là khoảng thời gian nhỏ nhất của tín hiệu tuần hoàn trên miền thời gian. Tần số cơ bản mang thông tin có ý nghĩa vật lý đặc trưng cho tín hiệu tuần hoàn nên việc xác định nó là rất quan trọng trong xử lý tín hiệu số nói chung và tín hiệu giọng nói nói riêng. Trong bài báo cáo này, hàm tự tương quan được sử dụng để tính tần số cơ bản trên miền thời gian và sử dụng phép biến đổi nhanh Fourier (FFT) để phân tích phổ và tính tần số cơ bản trên miền tần số. Lọc trung vị (median smoothing) được sử dụng để làm trơn kết quả F0 thu được.

**Bố cục của bài báo:**

* Phần II trình bày tổng quan về cơ sở lý thuyết, sơ đồ khối, cài đặt thuật toán của hàm tự tương quan và các vấn đề phát sinh cũng như các giải pháp khắc phục khi sử dụng hàm tự tương quan.
* Phần III trình bày tổng quan về cơ sở lý thuyết, sơ đồ khối, cài đặt thuật toán của việc phân tích phổ tín hiệu và các vấn đề phát sinh cũng như các giải pháp khắc phục khi sử dụng phép biến đổi nhanh Fourier.
* Phần IV trình bày tổng quan về cơ sở lý thuyết, cài đặt thuật toán của lọc trung vị (median smoothing) và các vấn đề phát sinh cũng như các giải pháp khắc phục.
* Phần V khảo sát ảnh hưởng của việc dùng hàm cửa sổ chữ nhật hoặc hàm cửa sổ Hamming và số điểm tính FFT khi phân tích phổ của tín hiệu.
* Phần VI là các kết quả thực nghiệm với tín hiệu của 5 nguyên âm (/a/, /e/, /i/, /o/, /u/).
* Phần VII trình bày kết luận.
* Phần VIII là lời cảm ơn.
* Phần IX là tài liệu tham khảo.

## II. TÌM TẦN SỐ CƠ BẢN CỦA TÍN HIỆU TRÊN MIỀN THỜI GIAN DÙNG HÀM TỰ TƯƠNG QUANG (AUTOCORRELATION)

### 1. Vấn đề cần giải quyết

### Xử lý tín hiệu số đang bùng nổ nhanh chóng trong ngành công nghiệp điện tử và viễn thông hiện nay bởi nhiều lợi thế hơn so với xử lí tín hiệu liên tục. Xử lý tín hiệu số có nhiều ứng dụng đa dạng, đặc biệt trong việc xử lí tiếng nói hay xử lí âm thanh. Trong quá trình xử lí tín hiệu tiếng nói, việc xác định tần số cơ bản của tín hiệu là vấn đề cần thiết trong bất cứ bài toán nào. Vậy tần số cơ bản là gì? Tần số cơ bản của một tín hiệu tiếng nói chính là tốc độ rung của dây thanh [1], được tính bằng nghịch đảo của chu kỳ cơ bản của tín hiệu, chu kỳ được xác định bằng khoảng thời gian ngắn nhất mà tín hiệu lặp lại trên miền thời gian [2].

### Trong xử lí tín hiệu số, có rất nhiều phương pháp để xác định tần số cơ bản của tín hiệu tiếng nói, tuy nhiên trong báo cáo này, chúng ta sẽ sử dụng một trong những thuật toán phổ biến để xác định chu kỳ cơ bản của tín hiệu trên miền thời gian bằng cách ứng dụng hàm tự tương quan, từ đó tính được tần số cơ bản của tín hiệu tiếng nói.

### 2. Cơ sở lý thuyết của thuật toán

*A.* ***Phân tích tiếng nói trong miền thời gian (Time – Dependent Processing of Speech)***

Các đặc điểm của tín hiệu tiếng nói luôn biến đổi theo thời gian. Chẳng hạn như cao độ (pitch), mức độ kích thích giữa âm vô thanh, âm hữu thanh và khoảng lặng, các biên độ khác nhau của tín hiệu, hay sự biến thiên tần số cơ bản trong miền tín hiệu tuần hoàn. Chính vì các đặc điểm trên luôn luôn biến đổi nên giả định trong hầu hết các phương pháp xử lí tín hiệu tiếng nói là các thành phần trong tín hiệu tiếng nói thay đổi không đáng kể hoặc cố định so với thời gian. Để áp dụng giả định trên, trong lý thuyết phân tích, các phân tích phải được tiến hành trong một khoảng thời gian giới hạn. Vì vậy nên khi đi phân tích tín hiệu tiếng nói có khoảng thời gian tương đối dài, phải áp dụng các phương pháp xử lí ngắn hạn nhằm chia tín hiệu thành từng đoạn (segment) hay từng khung nhỏ (frame) trong một khoảng thời gian tương đối nhỏ (thường khoảng từ 10-30ms). Các đoạn này thường được lấy chồng lên nhau nhằm tránh việc mất mát tín hiệu.

Trong hầu hết các phương pháp xử lí ngắn hạn, việc chia tín hiệu thành các đoạn hoặc khung được thực hiện bằng cách nhân tín hiệu tiếng nói với 1 hàm cửa sổ có chiều dài hữu hạn tương ứng với chiều dài của khung tín hiệu mong muốn.

