**PHÂN ĐOẠN TÍN HIỆU TIẾNG NÓI VÀ KHOẢNG LẶNG DÙNG HÀM TỰ TƯƠNG QUAN, HÀM VI SAI BIÊN ĐỘ TRUNG BÌNH VÀ PHÂN TÍCH PHỔ DÙNG BIẾN ĐỔI FOURIER NHANH**

**Phùng Đình Dương, Nguyễn Đức Chinh, Nguyễn Đặng Tuấn Kiệt, Mai Xuân Nhật**

**Nhóm 10, lớp HP: 19.99**

|  |  |  |  |
| --- | --- | --- | --- |
| **Điểm** | **Bảng phân công nhiệm vụ** | | **Chữ ký của SV** |
|  | Phùng Đình Dương  (nhóm trưởng) | Đọc tài liệu, viết code Matlab hàm vi sai biên độ trung bình (AMDF), hỗ trợ làm slide PowerPoint, chịu trách nhiệm phân công và theo dõi tiến độ của nhóm. |  |
|  | Nguyễn Đức Chinh | Đọc tài liệu, viết code Matlab thuật toán lọc trung vị (median filtering), xuất hình ảnh và nhận xét kết quả F0 thu được từ hàm AMDF, hỗ trợ làm slide PowerPoint. |  |
|  | Nguyễn Đặng Tuấn Kiệt | Đọc tài liệu, viết code Matlab hàm tự tương quan (AutoCorrelation), xuất hình ảnh và nhận xét kết quả F0 thu được từ hàm tự tương quan, viết báo cáo hoàn chỉnh. |  |
|  | Mai Xuân Nhật | Đọc tài liệu, viết code Matlab thuật toán tự động tính F0 dùng phép biến đổi Fourier nhanh, xuất hình ảnh và nhận xét kết quả F0 thu được, hỗ trợ viết 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 liệt kê ở cuối báo cáo. 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— Bài thực hành này thực hiện việc tính toán và xuất ra đồ thị tần số cơ bản F0 của file audio tín hiệu tiếng nói bằng cách sử dụng 3 thuật toán khác nhau (Hàm tự tương quan, hàm vi sai biên độ trung bình, thuật toán biến đổi Fourier nhanh). Sau đó, sử dụng thuật toán lọc trung vị để làm trơn kết quả tần số F0 thu được từ 3 thuật toán trên. Từ đó, tiến hành so sánh và rút ra nhận xét, kết luận về dữ liệu tần số cơ bản F0 của tiếng nói thu được từ 3 thuật toán trên. Bài thực hành này cũng thực hiện việc đánh giá ưu, nhược điểm của từng thuật toán cũng như cách khắc phục các nhược điểm, cải thiện thuật toán. Kết quả thực nghiệm cũng cho thấy ta hoàn toàn có thể xác định được tần số cơ bản F0 của tín hiệu tiếng nói trên cả miền thời gian và tần số.

Từ khóa— Hàm tự tương quan, hàm vi sai biên độ trung bình, biến đổi Fourier nhanh, lọc trung vị.

Mục lục

[I. ĐẶT VẤN ĐỀ 4](#_Toc57930881)

[II. LÝ THUYẾT XỬ LÝ TÍN HIỆU TIẾNG NÓI VÀ CÁC THUẬT TOÁN 4](#_Toc57930882)

[A. Các lý thuyết chung 4](#_Toc57930883)

[*1.* Sơ đồ thuật toán chung 4](#_Toc57930884)

[*2.* Các tham số quan trọng 5](#_Toc57930885)

[*3.* Vấn đề chung và giải pháp khắc phục 5](#_Toc57930886)

[B. Hàm tự tương quan 5](#_Toc57930887)

[*1.* Cơ sở lý thuyết 5](#_Toc57930888)

[*2.* Sơ đồ khối thuật toán 6](#_Toc57930889)

[*3.* Các tham số quan trọng của thuật toán 6](#_Toc57930890)

[C. Hàm vi sai biên độ trung bình 6](#_Toc57930891)

[*1.* Cơ sở lý thuyết 6](#_Toc57930892)

[*2.* Sơ đồ thuật toán 7](#_Toc57930893)

[*3.* Các tham số quan trọng của thuật toán 8](#_Toc57930894)

[D. Biến đổi Fourier nhanh 8](#_Toc57930895)

[*1.* Cơ sở lý thuyết 8](#_Toc57930896)

[*2.* Sơ đồ thuật toán 9](#_Toc57930897)

[*3.* Các tham số quan trọng của thuật toán 9](#_Toc57930898)

[E. Lọc trung vị 10](#_Toc57930899)

[*1.* Cơ sở lý thuyết 10](#_Toc57930900)

[*2.* Sơ đồ thuật toán 10](#_Toc57930901)

[*3.* Các tham số quan trọng 11](#_Toc57930902)

[F. Hàm cửa sổ 11](#_Toc57930903)

[*1.* Cơ sở lý thuyết 11](#_Toc57930904)

[*2.* Các tham số quan trọng của thuật toán 12](#_Toc57930905)

[III. MÃ CHƯƠNG TRÌNH CÀI ĐẶT CÁC THUẬT TOÁN 12](#_Toc57930906)

[A. Hàm cửa sổ Hamming 12](#_Toc57930907)

[B. Hàm tự tương quan (AutoCorrelation) 12](#_Toc57930908)

[C. Hàm vi sai biên độ trung bình tín hiệu (AMDF) 13](#_Toc57930909)

[D. Phân tích phổ của tín hiệu sử dụng biến đổi Fourier nhanh 14](#_Toc57930910)

[E. Hàm lọc trung vị (Median Filtering) 15](#_Toc57930911)

[F. Chương trình chính demo (main.m) 15](#_Toc57930912)

[IV. KẾT QUẢ THỰC NGHIỆM 17](#_Toc57930913)

[A. Tín hiệu mẫu thử nghiệm 17](#_Toc57930914)

[B. Kết quả tần số cơ bản của hàm tự tương quan trên 4 tín hiệu mẫu 17](#_Toc57930915)

[C. Kết quả tần số cơ bản của hàm vi sai biên độ trung bình trên 4 tín hiệu mẫu 18](#_Toc57930916)

[D. Kết quả tần số cơ bản của hàm phân tích phổ tín hiệu sử dụng biến đổi Fourier nhanh trên 4 tín hiệu mẫu 18](#_Toc57930917)

[E. Hàm tự tương quan của 1 đoạn tín hiệu âm hữu thanh và 1 đoạn tín hiệu âm vô thanh dài 20ms 19](#_Toc57930918)

[F. Hàm vi sai biên độ trung bình của 1 đoạn tín hiệu âm hữu thanh và 1 đoạn tín hiệu âm vô thanh dài 20ms 19](#_Toc57930919)

[G. Phân tích phổ tín hiệu dùng FFT của 1 đoạn tín hiệu âm hữu thanh và 1 đoạn tín hiệu âm vô thanh dài 20ms 20](#_Toc57930920)

[H. Khảo sát ảnh hưởng của độ dài của bộ lọc trong công thức lọc trung vị tới kết quả F0 thu được 20](#_Toc57930921)

[I. Khảo sát ảnh hưởng của các FFT points khác nhau tới kết quả F0 thu được 21](#_Toc57930922)

[V. KẾT LUẬN 21](#_Toc57930923)

[A. Hàm tự tương quan (AutoCorrelation) 21](#_Toc57930924)

[B. Hàm vi sai biên độ trung bình (AMDF) 21](#_Toc57930925)

[C. Phân tích phổ tín hiệu dùng biến đổi Fourier nhanh (FFT) 21](#_Toc57930926)

[D. Lọc trung vị 21](#_Toc57930927)

[VI. TÀI LIỆU THAM KHẢO 22](#_Toc57930928)

# 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, việc xử lý tiếng nói của con người ngày càng trở nên cần thiết. Xử lý tiếng nói có ứng dụng về nhiều mặt, chẳng hạn như nhận dạng tiếng nói, người nói, tăng chất lượng giọng nói và tổng hợp tiếng nói. Để làm được điều đó, việc xác định tần số cơ bản là điều tiên quyết, là cơ sở cho mọi đột phá sau này. Có nhiều phương pháp khác nhau để xác định được tần số cơ bản F0 của tiếng nói như AMDF, LPC, xử lý đồng hình, tự tương quan, phép biến đổi Fourier nhanh, …

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 mà 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 thực hành này, chúng em tiến hành tìm hiểu các thuật toán tìm tần số cơ bản F0 của tín hiệu tiếng nói, ý nghĩa của các đại lượng có trong công thức và các tham số quan trọng của từng thuật toán. Sau đó, cài đặt các thuật toán đó trên Matlab để tính toán và xuất kết quả F0 thu được, dưới dạng đồ thị, của mỗi file tín hiệu (có thể được thu được từ cả phòng thí nghiệm lẫn studio). Từ đó, quan sát số liệu và đồ thị để rút ra nhận xét, kết luận, đánh giá về ưu, nhược điểm, độ chính xác, độ tối ưu, tốc độ,… của cả 3 thuật toán tính tần số cơ bản này.

Bài báo cáo có bố cục như sau:

Phần I trình bày chủ đề chung của bài thực hành và các vấn đề cần giải quyết.

Phần II trình bày về cơ sở lý thuyết của các thuật toán (Hàm tự tương quan, hàm vi sai biên độ trung bình, thuật toán biến đổi Fourier nhanh), các tham số quan trọng của thuật toán, các vấn đề và ưu, nhược điểm của từng thuật toán, cũng như đề ra các giải pháp, cách khắc phục.

Phần III trình bày mã nguồn cài đặt các thuật toán. Phần IV trình bày kết quả thực nghiệm mô tả dữ liệu dùng để đánh giá độ chính xác và tối ưu của từng thuật toán, đưa ra các đánh giá định tính và định lượng.

Phần V trình bày kết luận và các nhận xét rút ra được từ bài thực hành.

# LÝ THUYẾT XỬ LÝ TÍN HIỆU TIẾNG NÓI VÀ CÁC THUẬT TOÁN

## Các lý thuyết chung

### Sơ đồ thuật toán chung

Có nhiều thuật toán để tìm tần số cơ bản F0 của tín hiệu tiếng nói chẳng hạn như hàm tự tương quan, hàm vi sai biên độ trung bình, biến đổi Fourier nhanh,… Những hàm này đều có đặc tính chung là đều có các đỉnh cực đại/ cực tiểu tuần hoàn với chu kỳ xấp xỉ bằng với chu kỳ cơ bản của tín hiệu đầu vào. Hơn thế nữa, chúng còn có thể được sử dụng để phân biệt âm vô thanh (nhiễu nền hoặc khoảng lặng) với âm hữu thanh (tiếng nói) nhờ những điểm khác biệt rõ rệt của đồ thị thu được sau khi áp dụng lên 2 loại âm này. Nhờ vậy mà các thuật toán đó được ứng dụng rất rộng rãi trong việc xác định tần số cơ bản của tín hiệu.