Hàm cửa sổ trong xử lí tín hiệu số là một hàm toán học có giá trị 0 tại những điểm không thuộc khoảng giới hạn (chiều dài cửa sổ). Sau khi nhân tín hiệu với một hàm cửa sổ (phép lấy cửa sổ) ngoại trừ những vị trí thuộc chiều dài cửa sổ, tất cả các vị trí không thuộc chiều dài cửa sổ đều có giá trị bằng 0. Tuy nhiên, việc lấy cửa sổ tín hiệu không chỉ nhằm phân đoạn tín hiệu, mà mục đích chính của việc này là làm thon tín hiệu, nhằm cải thiện các đặc điểm của tín hiệu trên miền tần số.

Hàm cửa sổ có nhiều loại, phụ thuộc vào mục đích sử dụng khi phân tích tín hiệu. Có nhiều hàm cửa sổ được sử dụng trong việc phân tích tín hiệu sẽ được trình bày tại Phần V. Trong phần này, hàm cửa sổ được sử dụng là Hamming.

B. ***Hàm tự tương quan (Autocorrelation Function)***

Hàm tự tương quan là công cụ được sử dụng phổ biến để xác định chu kỳ cơ bản của tín hiệu tiếng nói (có thể lẫn nhiễu) và nó cũng là cơ sở cho nhiều phương pháp phân tích phổ khác. Với tín hiệu tuần hoàn, ta có định nghĩa hàm tự tương quan:

rxx(l) =

Với N là độ rộng của cửa sổ, l là độ trễ được tính tại thời điểm n.

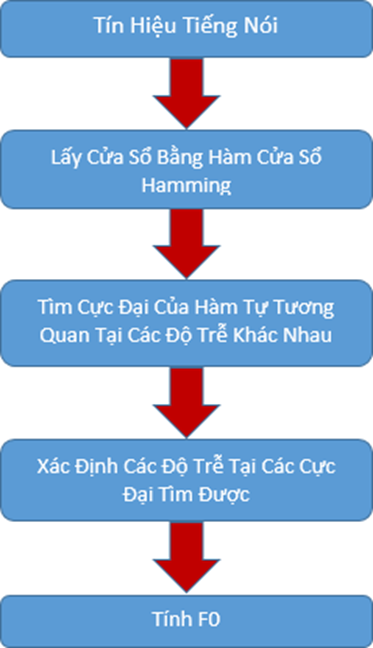


Hình 1. Tín hiệu và hàm tự tương quan của nó.

Hàm tự tương quan là hàm chẵn, đạt giá trị cực đại tại l = 0. Đại lượng rxx(0) chính bằng năng lượng của tín hiệu. Tầm quan trọng hàm tự tương quan nằm ở việc hàm sẽ đạt các giá trị cực đại tương ứng tại các điểm là bội của chu kỳ cơ bản của tín hiệu. Khi đó các tần số cơ bản là tần số xuất hiện của các cực đại đó. Tính chất này khiến hàm tự tương quan trở thành cơ sở cho việc tính toán chu kỳ của tất cả các loại tín hiệu, bao gồm cả tín hiệu tiếng nói.

### 3. Sơ đồ khối

Sơ đồ khối thuật toán hàm tự tương quan được trình bày trên Hình 2



Hình 2. Sơ đồ khối hàm tự tương quan.

Tín hiệu tiếng nói

Lấy cửa sổ bằng hàm cửa sổ Hamming

### 4. Các tham số quan trọng của hàm tự tương quan

Công thức tính tần số cơ bản của tín hiệu: F0 = Fs / L, trong đó F0 là tần số cơ bản, Fs là tần số lấy mẫu và L là độ trễ mà tại đó hàm tự tương quan đạt cực đại.

Vì tần số lấy mẫu Fs đã biết, nên để tính tần số cơ bản, ta phải đi tìm độ trễ L.

Vì tín hiệu tiếng nói gần như tuần hoàn, nên hàm tự tương quan sẽ cho ra các cực đại tại những thời điểm là bội số của chu kỳ tín hiệu[2]. Từ đồ thị hàm tự tương quan, ta sẽ xác định các cực đại và độ trễ tương ứng tại các cực đại đó, từ độ trễ ta sẽ xác định được tần số cơ bản của tín hiệu.

Do tín hiệu ban đầu được chia nhỏ thành từng khung qua phép lấy cửa sổ, đồng thời do tín hiệu tiếng nói chưa tuần hoàn hoàn hảo, nên ứng với mỗi khung tín hiệu sẽ có 1 giá trị F0, các giá trị này xấp xỉ nhau, bằng việc chia nhỏ tín hiệu và sử dụng hàm tự tương quan lên mỗi khung tín hiệu, kết quả thu được là mảng 1 chiều có số phần tử bằng số khung tín hiệu, mỗi phần tử là 1 tần số cơ bản tương ứng với 1 khung tín hiệu.

### 5. Cài đặt thuận toán trên Matlab

Chương trình sẽ đọc tín hiệu từ các file nguyên âm /a/, /i/, /u/, /e/, /o/ đã thu từ 2 thí nghiệm đã làm trước đó, và tính toán được chu kỳ cơ bản của các nguyên âm trên, từ đó rút ra tần số cơ bản.

Trong thư viện Matlab, sử dụng hàm xcorr với đối số truyền vào là tín hiệu, sử dụng hàm plot vẽ được đồ thị hàm tự tương quan, sau đó xác định các cực đại địa phương để tìm độ trễ tại các cực đại đó, từ độ trễ ta tính được tần số cơ bản F0. Kết quả thủ được sẽ đi qua bộ lọc trung vị (được trình bày Phần IV), từ đó sẽ tính được tần số cơ bản của tín hiệu một cách tương đối chính xác.