Ta có thể lập một sơ đồ thuật toán chung nhất cho các phương pháp tìm tần số cơ bản trên như sau:



1. Sơ đồ thuật toán chung cho các hàm tự tương quan, AMDF, phổ biên độ

### Các tham số quan trọng

#### Độ dài khung tín hiệu:

Thuật toán tính F0 dựa trên hàm tự tương quan có sử dụng kỹ thuật xử lý ngắn hạn (phân tín hiệu thành nhiều khung nhỏ) nên việc chọn loại cửa sổ và độ dài cửa sổ thích hợp là quan trọng. Tín hiệu tiếng nói có một tính chất quan trọng là các đặc tính của nó thay đổi tương đối chậm theo thời gian. Thông thường, các đặc tính của tín hiệu ổn định trong khoảng thời gian từ 10 ms đến 30 ms. Do đó, người ta thường chia tín hiệu cần xử lý thành các khung tín hiệu liên tiếp nhau, mỗi khung có độ dài từ 10 ms đến 30 ms [1]. Trong bài thực hành này, chúng em chọn 20ms / 1 frame.

#### Ngưỡng xác định hữu thanh / vô thanh:

Để phân biệt giữa âm vô thanh và âm hữu thanh, chúng ta dựa vào đặc trưng biên độ của tín hiệu. Nếu cực đại cục bộ của khung đang xét nhỏ hơn ngưỡng (ngưỡng lấy khoảng 0.03, dùng để loại nhiễu) thì khung tín hiệu đang xét là âm vô thanh và có tần số cơ bản không xác định được. Ngược lại thì là âm hữu thanh, có tần số xác định [1].

#### Công thức tính tần số cơ bản F0:

Công thức tính tần số cơ bản của tín hiệu: F0 = Fs / Lag

Trong đó: F0 là tần số cơ bản (Hz), Fs là tần số lấy mẫu (Hz) và Lag là độ trễ (mẫu) 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 Lag.

Do tín hiệu ban đầu được chia nhỏ thành từng khung qua phép lấy cửa sổ, đồng thời do tín hiệu tiếng nói trong thực tế chưa tuần hoàn một cách hoàn hảo, nên ứng với mỗi khung tín hiệu sẽ có 1 giá trị F0, các giá trị này xấp xỉ nhau.

Riêng đôi với phương pháp phân tích phổ biên độ tín hiệu dùng hàm fft(), độ trễ tại đỉnh cực đại gần gốc nhất cũng chính là tần số cơ bản F0 cần tìm nên không cần áp dụng công thức này.

### Vấn đề chung và giải pháp khắc phục

#### Số lượng đỉnh cực đại:

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 tự tương quan càng về sau sẽ càng giảm dần. Điều này khiến cho việc xác định cực đại sẽ khó khăn, do đó ta chỉ chọn ra 1 đỉnh có biên độ lơn hơn hoặc bé hơn hẳn so với các đỉnh còn lại với độ trễ , qua đó xác định tần số cơ bản, công việc xử lý và tính toán cũng nhờ vậy mà trở nên dễ dàng hơn.

#### Kết quả F0 thu được:

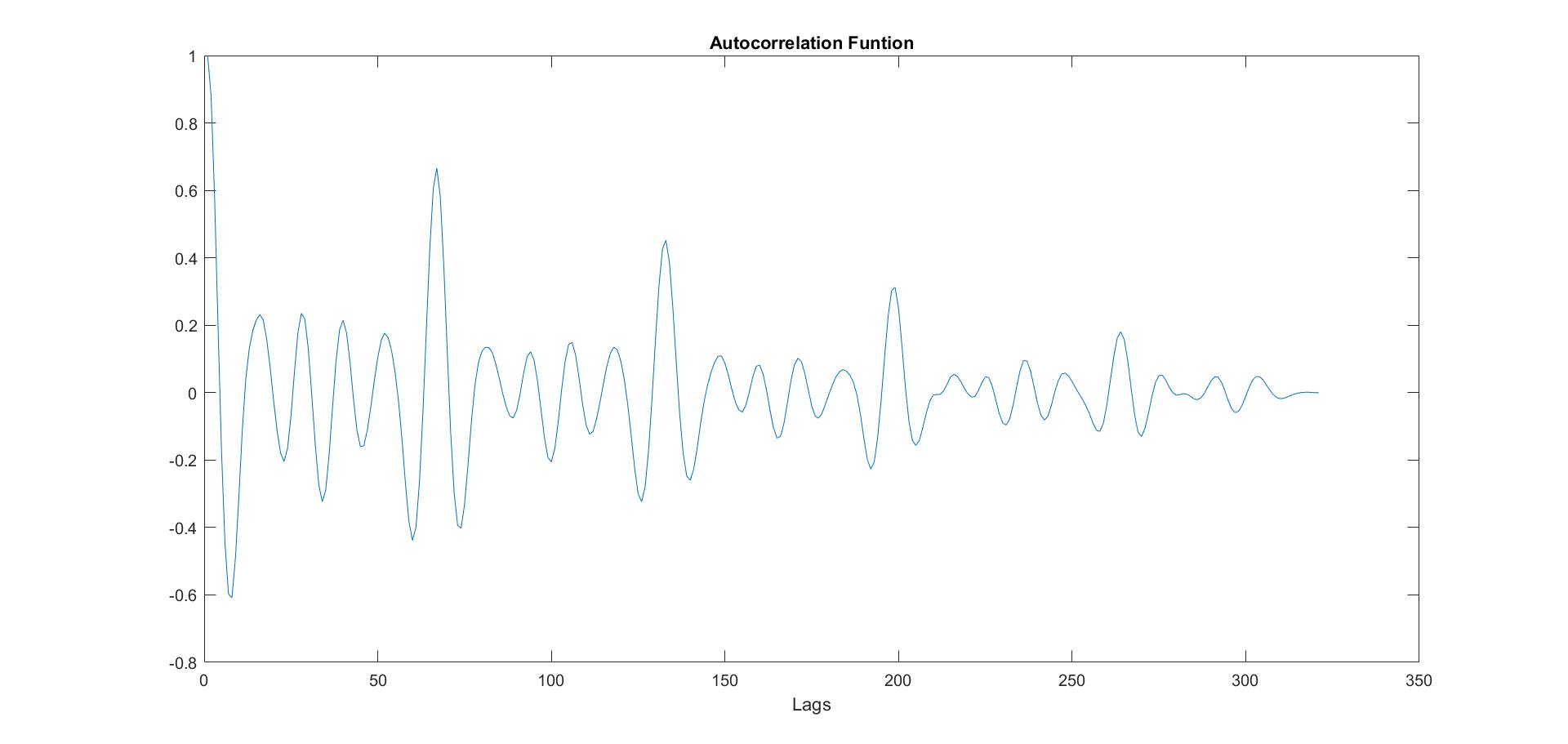
Vì tín hiệu được chia thành rất nhiều khung khác nhau nên cũng sẽ có rất nhiều kết quả F0 khác nhau thu được trên cùng 1 tín hiệu. Các kết quả này hầu như đều xấp xỉ nhau (khoảng từ 75Hz đến 350Hz). Tuy nhiên, trong thực tế vẫn sẽ có một số khung cho ra kết quả F0 chênh lệch khá nhiều so với những kết quả còn lại. Để làm trơn kết quả F0 thu được thông qua các thuật toán, ta sử dụng phương pháp lọc trung vị (Median Filtering) sẽ được trình bày ở phần sau.

## Hàm tự tương quan

### Cơ sở lý thuyết

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. Hàm tự tương quan đo sự phụ thuộc của tín hiệu x[n] và phiên bản dịch thời gian k (mẫu) của nó. Với tín hiệu tuần hoàn, ta có định nghĩa hàm tự tương quan:

Trong đó, rxx(k) là giá trị hàm tự tương quan tại thời điểm dịch k mẫu, N là độ dài của tín hiệu, xn là giá trị biên độ của tín hiệu tại thời điểm n và xn-k là giá trị biên độ của tín hiệu tại thời điểm trước đó k mẫu, với k chạy từ đến , N\* = N-k.

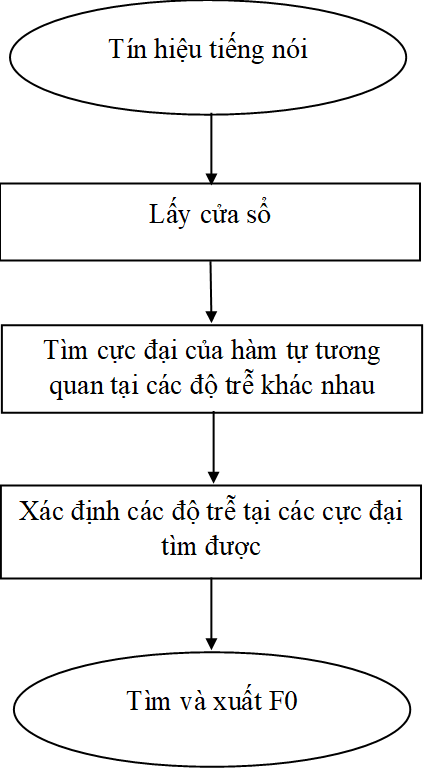


1. Đồ thị hàm tự tương quan của 1 đoạn tín hiệu tiếng nói ngắn (0.02s) từ file lab\_female.wav

Tín hiệu của âm hữu thanh có dạng sóng gần như tuần hoàn nên hàm tự tương quan của nó sẽ xuất hiện các điểm cực đại cục bộ tại các độ trễ có giá trị bằng bội số nguyên lần của chu kỳ cơ bản. Như vậy, từ đồ thị hàm tự tương quan, ta sẽ xác định các cực đại và độ trễ tương ứng tại các cực đại và từ độ trễ ta sẽ xác định được tần số cơ bản của tín hiệu. Ngược lại, tín hiệu của âm vô thanh (nhiễu nền hoặc khoảng lặng) có dạng sóng không tuần hoàn nên hàm tự tương quan của nó sẽ không có tính chất tương tự như âm hữu thanh [1]. Vậy, giá trị cao hay thấp của điểm cực đại cục bộ có biên độ lớn nhất của hàm tự tương quan có thể dùng để phân biệt một khung tín hiệu là hữu thanh hay vô thanh.

### Sơ đồ khối thuật toán

Vậy để áp dụng hàm tự tương quan vào công việc tìm tần số cơ bản F0 của tiếng nói, sau khi tìm đọc các tài liệu liên quan, nhóm chúng em đề xuất thuật toán như trong sơ đồ sau:



1. Sơ đồ thuật toán tính tần số cơ bản của tín hiệu bằng hàm tự tương quan

Bằng kỹ thuật xử lý ngắn hạn, tín hiệu tiếng nói đầu vào được chia nhỏ thành các khung tín hiệu ngắn (có độ dài từ 10 ms đến 30 ms) để xử lý. Trong bài tập nhóm này, chúng em sử dụng hàm cửa sổ Hamming, được xác định bởi công thức [1]:

h(n) =

### Các tham số quan trọng của thuật toán

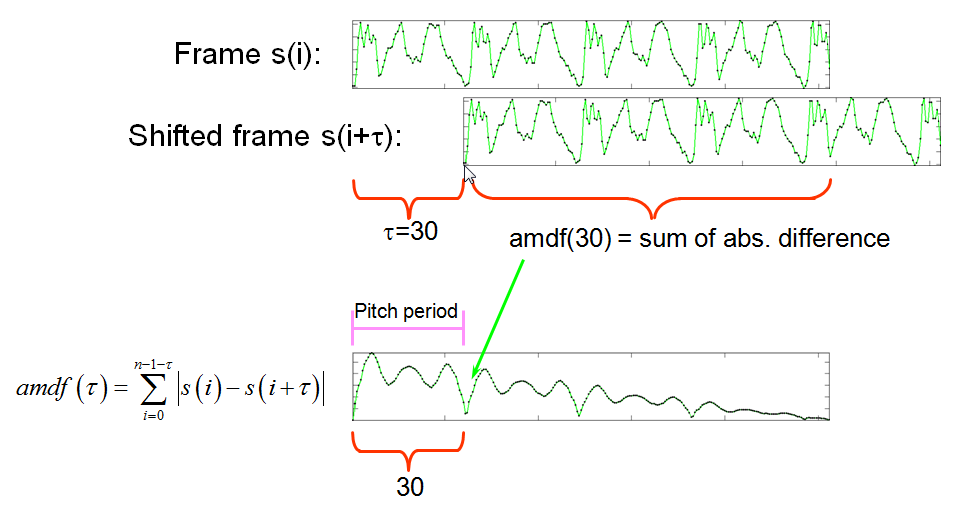
* Chúng em đã cài đặt chương trình Matlab cho hàm tự tương quan dựa trên công thức trong phần cơ sở lý thuyết. Hàm sẽ được gọi với cú pháp AutoCorr(x, Fs), với x là tín hiệu đầu vào và Fs là tần số lấy mẫu của tín hiệu đó. Vào trong hàm, tín hiệu sẽ được chia khung và các khung này gối đầu (overlap) lên nhau. Sau đó, xác định các đỉnh cực đại có biên độ lớn hơn ngưỡng (khoảng 30% cực đại cục bộ) trong phạm vi miền tần số giới hạn để xác định âm vô thanh / hữu thanh và độ trễ của chúng (nếu có). Cuối cùng tính toán và lưu lại các giá trị F0. Kết quả trả về của hàm này là 1 mảng một chiều của các giá trị F0 tính toán được từ công thức trên tất cả các khung (F0 = -inf đối với những khung của tín hiệu âm vô thanh).
* Các tham số khác của thuật toán như độ dài khung tín hiệu, ngưỡng xác định âm vô thanh / hữu thanh, công tính tính tần số cơ bản F0,… đều đã được đề cập trong phần lý thuyết chung.

## Hàm vi sai biên độ trung bình

### Cơ sở lý thuyết

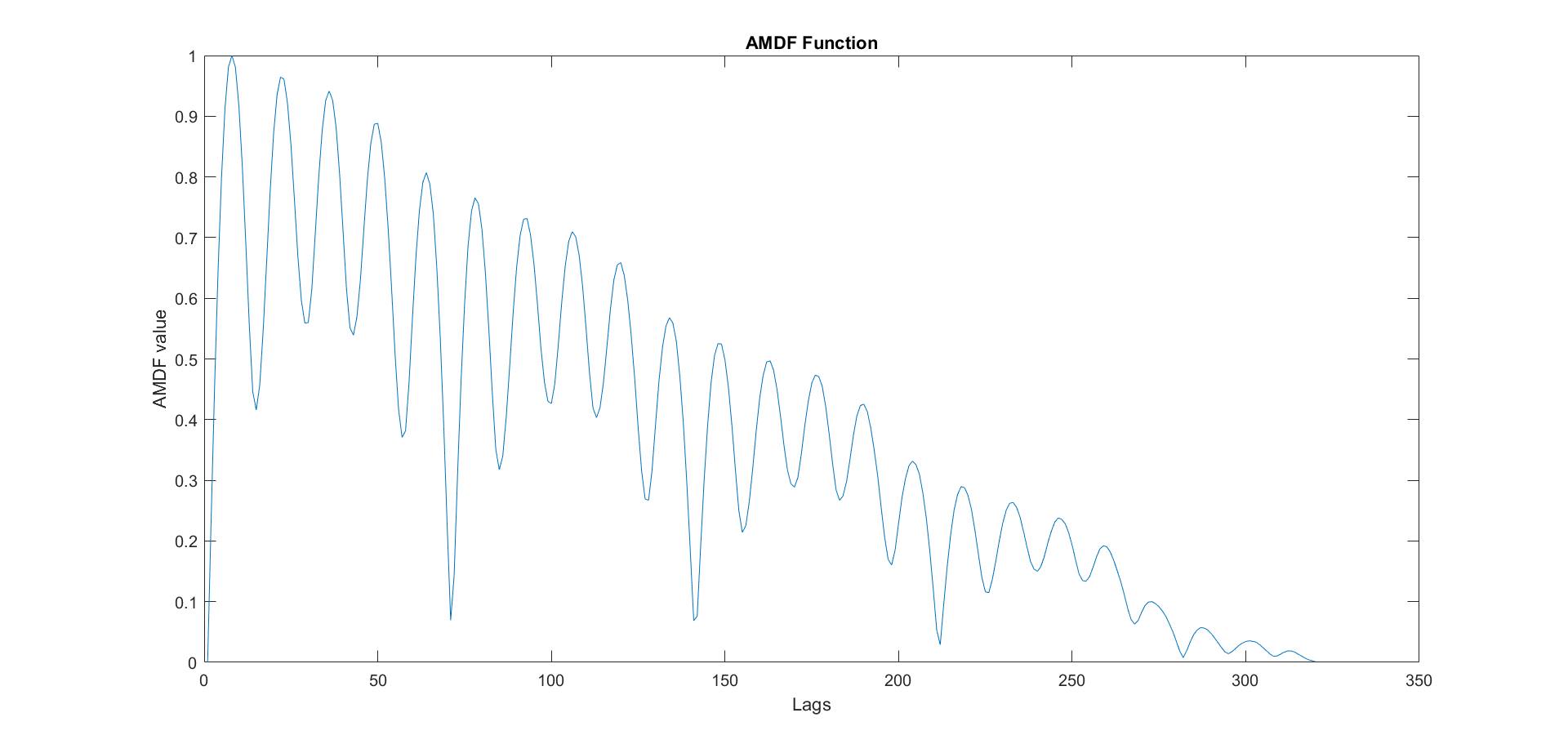
Khái niệm AMDF (hàm vi sai biên độ trung bình) rất gần với khái niệm hàm AutoCorrelation (hàm tự tương quan) ngoại trừ việc nó ước tính khoảng cách thay vì độ tương quan giữa khung s(i) (i = ) và phiên bản trễ của nó thông qua công thức sau [2]:

Trong đó  là độ trễ thời gian tính theo đơn vị mẫu và n là độ dài khung tín hiệu (mẫu). Giá trị của  mà tại đó amdf() đạt giá trị cực tiểu nhỏ nhất cũng chính là bội chu kỳ cao độ (pitch period) của tín hiệu mẫu. Hình sau minh họa hoạt động của hàm AMDF [2]:

[](http://mirlab.org/jang/books/audiosignalprocessing/image/amdf.png)

1. Minh họa cách thức hoạt động của hàm AMDF

Nói cách khác, chúng ta dịch chuyển phiên bản bị trễ n lần theo trục thời gian và tính tổng tuyệt đối các vi sai (hay độ chênh lệch) biên độ của các đoạn gối đầu lên nhau để thu được n giá trị của hàm AMDF, sau đó tiến hành tìm các giá trị mà tại đó amdf( đạt cực tiểu nhỏ nhất và áp dụng công thức để tính tần số cơ bản F0 của khung tín hiệu [2].



1. Đồ thị hàm vi sai biên độ trung bình của 1 đoạn tín hiệu tiếng nói ngắn (0.02s) từ file lab\_female.wav

### Sơ đồ thuật toán

Thuật toán tìm tần số cơ bản bằng hàm vi sai biên độ trung bình được thể hiện ở sơ đồ dưới đây:



1. Sơ đồ khối thuật toán tính tần số cơ bản bằng hàm vi sai biên độ trung bình

### Các tham số quan trọng của thuật toán

* Để tìm cần số cơ bản của tín hiệu dùng hàm vi sai biên độ trung bình, chúng em đã cài đặt 1 hàm AMDF bằng Matlab có thể được gọi bằng cú pháp AMDF(x, Fs), với x là tín hiệu đầu vào và Fs là tần số lấy mẫu của tín hiệu. Tương tự như hàm tự tương quan, tín hiệu cũng sẽ được chia thành nhiều khung và các khung này gối đầu (overlap) lên nhau. Sau đó, xác định các đỉnh cực tiểu trong phạm vi miền tần số giới hạn để nhận biết âm vô thanh / hữu thanh và lưu lại độ trễ của chúng (nếu có). Cuối cùng tính toán các giá trị F0. Kết quả trả về của hàm là 1 mảng 1 chiều các giá trị F0 thu được của từng khung tín hiệu (F0 = NaN đối với những khung của tín hiệu âm vô thanh).
* Các tham số khác của thuật toán như độ dài khung tín hiệu, ngưỡng xác định âm vô thanh / hữu thanh, công tính tính tần số cơ bản F0,… đều đã được đề cập trong phần lý thuyết chung.

## Biến đổi Fourier nhanh

### Cơ sở lý thuyết

#### 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 của một tín hiệu. Biến đổi này được tính nhanh bởi thuật toán biến đổi Fourier nhanh (FFT) [3].

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 cho N điểm:

Trong đó, X[k] là 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 của x[n]. Phép biến đổi Fourier rời rạc có một số tính chất như tính tuần hoàn, tính tuyến tính, tính dịch thời gian, tính dịch tần số, tính chập thời gian và tính nhân thời gian [3]:

+ Tính tuần hoàn: X[k] tuần hoàn với chu kỳ N, nghĩa là X[k+N] = X[k] ∀k [10].