*%calculate the fundamental frequency of a speech signal*

*[ip , fs] = audioread('a.wav');*

*%T0max = 1/F0min \* Fs*

*%F0min = 80Hz*

*max\_expected\_period = round(1/80\*fs);*

*%T0min = 1/F0max \* Fs*

*%F0max = 400Hz*

*min\_expected\_period = round(1/400\*fs);*

*%create an array to store all calculated frequencies*

*result = zeros(1,fix (length(ip)/frame\_len -1));*

*frame\_len = 2\*max\_expected\_period; %length of each frame <= 2 \* T0 max*

*h = hamming(frame\_len); %create a impulse response of the hamming window*

*for k = 1 : length(ip)/frame\_len -1; % length(ip)/frame\_len -1 = number of frame*

*range = (k-1)\*frame\_len + 1:k\*frame\_len; % start position and end position*

*frame = h.\*ip(range); %value of each n multiplied by the hamming window*

*%show the input in blue and the selected frame in red*

*plot(ip); %draw the input speech signal*

*%adjust graphic properties*

*set(gca, 'xtick',[],'position',[ 0.05 0.82 0.91 0.13])*

*hold on; %used to add a new graph (temp\_sig) to the current graph*

*%crate an array that make all value non-number*

*temp\_sig = ones(size(ip))\*NaN;*

*temp\_sig(range) = frame;%put the frame into the array*

*plot(temp\_sig,'r');%draw the current frame*

*hold off*

*%use xcorr to determine the local period of the frame*

*[rxx , lag] = xcorr(frame, frame);%autocorrelation function on the frame*

*rxx(rxx < 0) = 0; %set any negative correlation values to zero*

*subplot(3,1,3)*

*plot(lag, rxx,'r') %draw the correlation graph rxx*

*center\_peak\_width = find(rxx(frame\_len:end) == 0 ,1); %find first zero after center*

*%center of rxx is located at length(frame)+1*

*%set zero to the central maxima*

*rxx(frame\_len-center\_peak\_width : frame\_len+center\_peak\_width ) = min(rxx);*

*%find next maxima, probably the nearest maxima from the central one*

*[max\_val , loc] = max(rxx);*

*%calculate the lag between the central maxima and its nearest maxima*

*period = abs(loc - length(frame)+1);*

*result(k) = fs/period;*

*title(['Period estimate = ' num2str(period) 'samples (' num2str(fs/period) 'Hz)']);*

*%adjust graphic properties to zoom*

*set(gca, 'position', [ 0.05 0.07 0.91 0.25])*

*subplot(3,1,2)*

*plot(frame);*

*set(gca, 'position',[ 0.05 0.47 0.91 0.33]) %adjust graphic properties to zoom*

*pause*

*end*

*figure(2)*

*stem(result,'filled'); %draw the frequency graph*

### 6. Vấn đề phát sinh và giải pháp

A. ***Phép lấy cửa sổ***

Do đặc tính của mỗi hàm cửa sổ, nên trong quá trình thực hiện việc lấy cửa sổ sẽ phát sinh hiện tượng gọi là rò phổ. Để xử lí vấn đề này, ta sẽ sử dụng cửa sổ Hamming cho phép lấy cửa sổ nhằm hạn chế tối đa khả năng rò phổ so với cửa sổ chữ nhật (rectangular) sẽ được phân tích ở Phần V.

1. ***Số lượng cực đại của khung tín hiệu***

Vì khung tín hiệu có chiều dài hữu hạn nên năng lượng cũng hữu hạn, điều này dẫn đến biên độ các đỉnh trong đồ thị tương quan sẽ giảm dần về 2 bên, với trục đối xứng là điểm chính giữa đồ thị. Điều này khiến cho việc xác định cực đại sẽ khó khăn, do đó ta chỉ lấy cực đại kế cận cực đại trung tâm, qua đó việc tính toán tần số cơ bản sẽ bớt phức tạp hơn.

## III. TÌM TẦN SỐ CƠ BẢN CỦA TÍN HIỆU TRÊN MIỀN TẦN SỐ

1. ***Vấn đề cần giải quyết***

Tần số cơ bản (còn gọi là F0 hoặc cao độ) của một tín hiệu tuần hoàn bằng nghịch đảo của chu kỳ tín hiệu đó [1]. Chu kỳ được xác định bằng khoảng thời gian ngắn nhất mà tín hiệu lặp lại trên miền thời gian. Tần số F0 là tần số giao động của dây thanh. Trong bài báo cáo này, tín hiệu được xử lý là tín hiệu tiếng nói nên tần số cơ bản phụ thuộc vào giới tính và độ tuổi. F0 của nữ thường cao hơn của nam, F0 của người trẻ thường cao hơn của người già. Thường với giọng của nam, F0 nằm trong khoảng từ 80-250Hz, với giọng của nữ, F0 trong khoảng 150-500Hz. Sự biến đổi của F0 có tính quyết định đến thanh điệu của từ cũng như ngữ điệu của câu. Câu hỏi đặt ra là làm thế nào để xác định tần số cơ bản (fundamental frequency).

Vào những năm thập kỷ 60, khi công nghệ vi xử lý phát triển chưa mạnh thì thời gian xử lý phép toán DFT (Discrete Fourier Transform: Biến đổi Fourier rời rạc: liên kết thời gian rời rạc và tần số rời rạc) trên máy tương đối chậm, do số phép nhân phức tương đối lớn. Để khắc phục về mặt tốc độ xử lý của phép tính DFT, nhiều tác giả đã đưa ra các thuật toán riêng dựa trên DFT gọi là FFT (Fast Fourier Transform: Biến đổi Fourier nhanh).