+ Tính tuyến tính:

+ Tính dịch thời gian:

+ Tính dịch tần số:

+ Tính chập thời gian:

+ Tính nhân thời gian:

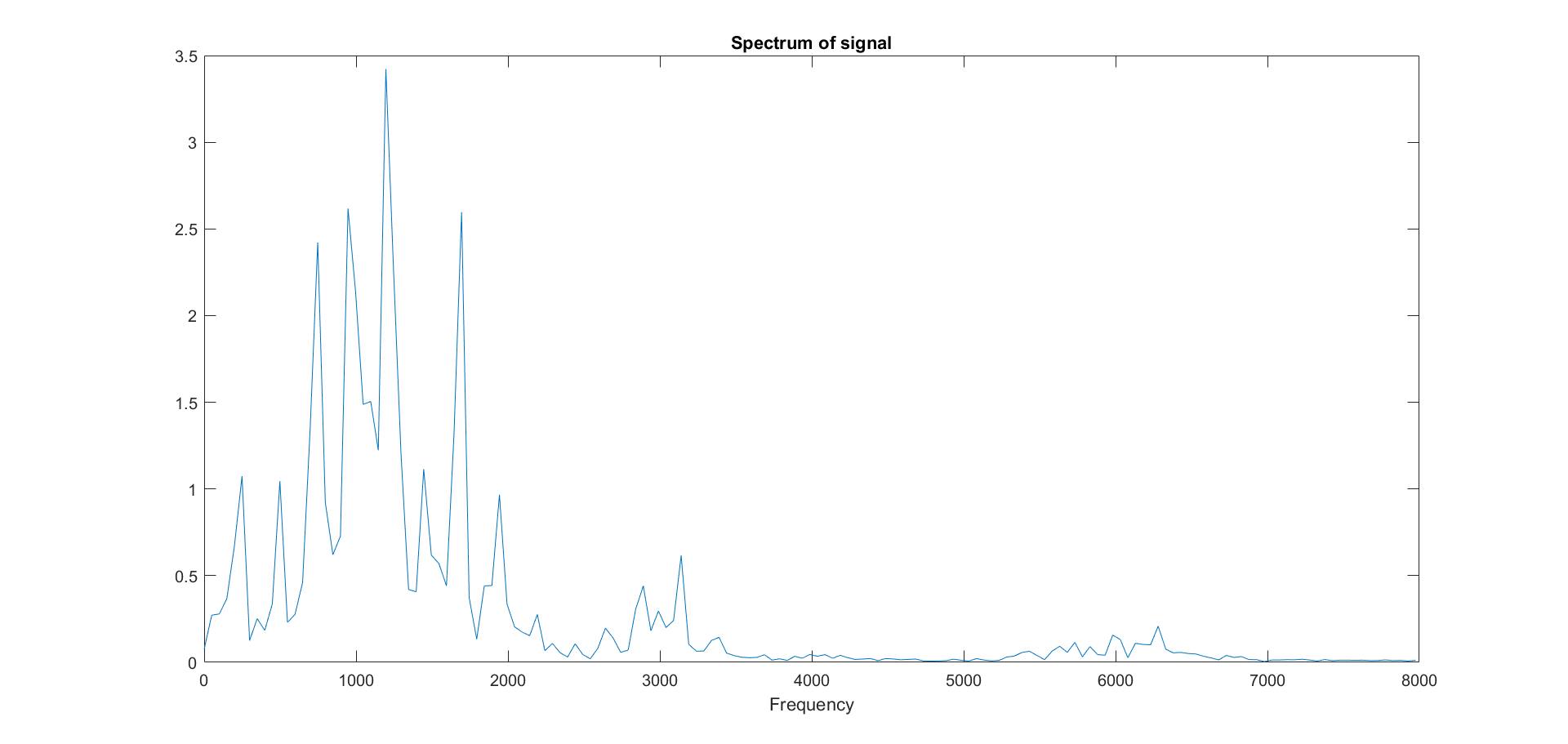
#### Phép biến đổi Fourier nhanh (Fast Fourier Transform – FFT)

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 [4]. Thuật toán FFT tính nhanh DFT N-điểm của tín hiệu rời rạc x[n] với N = 2m ≥ length(x[n]). Vì thuật toán FFT chỉ áp dụng cho trường hợp N = 2m nên công thức tổng của DFT có thể phân tích thành hai tổng như sau:

Trong đó, Xe[k] và Xo[k] lần lượt là phép 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.

Bằng thuật toán FFT, cần log2N phép nhân phức thay cho (N−1)2 phép nhân phức và Nlog2N phép cộng phức thay vì N(N − 1). Vậy, tính trực tiếp từ định nghĩa DFT (\*) đòi hỏi O(N2) phép tính, trong khi đó FFT sẽ giúp tính cùng một kết quả nhưng chỉ trong O(N\*logN) phép tính [4].

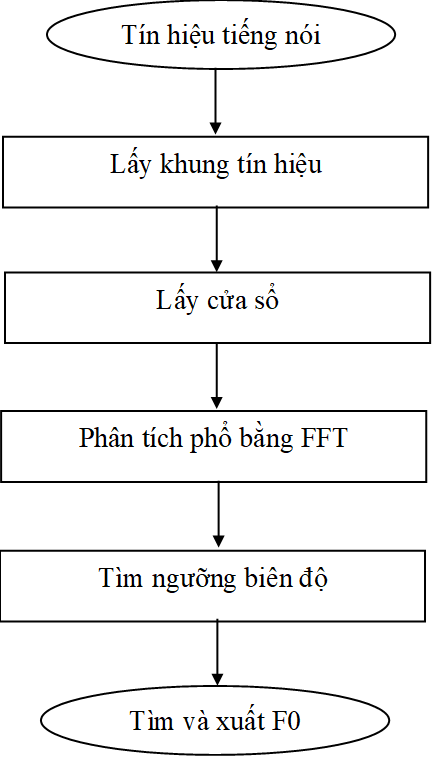
Trong bài thực hành này, chúng em sử dụng hàm fft() của thư viện Matlab để phân tích phổ của tín hiệu tiếng nói. Sau đó cũng tiến hành tìm ngưỡng biên độ để phân biệt âm vô thanh và âm hữu thanh. Cuối cùng tìm các đỉnh cực đại có biên độ lớn hơn hoặc bằng ngưỡng mà tại đó độ trễ (lag) của chúng đúng bằng với giá trị tần số cơ bản F0.



1. Phân tích phổ của tín hiệu sử dụng phép biến đổi Fourier nhanh (FFT)

### Sơ đồ thuật toán

Sơ đồ khối thuật toán tìm tần số cơ bản F0 trên miền tần số được trình bày trên Hình 5.



1. Sơ đồ khối thuật toán tính tần số cơ bản F0 bằng phép biến đổi Fourier nhanh

### 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 Fourier (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 biểu diễn phổ (spectrum) của x[n].

* Các tham số khác của thuật toán như độ dài khung tín hiệu, ngưỡng xác định âm vô thanh / hữu thanh đều tương tự như của hàm tự tương quan đã được đề cập trước.
* Bằng phép biến đổi Fourier nhanh, độ trễ các cực đại có biên độ lớn hơn ngưỡng trong đồ thị phổ của tín hiệu tìm được sẽ đúng bằng với tần số cơ bản F0 của tín hiệu. Vậy ta chỉ cần xác định cực đại thỏa mãn và lưu lại độ trễ tương ứng.

## Lọc trung vị

### Cơ sở lý thuyết

Trong hầu hết các trường hợp, một bộ lọc tuyến tính nói chung có chức năng loại bỏ các thành phần tạp âm ra khỏi tín hiệu. Tuy nhiên, vẫn có một số trường hợp, các bộ lọc tuyến tính không hoàn toàn phù hợp do loại tín hiệu cần được làm trơn. 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 [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 những tính chất mong muốn [5].

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]).

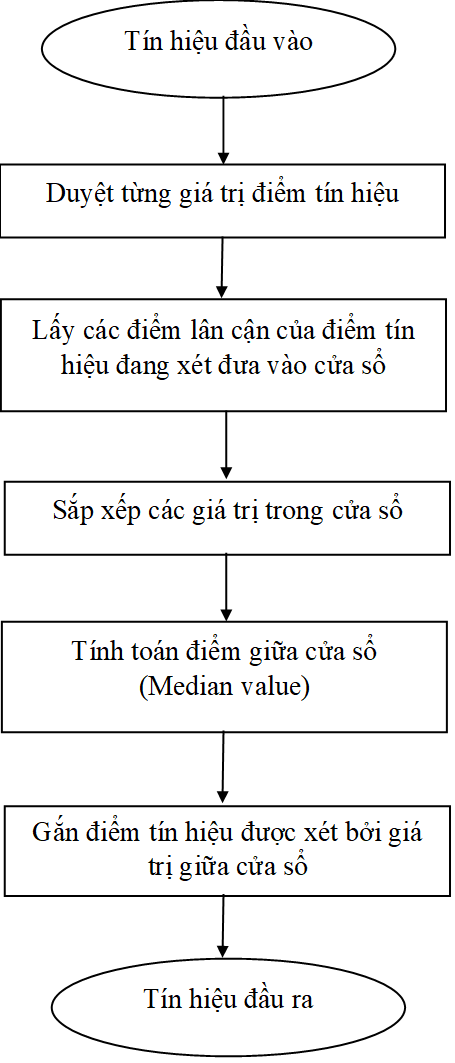
Số trung vị là số nằm chính giữa trong một tập hợp dãy số (n phần tử) đã được sắp xếp theo thứ tự tăng dần [5].

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 [5].

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) [5].