Phương pháp được sử dụng trong nghiên cứu này là phương pháp biến đổi nhanh Fourier (FFT) để xác định tần số cơ bản (F0) trên miền tần số kết hợp với lọc trung vị để làm trơn kết quả F0 thu được.

1. ***Cơ sở lý thuyết của thuật toán***
2. ***Phép biến đổi Fourier rời rạc (Discrete Fourier Transform –DFT)***

Trong toán học, phép biến đổi Fourier rời rạc, đôi khi còn được gọi là biến đổi Fourier hữu hạn, là một phép biến đổi trong giải tích Fourier cho các tín hiệu thời gian rời rạc. Đầu vào của biến đổi này là một chuỗi hữu hạn các số phức hoặc các số thực. 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. Biến đổi này được tính nhanh bởi thuật toán biến đổi Fourier nhanh (FFT)[1].

Biến đổi Fourier rời rạc cho phép tính phiên bản tần số rời rạc của biển đổi Fourier của tín hiệu rời rạc (Discrete-time Fourier Transform – DTFT).

Công thức DFT N-điểm

X[k] = X(ejω) với ω =và 0≤ k ≤N-1 (\*)

X[k] đại diện cho biên độ và pha ở các bước sóng khác nhau của tín hiệu vào x[n]. Phép biến đổi DFT tính các giá trị X[k] từ các giá trị x[n] [1].

1. ***Phép biến đổi Fourier nhanh (Fast Fourier Transform - FFT)***

Một biến đổi Fourier nhanh (FFT) là một thuật toán hiệu quả để tính biến đổi Fourier rời rạc (DFT) và biến đổi ngược. Có nhiều thuật toán FFT khác nhau sử dụng kiến thức từ nhiều mảng khác nhau của toán học [2].

Các thuật toán FFT tính nhanh DFT N-điểm của tín hiệu rời rạc x[n], trong đó, N = 2m length(x[n]).

Tính trực tiếp từ định nghĩa DFT (\*) đòi hỏi O(N2) phép tính: có N số Xk cần tính, để tính mỗi số cần tính một tổng N số hạng. Một FFT là một phương pháp để tính cùng kết quả đó trong O(N\*logN) phép tính.

Trong bài báo cáo này, sử dụng hàm fft() trong thư viện Matlab để phân tích phổ kết hợp với hàm findpeaks() để tìm tần số cơ bản.

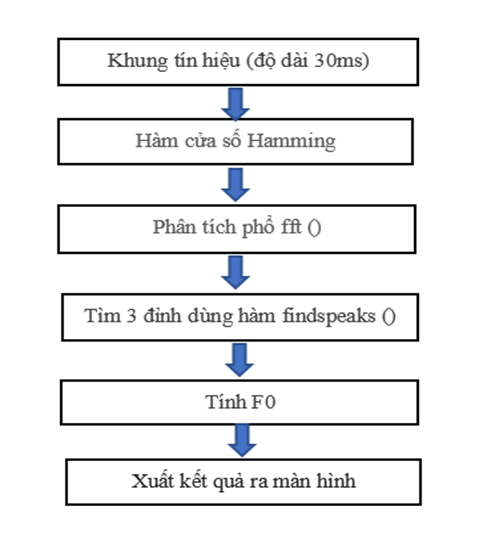


Hình 3 Tín hiệu và phổ tín hiệu của nó.

1. ***Sơ đồ khối***

Sơ đồ khối thuật toán tìm tần số cơ bản F0 trên miền tần số được trình bày trên Hình 4

>....

******

Hình 4. Sơ đồ khối tìm tần số cơ bản F0 trên miền tần số

1. ***Các tham số quan trọng của thuật toán***

Hàm fft() trong thư viện Matlab:

Cú pháp Y = fft(X,N) với X là tín hiệu vào (trên miền thời gian rời rạc) và N là số điểm trong phép biến đổi DFT-N điểm.

Nếu length(X) > N thì X được thêm vào các giá trị 0 cho tới khi length(X) = N để thực hiện phép biến đổi.

Nếu length(X) < N thì X được cắt ngắn sao cho length(X) = N.

Kết quả trả về (Y) là mảng chứa các số phức biển diễn tín phổ (spectrum) của x[n]

1. ***Cài đặt thuận toán trên Matlab***

*[ip , fs] = audioread('i.wav');*

*frame\_len = round(0.03\*fs); %Do dai khung tin hieu (30ms)*

*half = round(frame\_len/2);*

*h = hamming(frame\_len); %Hamming window function*

*yyy\_F0 = [];*

*i=1; %Index of number of elements in yyy\_F0*

*for k = 1 : length(ip)/half -1 %number of frame*

*range = (k-1)\*half + 1:(frame\_len + (k-1)\*half); %Index of each element in window*

*frame = h.\*ip(range); %Value of each element in window*

*%show the input in blue and the selected frame in red*

*plot(ip);*

*set(gca, 'xtick',[],'position',[ 0.05 0.82 0.91 0.13])*

*hold on;*

*temp\_sig = ones(size(ip))\*NaN;*

*temp\_sig(range) = frame;*

*plot(temp\_sig,'r');*

*hold off*

*subplot(3,1,2)*

*plot(frame);*

*set(gca, 'position',[ 0.05 0.47 0.91 0.33])*

*%use fft() to find the frequency of the frame*

*P2 = abs(fft(frame,1024)); %The two-sided spectrum P2*

*P1 = P2(1:length(P2)/2+1); %The single-sided spectrum P1*

*P1(2:end-1) = 2\*P1(2:end-1);*

*freq=linspace(1/fs,fs/2,length(P1));*

*subplot(3,1,3)*

*plot(freq,P1,'r')*

*xlabel ('Frequency F(Hz)');*

*set(gca, 'position', [ 0.05 0.07 0.91 0.25])*

*[yy\_value\_highest,y\_peak1] = findpeaks(P1,freq,'NPeaks',1,'SortStr','descend','MinPeakHeight',10);*