### Sơ đồ thuật toán

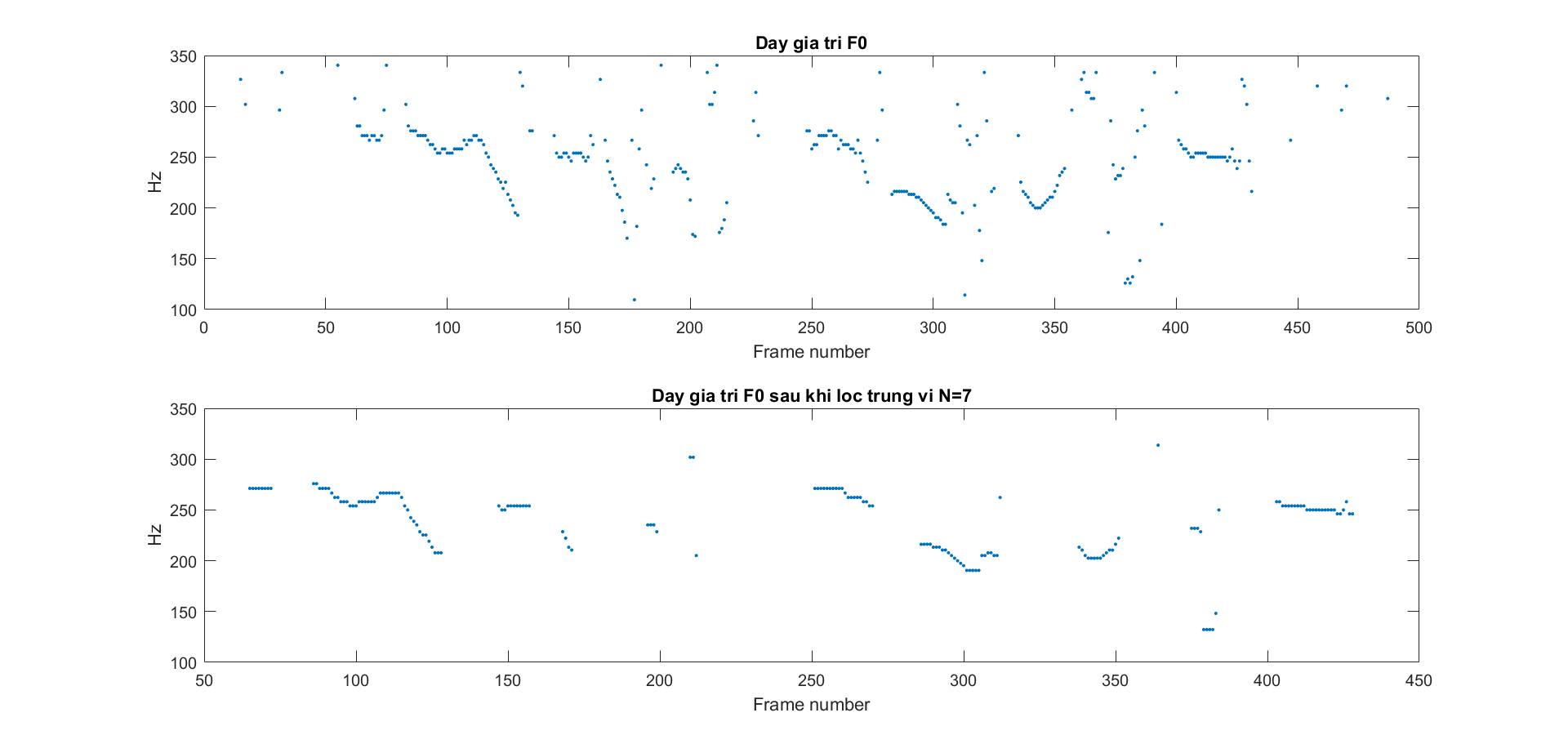
Sơ đồ thuật toán lọc trung vị (Median Filtering) để làm trơn kết quả F0 thu được từ các thuật toán tìm tần số cơ bản của tiếng nói được trình bày trên hình 6.



1. Sơ đồ thuật toán lọc trung vị

### Các tham số quan trọng

* Trong matlab, để lọc trung vị của kết quả F0 thu được, chúng em tiến hành cài đặt hàm MedFiltering để lọc trung vị các tín hiệu một chiều. Cú pháp của hàm là MedFiltering(x,N). Với x là mảng một chiều các giá trị F0 tính được bằng hàm tự tương quan, hàm vi sai biên độ trung bình và biến đổi Fourier nhanh; 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.



1. Kết quả F0 của 1 thuật toán trước (hình trên) và sau (hình dưới) khi lọc trung vị

* Ta 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.

## Hàm cửa sổ

### Cơ sở lý thuyết

#### Chức năng của hàm cửa sổ

Chức năng của hàm cửa sổ là hàm toán học có dạng hình sin. Về mặt toán học, khi một hàm khác dạng sóng hoặc chuỗi dữ liệu được "nhân" bởi hàm cửa sổ, sẽ có giá trị bằng chính nó trong phạm vi thời gian của hàm cửa sổ, bằng 0 nếu nằm ngoài khoảng thời gian được chọn của hàm cửa sổ.

Các chức năng khác của hàm cửa sổ được sử dụng trong phân tích quang phổ, các bộ lọc đáp ứng xung hữu hạn,  
cũng như thiết kế dầm và ăng-ten [6].

#### Hàm cửa sổ hình chữ nhật

Cửa sổ hình chữ nhật (thỉnh thoảng được gọi là cửa sổ Dirichlet) là cửa sổ đơn giản nhất trong các loại cửa sổ, thay thế tất cả các giá trị N của chuỗi dữ liệu bằng 0 (với N là chiều dài mẫu của một hàm thời gian rời rạc), làm cho nó xuất hiện như thể dạng sóng với w(n) =1. Một cửa sổ hình chữ nhật không thay đổi phân đoạn tín hiệu. Nó làm cho tín hiệu trong khung tín hiệu trong hàm cửa sổ giữ nguyên, bằng không khi nằm ngoài khung tín hiệu được chọn [6].

Hàm cửa sổ chữ nhật: h(n) =

#### Hàm cửa sổ Hamming

Hàm cửa sổ Hamming là hàm cửa sổ đặc biệt được sử dụng trong việc phân tích phổ để giảm hiện tượng rò phổ. Hàm cửa sổ Hamming với mức độ của hiện tượng rò phổ thấp sẽ cần thiết để phát hiện tín hiệu có vùng quang phổ rộng.

Hàm cửa sổ Hamming: h(n) =

Do mức độ của hiện tượng rò phổ, hàm cửa sổ hình chữ nhật thường không được sử dụng trong tín hiệu có quang phổ phạm vi tín hiệu rộng. Còn hàm cửa sổ Hamming có mức độ của hiện tượng rò phổ thấp nên thường được sử dụng trong các tín hiệu có phạm vi quang phổ rộng. Nhưng trong tín hiệu có phạm vi quang phổ hẹp, hàm cửa sổ hình nhật sẽ tối ưu hơn so với hàm cửa sổ Hamming vì nó không làm mất một số tín hiệu quan trọng [6].

### Các tham số quan trọng của thuật toán

* Dựa vào công thức của 2 hàm cửa sổ, chúng em đã cài đặt 2 đoạn code trong Matlab để tạo 2 hàm cửa sổ này.
* Hàm cửa sổ hình chữ nhật sẽ được gọi với cú pháp w = Rectwin(L) (Hàm chuẩn trong thư viện Matlab là rectwin()) trả về một cửa sổ hình chữ nhật có chiều dài L trong vectơ cột w, với L là số điểm để tạo cửa sổ hình chữ nhật.
* Hàm cửa sổ Hamming sẽ được gọi với cú pháp w = Hamming (L) (Hàm chuẩn trong thư viện Matlab là hamming()) trả về một cửa sổ Hamming với L là số điểm đối xứng.

# MÃ CHƯƠNG TRÌNH CÀI ĐẶT CÁC THUẬT TOÁN

## Hàm cửa sổ Hamming

function [w] = Hamming(L)

% The function is used for creating the Hamming window function of length L.

% Input is L - the length of window.

% The output is w, which is an array containing all calculated value of Hamming window function

for i = 0:L-1

w(i+1) = 0.54 - 0.46\*cos((2\*pi\*i)/L);

end

## Hàm tự tương quan (AutoCorrelation)

function [F0] = AutoCorr(signal,fs)

% This function is used for finding the basic frequency F0 of a signal with AutoCorrelation Function.

% signal is the input signal.

% fs is sampling frequency.

% The output is F0, which is an array containing all calculated basic frequencies of every frame.

% Limit the range of F0 is from 75Hz to 350Hz.

min\_f0 = 75;

max\_f0 = 350;

% Calculate the limit of period.

min\_period = fs/max\_f0;

max\_period = fs/min\_f0;

frame\_len = round(0.02\*fs); % Length of each frame is 20ms.

half = round(frame\_len/2); % For overlapping frames.

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

for k = 1 : (length(signal)/half - 1) % (length(signal)/half -1) = number of frames.

width = (k-1)\*half + 1 : (frame\_len + (k-1)\*half); % Beginning point to ending point of frame.

frame = h.\*signal(width); % Create frame.

N = length(frame); % N is the length of frame.

% Auto Correlation Function.

for lag = 0 : N-1

sum1 = 0;

for j = 0 : N-1-lag

sum1 = sum1 + frame(j+1)\*frame(j+lag+1);

end

rxx(lag+1) = sum1;

end

max\_rxx = rxx(1); % Local maximum of AutoCorrelation Func is at lag 0.

rxx(1:end) = rxx(1:end)/max\_rxx; % Standardlized rxx vector.

% Find all the local maxima of AutoCorrelation Function graph.

local\_maxima = zeros(1,N); % Create local maxima vector, which will contain all local maxima.

for i = 2:N-1

plotPoint=0;

if (rxx(i-1) < rxx(i) && rxx(i) > rxx(i+1)) % If rxx(i) is a local maxima.

plotPoint = rxx(i);

end

local\_maxima(i) = plotPoint; % Then store rxx(i) to local maxima vector.

end

all\_lags = 0:N-1; % Create lag vector

limited\_range = (all\_lags > min\_period) & (all\_lags < max\_period); % Boolean vector of limited range F0.

frame\_max = max(frame); % Global maximum of frame.

maxima = local\_maxima.\*limited\_range; % Remove all local maximas which out of limited range F0.

[maxima\_val, maxima\_lag] = max(maxima); % Find the greatest maxima in maxima vector.

if (frame\_max > 0.03) % Remove the noise by limiting the amplitude of signal in frame.

if (maxima\_val > 0.3) % If the greatest maxima's amplitude is greater than threshold = 0.3

F0(k) = fs/maxima\_lag; % Then current frame is voice, calculate the F0.

else

F0(k) = -inf; % Else current frame is noise, F0 is can't be determined.

end

else

F0(k) = -inf; % Else current frame is noise, F0 is can't be determined.

end

end

## Hàm vi sai biên độ trung bình tín hiệu (AMDF)

function [F0] = AMDF(signal,fs,min\_f0,max\_f0)

% This function is used for finding the basic frequency F0 of a signal with Average Magnitude Difference Function.

% signal is the input signal.

% fs is sampling frequency.

% min\_f0 and max\_f0 limit the range to find basic frequence F0 (min\_f0 <= F0 <= max\_f0).

% The output is F0, which is an array containing all calculated basic frequencies of every frame.

% Calculate the limit of period.

minperiod = fs/max\_f0;

maxperiod = fs/min\_f0;

frame\_len = round(0.02\*fs); % Length of each frame is 20ms.

half = round(frame\_len/2); % For overlapping frames.

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

for k = 1 : (length(signal)/half - 1) % (length(signal)/half -1) = number of frames.

width = (k-1)\*half + 1:(frame\_len + (k-1)\*half); % Beginning point to ending point of frame.

frame = h.\*signal(width); % Create frame.

N = length(frame); % N is the length of frame.

% Average Magnitude Difference Function.

for i = 0 : N-1

sum1 = 0;

for j = 0 : N-1-i

sum1 = sum1 + abs(frame(j+1)-frame(j+i+1));

end;

amdf(i+1) = sum1;

end

amdf\_max = max(amdf); % Local maximum of AMDF.

amdf = amdf/amdf\_max; % Standardlized amdf vector.

% Find all the local minima of AMDF graph.

local\_minima = zeros(1,N); % Create local minima vector, which will contain all local minima.

for i = 2:N-1

plotPoint=0;

if(amdf(i-1) > amdf(i) && amdf(i) < amdf(i+1)) % If amdf(i) is a local minima.

plotPoint=amdf(i);

end

local\_minima(i)=plotPoint; % Then store amdf(i) to local minima vector.

end

all\_lags = 0:N-1;% Create lag vector

limited\_range = (all\_lags > minperiod) & (all\_lags < maxperiod); % Boolean vector of limited range F0.

local\_minima = local\_minima.\*limited\_range; % Remove all local minima which out of limited range F0.

for i = 1:length(local\_minima)

if (local\_minima(i) == 0)

local\_minima(i) = 1;

end

end

[minima\_val, minima\_index] = min(local\_minima); % Find the smallest minima in minima vector.

frame\_max = max(frame); % Global maximum of frame.

if (frame\_max > 0.03) % Remove the noise by limiting the amplitude of signal in frame.

if (minima\_val < 0.7) % If the smallest minima's amplitude is smaller than threshold = 0.7

F0(k) = fs/minima\_index; % Then current frame is voice, calculate the F0.

else

F0(k) = -inf; % Else current frame is noise, F0 is can't be determined.

end

else

F0(k) = -inf; % Else current frame is noise, F0 is can't be determined.

end

end

## Phân tích phổ của tín hiệu sử dụng biến đổi Fourier nhanh

function [f0] = FFT(signal,fs,N)

% This function is used for finding the basic frequency F0 of a signal with spectral analysis using FFT

% signal is the input signal.

% fs is sampling frequency.

% N is FFT points.

% The output is F0, which is an array containing all calculated basic frequencies of every frame.

frame\_len = round(0.02\*fs); % Length of frame (20ms)

half = round(frame\_len/2); % For overlapping frame

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

for k = 1 : length(signal)/half -1 % (length(signal)/half -1) = number of frames.

width = (k-1)\*half + 1:(frame\_len + (k-1)\*half); % Beginning point to ending point of frame.

frame = h.\*signal(width); % Create frame.

% Frame's spectral analysis using fft() function of Matlab's library.

P2 = abs(fft(frame,N)); % 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); % Conserving energy of P1 spectrum.

freq = linspace(0,fs/2,length(P1)); % Create frequency vector.

local\_maxima = zeros(1,N); % Create local maxima vector, which will contain all local maxima.

for i = 2:length(P1)-1

plotPoint = 0;

if (P1(i-1) < P1(i) && P1(i) > P1(i+1)) % If P1(i) is a local maxima.

plotPoint = P1(i);

end

local\_maxima(i) = plotPoint; % Then store P1(i) to local maxima vector.

end

pos = []; % Position vector.

for i = 1:length(local\_maxima)

if (local\_maxima(i) > 2) % If the maxima's amplitude is greater than threshold = 2

pos = [pos i]; % Then store its position index to pos vector.

end

end

frame\_max = max(frame); % Global maximum of frame.

if (frame\_max > 0.03) % Remove the noise by limiting the amplitude of signal in frame.

if (length(pos) >= 2) % At least 2 local maxima in pos vector.

lag1 = freq(pos(1)); % Frequency 1 of local maxima 1st.

lag2 = freq(pos(2))-freq(pos(1)); % Frequency 2 of local maxima 2nd.

if (lag1 > 75) && (lag1 < 350) % If frequency 1 is in the range

if (lag2 > 75) && (lag2 < 350) % If frequency 2 is in the range

f0(k) = (lag1 + lag2)/2; % Calculate the average frequency.

else

f0(k) = lag1; % Or else return to frequency 1.

end

else

f0(k) = -inf; % Else current frame is noise, F0 is can't be determined.

end

else

f0(k) = -inf; % Else current frame is noise, F0 is can't be determined.

end

else

f0(k) = -inf; % Else current frame is noise, F0 is can't be determined.

end

end

## Hàm lọc trung vị (Median Filtering)

function [res] = MedFiltering(F0,N)

% This function is used for smoothing obtained F0 result.

% F0 is the basic frequency vector before smoothing.

% N is input's dememsion (level of F0 vector).

% The output is res, which is an array containing F0 vector after smoothing.

res = F0; % Result vector.

for k = 1:length(F0)

% Create an array containing current filtering value.

% The middle element of array is the value that need to be filtered.

if (k < ceil(N/2))

array = [zeros(1,ceil(N/2)-k) F0(1:N-ceil(N/2)+k)];

else

if (k > length(F0) - ceil(N/2) + 1)

array = [F0(k-ceil(N/2)+1:end) zeros(1,k-length(F0)+ceil(N/2)-1)];

else

array = [F0(k-ceil(N/2)+1:k+ceil(N/2)-1)];

end

end

% Sort the array - Bubble Sort.

for j = 0 : N-1

for i = 1: N-j-1

if (array(i) > array(i+1))

% Swap array(i) with array(i+1)

temp = array(i);

array(i) = array(i+1);

array(i+1) = temp;

end

end

end

res(k) = array(ceil(N/2)); % Replacing the value that need to be filtered with the array's middle element.

end

## Chương trình chính demo (main.m)

% This main function used to call all the defined functions to draw the needed graphs.

% Input is 4 audio wav file from TinHieuMau folder.

% Output are 3 graphs of 3 difference function and 3 graphs of MedFiltering function's result on each audio file.

% And 3 graphs of result F0 after smoothing.

[y,Fs] = audioread('lab\_female.wav'); % Fetch the signal.

% AutoCorrelation Function.

F0\_1 = AutoCorr(y,Fs);

subplot(3,2,1);

plot(F0\_1,'\*');

xlabel('Frame number');

ylabel('Hz');

title('Calculated F0 - AutoCorrelation Func');

% Average Magnitude Difference Function.

F0\_2 = AMDF(y,Fs,75,250);

subplot(3,2,3);

plot(F0\_2,'\*');

xlabel('Frame number');

ylabel('AMDF value');

title('Calculated F0 - AMDF');

% Signal spectral analysis using Fast Fourier Transform Function.

F0\_3 = FFT(y,Fs,2^11);

subplot(3,2,5);

plot(F0\_3,'\*');

xlabel('Frequency(Hz)');

ylabel('Signal Spectral');

title('Calculated F0 - FFT - FFT points = 2^11');

% Medium Filtering Function.

F0\_1\_Smoothed = MedFiltering(F0\_1,7);

F0\_2\_Smoothed = MedFiltering(F0\_2,7);

F0\_3\_Smoothed = MedFiltering(F0\_3,7);

% Plot all the result

subplot(3,2,2);

plot(F0\_1\_Smoothed,'\*');

xlabel('Frame number');

ylabel('Hz');

title('Calculated F0 - AutoCorrelation Func - After smoothing');

subplot(3,2,4);

plot(F0\_2\_Smoothed,'\*');

xlabel('Frame number');

ylabel('AMDF value');

title('Calculated F0 - AMDF - After smoothing');

subplot(3,2,6);

plot(F0\_3\_Smoothed,'\*');

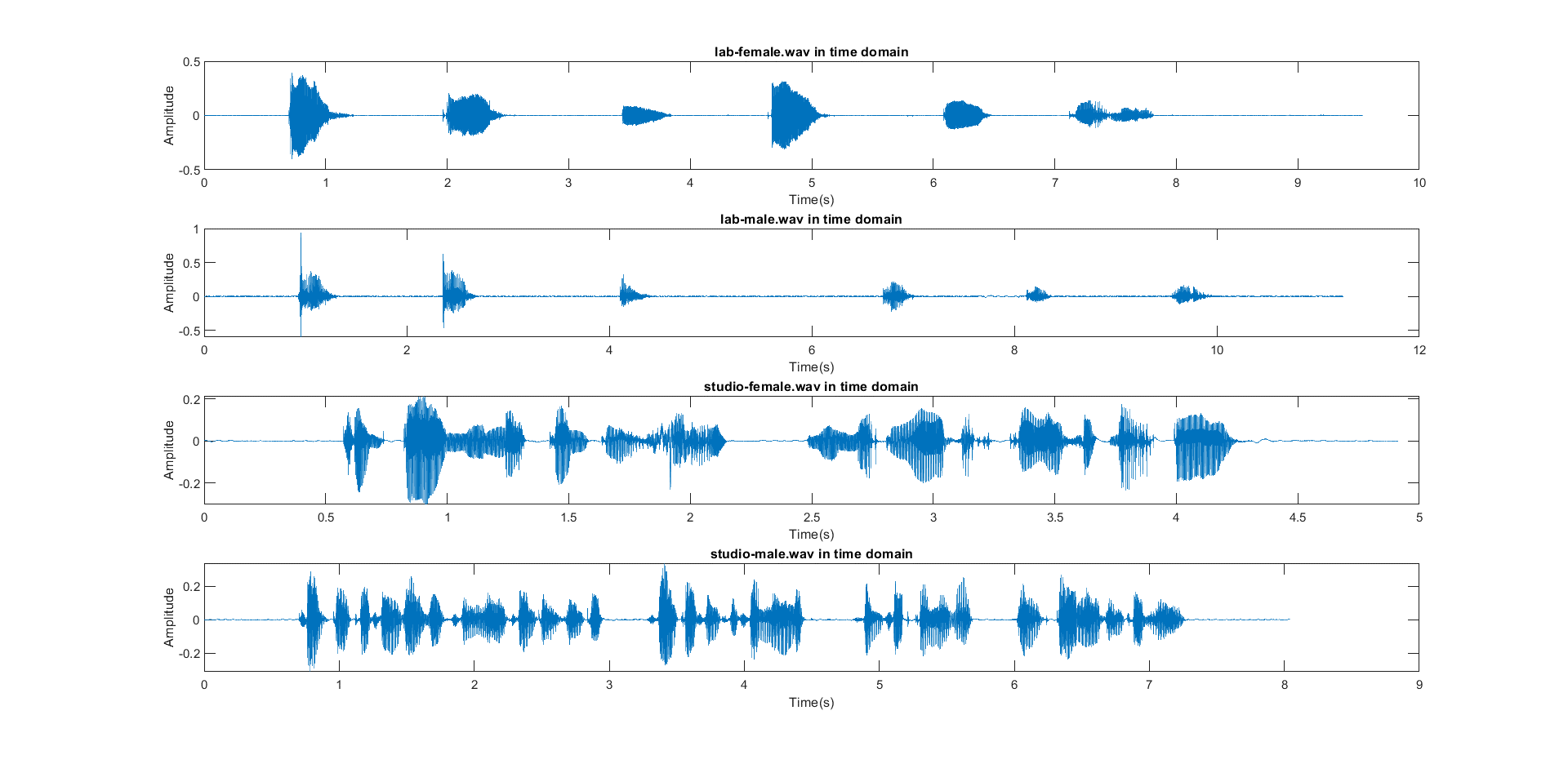
xlabel('Frequency(Hz)');

ylabel('Signal Spectral');

title('Calculated F0 - FFT - After smoothing');