*[zz\_result1,zz\_result2] = findpeaks(P1,freq);*

*for j= 1:length(zz\_result2)*

*if zz\_result2(j) == y\_peak1*

*y\_peak2 = zz\_result2(j+1);*

*y\_peak3 = zz\_result2(j+2);*

*break;*

*end*

*end*

*if isempty(y\_peak1)*

*else*

*z\_a = y\_peak2 - y\_peak1;*

*z\_b = y\_peak3 - y\_peak2;*

*if z\_a<400 && z\_a>80*

*if z\_b<400 && z\_b>80*

*yyy\_F0(i) = (z\_a+z\_b)/2;*

*i=i+1;*

*end*

*end*

*end*

*pause*

*end*

*yyy\_l = length(yyy\_F0);*

*yyy\_F0\_mean = mean(yyy\_F0);*

*yyy\_F0\_std = std(yyy\_F0);*

*figure(2);*

*stem(yyy\_F0,'filled');*

*title(['Fundemental frequency F0 (Hz), F0mean = ',num2str(yyy\_F0\_mean),' F0std = ',num2str(yyy\_F0\_std)]);*

1. ***Vấn đề phát sinh và giải pháp***

Một bài toán mở quan trọng về mặt lý thuyết là chứng minh chặn dưới cho độ phức tạp tính toán và số phép tính của biến đổi Fourier nhanh. Hiện vẫn chưa có chứng minh nào cho việc DFT có thực sự đòi hỏi Ω(N\*logN) phép tính, ngay cả trong trường hợp kích thước N là lũy thừa của hai, mặc dù không có thuật toán nào có độ phức tạp thấp hơn. Chú ý rằng tuy số phép tính thường là quan tâm chính về mặt lý thuyết, nhưng trên thực tế, tốc độ thực thi phụ thuộc nhiều yếu tố khác như các tối ưu hóa cho bộ nhớ đệm và ống lệnh CPU (CPU Pipes). Theo công trình của Winograd năm 1978 , chặn dưới chặt cho số phép nhân của FFT đã được biết là Θ(N)[2].

Vấn đề về lỗi octave thường xảy ra vì tín hiệu thường có năng lượng cao tại các chu kỳ đầu, đặc biệt đối với những tín hiệu có formant đầu tiên thấp (ví dụ như âm /u/). Do đó, ta dễ gặp trường hợp envelope có đỉnh gần đỉnh cực đại của phổ tín hiệu. Rất khó có thể xác đinh được đâu là đỉnh của tần số nhỏ hơn và đâu là đỉnh của nhiễu để tìm tần số cơ bản chính xác trong mọi trường hợp [4].

Một vấn đề khác cũng rất phổ biến đó là sự khó khăn trong việc xác định được các khung lựa chọn là khung của âm thanh hay khung của nhiễu [4].

Trong việc tìm phổ của tín hiệu liên tục cũng như tín hiệu rời rạc, tất cả các giá trị của tín hiệu trên miền thời gian đều rất cần thiết. Tuy nhiên, trên thực tế, chúng ta chỉ có thể tính toán và quan sát trong khoảng thời gian giới hạn. Do vậy, phổ tín hiệu chỉ có thể được tính xấp xỉ từ lượng dữ liệu giới hạn thông qua cửa sổ. Từ đó, ta sẽ bị hạn chế khả năng phân biệt hai thành phần tần số được tách ra nhỏ hơn tần số cơ bản. Phổ trong cửa sổ sẽ trải dài ra khắp miền tần số nên năng lượng của tín hiệu gốc tập trung tại một tần số sẽ bị trải ra khắp miền tần số bởi hàm cửa sổ. Hiện tượng này được gọi là rò phổ, đây là hiện tượng rất phổ biến khi sử dụng hàm cửa sổ [5].

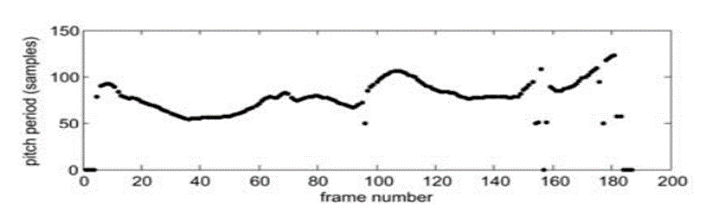
Giải pháp: Để giảm rò phổ, chúng ta nên lựa chọn những hàm cửa sổ có sidelobes nhỏ hơn hàm cửa sổ chữ nhật trên miền tần số. Trong bài thực hành này, chúng em sử dụng hàm cửa sổ Hamming.

## IV. LỌC TRUNG VỊ (MEDIAN SMOOTHING) DÙNG ĐỂ LÀM TRƠN F0 THU ĐƯỢC

1. ***Vấn đề cần giải quyết:***

Trong hầu hết trường hợp, một bộ lọc tuyến tính nói chung là loại bỏ các thành phần tạp âm ra khỏi tín hiệu. Tuy nhiên, trong một số trường hợp thì các bộ lọc tuyến tính không hoàn toàn phù hợp bởi loại tín hiệu cần được làm mịn. Một ví dụ là đường viền ở hình dưới đây, có những thành phần lỗi rõ ràng và phải được đưa trở lại phù hợp với phần còn lại của dữ liệu. Một bộ lọc tuyến tính lowpass không chỉ thất bại trong việc mang các điểm sai lệch trở lại dòng mà còn bóp méo đường viền tại quá trình chuyển đổi giữa giọng nói hữu thanh và vô thanh (giai đoạn 0).

Anfklsdlkfl (CHECK)



HÌNH 5

Đối với các trường hợp như vậy, một số thuật toán lọc tuyến tính có thể duy trì các tín hiệu gián đoạn nhưng tính chất lớn lọc ra không tồn tại, một bộ lọc phi tuyến sử dụng sự kết hợp giữa trung vị chạy (running median) và lọc tuyến tính được đưa ra để thỏa mãn tính chất mong muốn.

Một tín hiệu giọng nói x(n) sẽ có công thức: x(n) = S[x(n)] + R[x(n)].

Với S(x) là phần lọc (phần mịn), R(x) là phần thô (tạp âm) của tín hiệu x.

Một phép biến đổi phi tuyến có khả năng tách S[x(n)] ra khỏi R[x(n)] chính là trung vị chạy của tín hiệu x(n).

Đầu ra của trung vị chạy lọc, ML[x(n)] đơn giản là trung vị của L số, x(n), x(n-1), …, x(n-L+1).

Biến đổi trung vị với độ dài L tuân theo các tính chất mong muốn của bộ lọc: ML(ax[n]) = a. ML(x[n]). Các trung vị không bị nhòe gián đoạn (smeared out) nếu không có sự gián đoạn trong các mẫu .

ML(ax1[n] + bx2[n]) ≠ a.ML(x1[n]) + b.ML(x2[n]).

Các bộ lọc trung vị thường bảo toàn các gián đoạn sắc nét của tín hiệu, nhưng không đảm bảo việc lọc các thành phần tiếng ồn (tạp âm) một cách đầy đủ.ML(ax[n]) = a.ML(x[n])

1. ***Cơ sở lý thuyết của thuật toán***

Số trung vị là số nằm chính giữa trong một tập hợp dãy số (n phần tử) đã được sắp xếp theo thứ tự tăng dần.

Nếu số phần tử của dãy số là lẻ (n = 2k+1 phần tử), thì số trung vị sẽ là số ở vị trí thứ n + 1. Ví dụ trong dãy số 1, 2, 6, 7, 8, 16, 18 thì số trung vị sẽ là 7.

Nếu số phần tử của dãy là số chẵn (n = 2k phần tử), thì số trung vị sẽ là trung bình cộng của phần tử thứ n và thứ n+1. Ví dụ dãy số 1, 3, 4, 6, 7, 9, 15, 16 thì số trung vị sẽ là trung bình cộng của 6 và 7 (bằng 6.5).

1. ***Các tham số quan trọng của lọc trung vị (median smoothing)***

Trong matlab, đã có sẵn hàm thư viện để lọc trung vị của tín hiệu cần lọc. Đó là hàm median filter 1 (lọc trung vị 1 chiều) để lọc trung vị các tín hiệu một chiều. Cú pháp của hàm là medfilt1(x,N). Với x là tín hiệu đầu vào sau khi tính F0 bằng hàm tự tương quan (đã được rời rạc hóa) hoặc trên miền tần số, N là số chiều (bậc) của tín hiệu hay chiều dài hàm cửa sổ. Để lấy chính xác điểm trung vị, N thường là số lẻ.

Ví dụ cụ thể về lọc trung vị:

Xét đoạn code sau và chạy, dùng bộ lọc medfilt1 với bậc N là 5.

CODE

Kết quả của lọc trung vị được thể hiện ở Hình 6.



Hình 6. Tín hiệu trước và sau khi đi qua lọc trung vị.

Xét tín hiệu ban đầu, tại n = -5 thì giá trị của x là 6. Áp dụng lọc trung vị cho tín hiệu x với bậc là 5, tức là ta xét các giá trị của x tại n = -7, -6,-5,-4,-3. Mỗi giá trị tại đó tương ứng với 0, 0 , 6, 5, 2. Sau khi sắp xếp tăng dần và chọn trung vị, ta được số trung vị là 2. Và 2 sẽ được chọn là giá trị cho tín hiệu x sau khi lọc tại n = -5.

Một ví dụ khác, tại n = 0, giá trị của x sẽ là -50, sau khi lọc tín hiệu x với bậc là 5 (xét các giá trị n = -2, -1, 0, 1, 2 với các giá trị là -9 , 4, -50, 10, 1) thì giá trị của x tại n = 0 là -4.



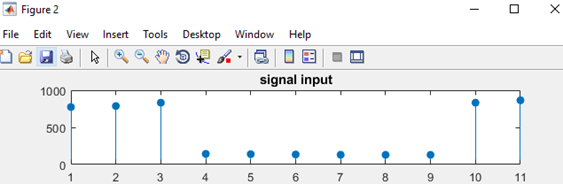
Hình 7 (fjklalsdkga;laktgahl;ad)

Từ ví dụ trên, có thể thấy rằng tác dụng của lọc trung vị là lọc những điểm có biên độ bất thường của tín hiệu, hay nói cách khác là làm mịn tín hiệu. Bậc N của bộ lọc cần phải được xem xét để tín hiệu sau khi lọc không được còn những điểm bất thường hoặc quá trơn.