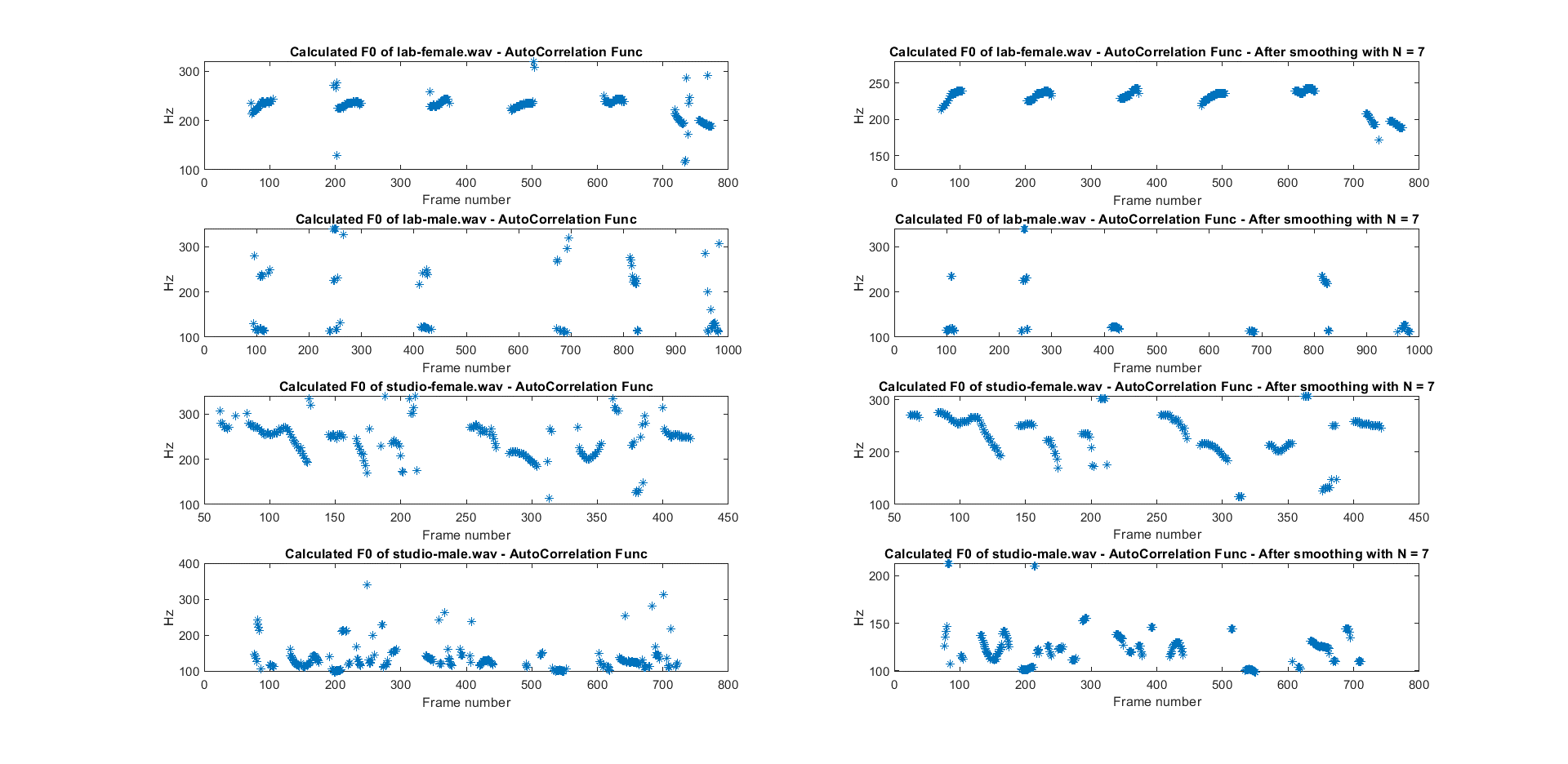
# KẾT QUẢ THỰC NGHIỆM

## Tín hiệu mẫu thử nghiệm



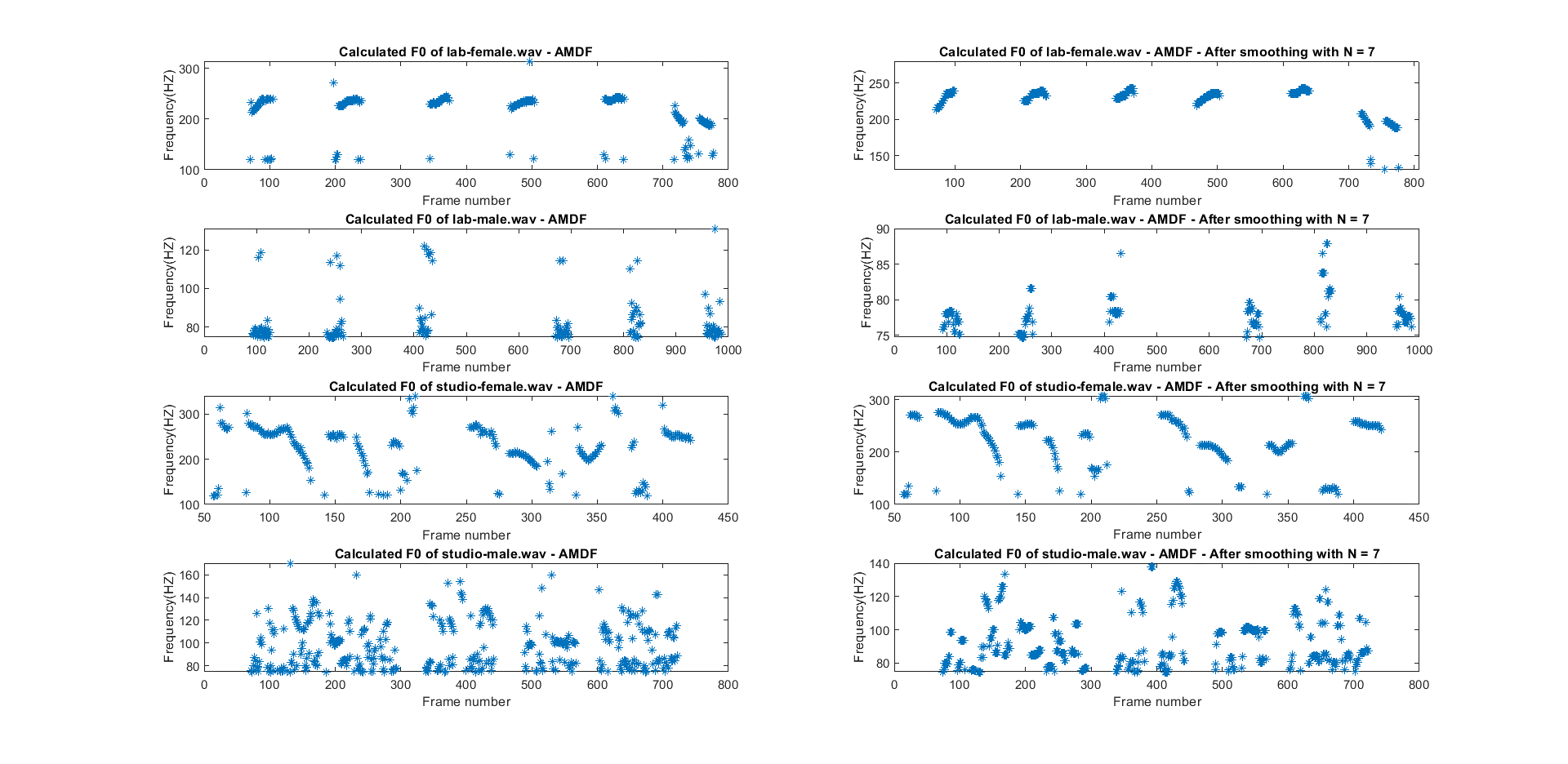
1. 4 tín hiệu mẫu từ folder TinHieuMau trên miền thời gian