Xét các nguyên âm cơ bản /a/, /e/, /i/, /o/, /u/ trước khi lọc (sau khi được tính F0 bằng hàm tự tương quan và đã được rời rạc hóa) và sau khi lọc trung vị với các bậc khác nhau:

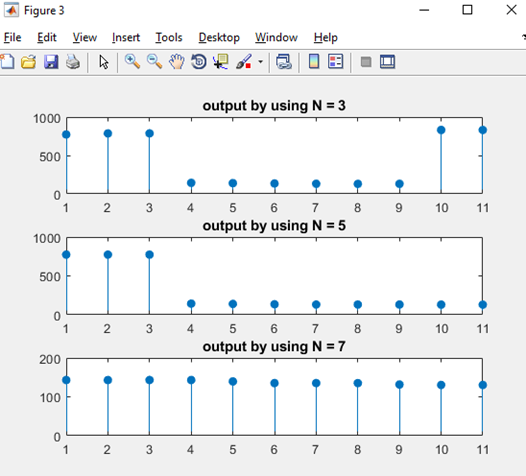
* Âm /a/:

Trước khi lọc:



HÌNH 8

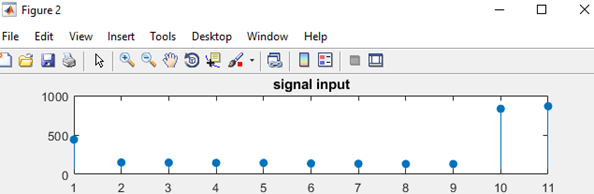
Sau khi lọc:



HÌNH 9

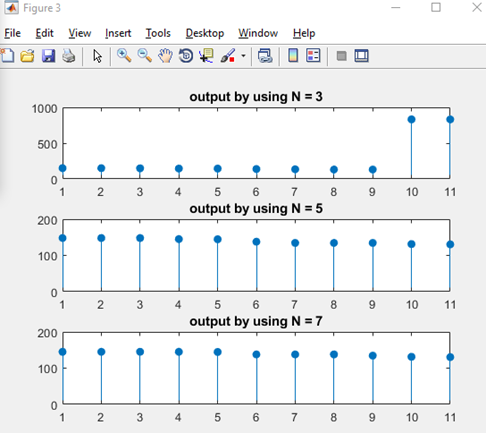
* Âm /e/

Trước khi lọc:



HÌNH 10

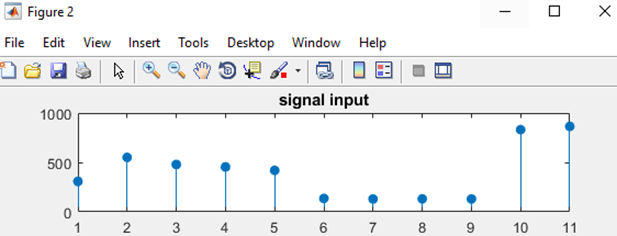
Sau khi lọc:



HÌNH 11

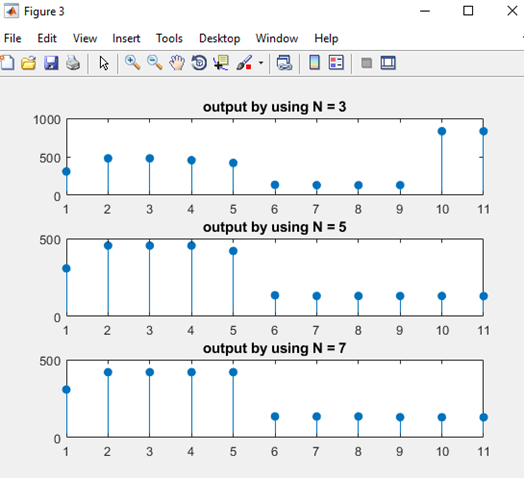
* Âm /i/

Trước khi lọc:



HÌNH 12

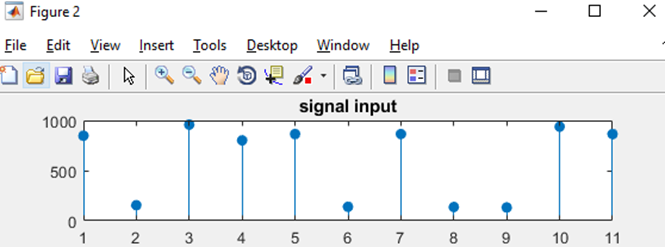
Sau khi lọc:



HÌNH 13

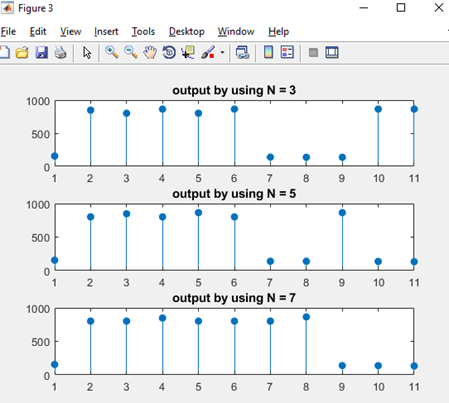
* Âm /o/

Trước khi lọc:



HÌNH 14

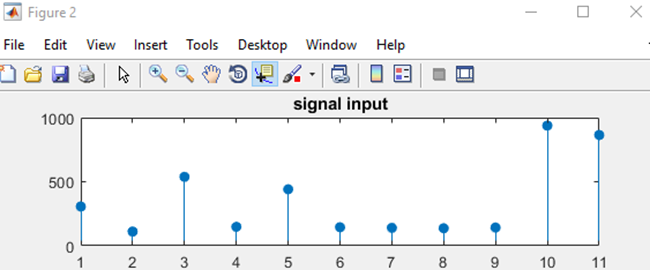
Sau khi lọc:



HÌNH 15

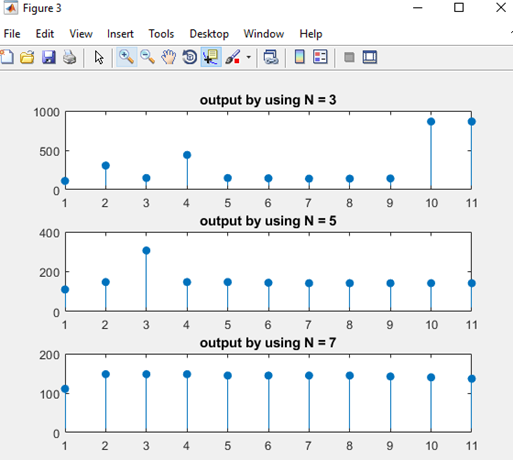
* Âm /u/

Trước khi lọc:



HÌNH 16

Sau khi lọc:



HÌNH 17

Từ những đồ thị trên, có thể thấy rằng nếu bậc N của lọc trung vị thấp (N = 3) thì tín hiệu vẫn không thay đổi nhiều, vẫn còn nhiều điểm bất thường ở các giá trị n. Tín hiệu đã được lọc mịn hơn tại bậc N = 5 và bậc N = 7. Với bậc N = 5 thì các tín hiệu vẫn còn một vài giá trị bất thường (tại nguyên âm /a/) nhưng các giá trị n khác của tín hiệu cũng giữ được độ chính xác. Khi bậc N = 7 thì hầu như các giá trị bất thường không còn, nhưng có một vài giá trị n đã bị thay thế (mất) thông tin. Giá trị N càng lớn thì tín hiệu càng mịn, tuy nhiên tín hiệu cũng sẽ càng mất tín hiệu quan trọng. Vì vậy, giá trị N nên là 5 hoặc 7 để tín hiệu có thể vừa loại bỏ được những giá trị bất thường của tín hiệu, vừa giữ được các thông tin quan trọng.

## V. KHẢO SÁT ẢNH HƯỞNG CỦA VIỆC DÙNG HÀM CỬA SỔ VÀ SỐ ĐIỂM TÍNH FFT

VI. **KẾT QUẢ KHẢO SÁT**

VII. **KẾT LUẬN**

Bài báo này thực hiện việc cài đặt thuật toán tìm tần số cơ bản của tín hiệu tiếng nói dùng hàm tự tương quan được cài đặt trên Matlab. Để làm trơn tần số cơ bản được xác định, lọc trung vị được sử dụng. Các thử nghiệm với tín hiệu của 5 nguyên âm (/a/, /e/, /i/, /o/, /u/) cho thấy …….

Trong tương lai chúng tôi sẽ thử nghiệm các thuật toán tính tần số cơ bản tiên tiến hơn để cải thiện độ chính xác của ……….

VIII. **LỜI CẢM ƠN**

## Cảm ơn thầy Ninh Khánh Duy, giảng viên khoa Công nghệ thông tin, trường Đại học Bách Khoa, Đại học Đà Nẵng đã đồng hành cùng chúng em trong quá trình nghiên cứu và tìm hiểu đề tài.

## IX. TÀI LIỆU THAM KHẢO

1. Alain de Cheveigne, Hideki Kawahara, “YIN, a fundamental frequency estimator for speech and music”, Journal of the Acoustical Society of America, vol. 111, no. 4, pp. 1917-1930, 2002.
2. [1] Tần số âm cơ bản, Wikipedia https://vi.wikipedia.org/wiki/Tần\_số\_âm\_cơ\_bản
3. [2] Nguyen Binh Thien, Ninh Khanh Duy, Bài Báo Fair 2017
4. [3] Prentice Hall - Digital Processing Of Speech Signals 1978
5. Abigail Lira, “Implementing a Pitch Detection Algorithm to Tune a Bass Guitar”, Technical report, El Paso Community College, 2015.
6. Cubic Spline Interpolation. Retrieved 2016-11-05, from https://en.wikiversity.org/wiki/Cubic\_Spline\_Interpolation.
7. Donald S. Reay, “Hands-on real-time DSP teaching using inexpensive ARM Cortex-M4 development systems”, Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 22242227, 2014.
8. David Talkin, “A robust algorithm for pitch tracking (RAPT),” in Speech Coding and Synthesis, eds. W. B. Kleijn and K. K. Paliwal, pp. 495–518, Elsevier, New York, 1995.
9. Duy Khanh Ninh, Yoichi Yamashita, “F0 parameterization of glottalized tones in HMM-based speech synthesis for Hanoi Vietnamese”, IEICE Transactions on Information and Systems, Vol. E98-D, No. 12, 2015.

## USING AUTOCORRELATION ALGORITHM COMBINING MEDIAN SMOOTHING FOR DETECTING FUNDAMENTAL FREQUENCY OF SPEECH SIGNALS IN MATLAB

## Nguyễn Trần Hậu, Đặng Xuân Lộc, Nguyễn Thái Minh, Nguyễn Hữu Hoàng Hưng

***ABSTRACT*** *- Detecting the fundamental frequency of signal is a common issue in audio signal processing, especially speech processing. This report implements pitch detection algorithm for speech signals using autocorrelation function on Matlab. In order to smooth the fundamental frequency found, median smoothing is used. Experiments on 5 vowel speech signals (/a/, /e/, /i/, /o/ and /u/) show that ……*

*Practical results also exhibit that ……*

*- Tiếng Anh: fundamental frequency detection, pitch period estimation, autocorrelation function, median function.*