## Kết quả tần số cơ bản của hàm tự tương quan trên 4 tín hiệu mẫu



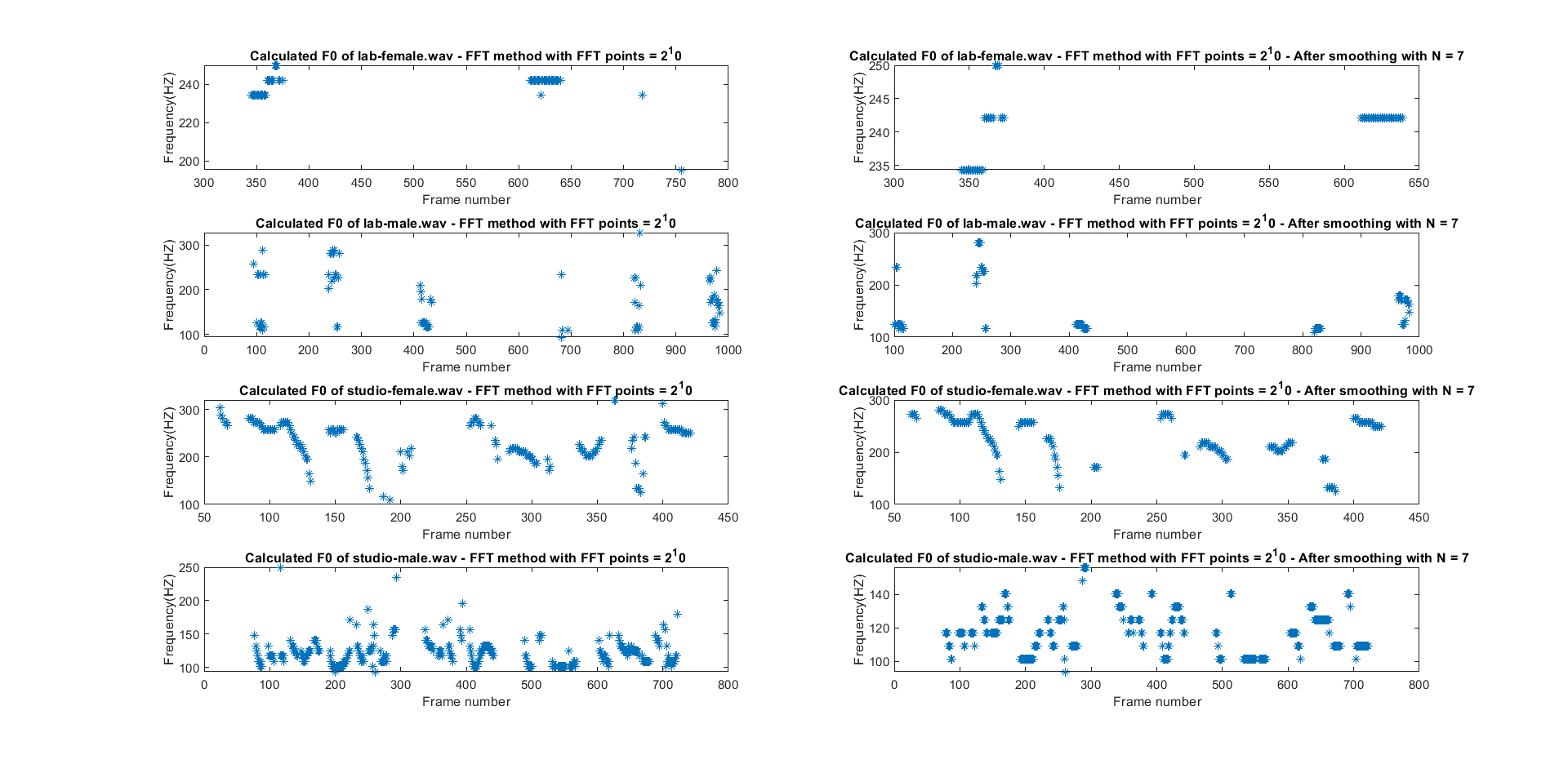
1. Kết quả F0 chạy hàm tự tương quan trên 4 tín hiệu mẫu (cột trái) và kết quả F0 sau khi lọc với N = 7 (cột trái)

## Kết quả tần số cơ bản của hàm vi sai biên độ trung bình trên 4 tín hiệu mẫu



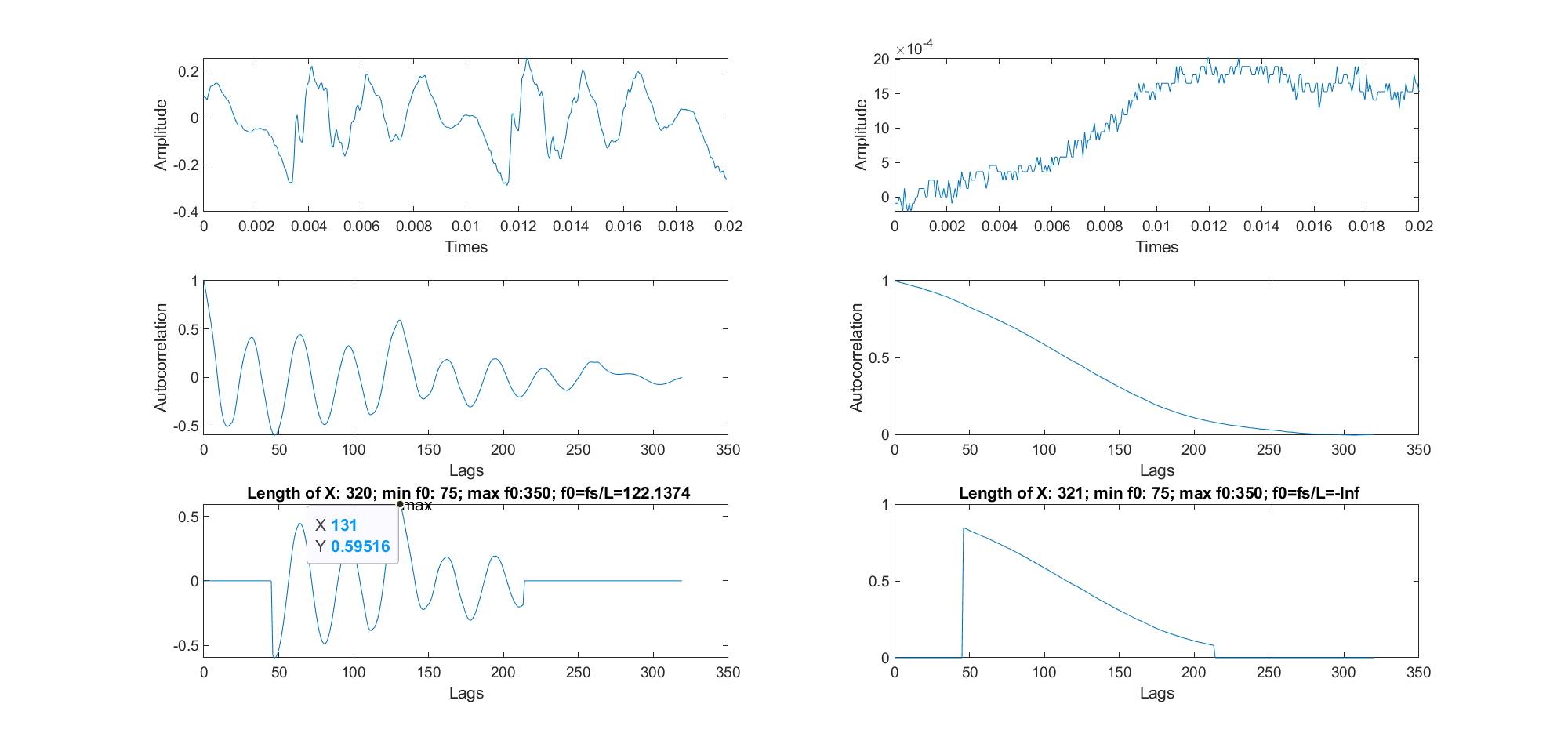
1. Kết quả F0 chạy hàm vi sai biên độ trung bình trên 4 tín hiệu mẫu (cột trái) và kết quả F0 sau khi lọc với N = 7 (cột trái)

## Kết quả tần số cơ bản của hàm phân tích phổ tín hiệu sử dụng biến đổi Fourier nhanh trên 4 tín hiệu mẫu



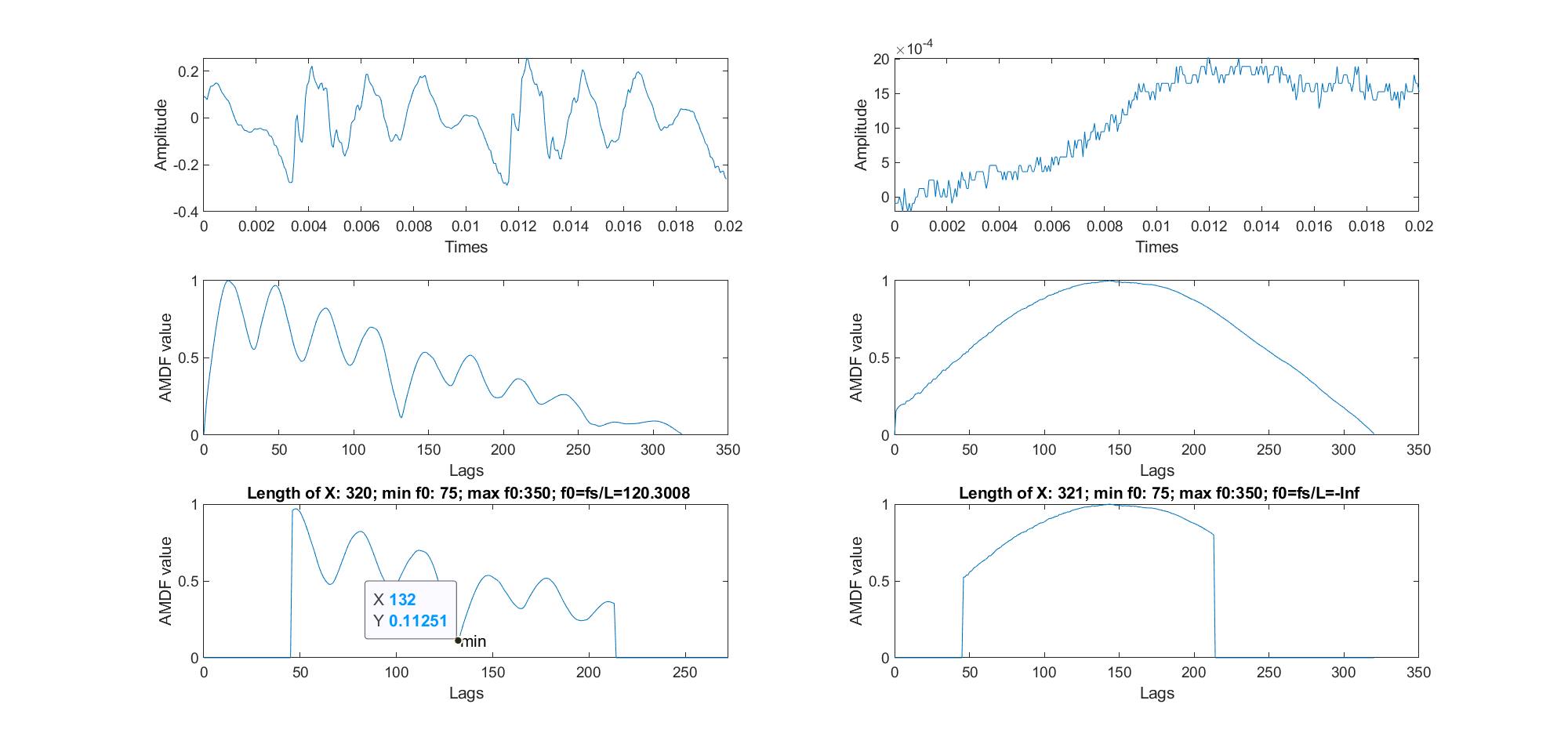
1. Kết quả F0 chạy hàm phân tích phổ tín hiệu dùng FFT (FFT points = 2^10) trên 4 tín hiệu mẫu (cột trái) và kết quả F0 sau khi lọc với N = 7 (cột trái)

## Hàm tự tương quan của 1 đoạn tín hiệu âm hữu thanh và 1 đoạn tín hiệu âm vô thanh dài 20ms



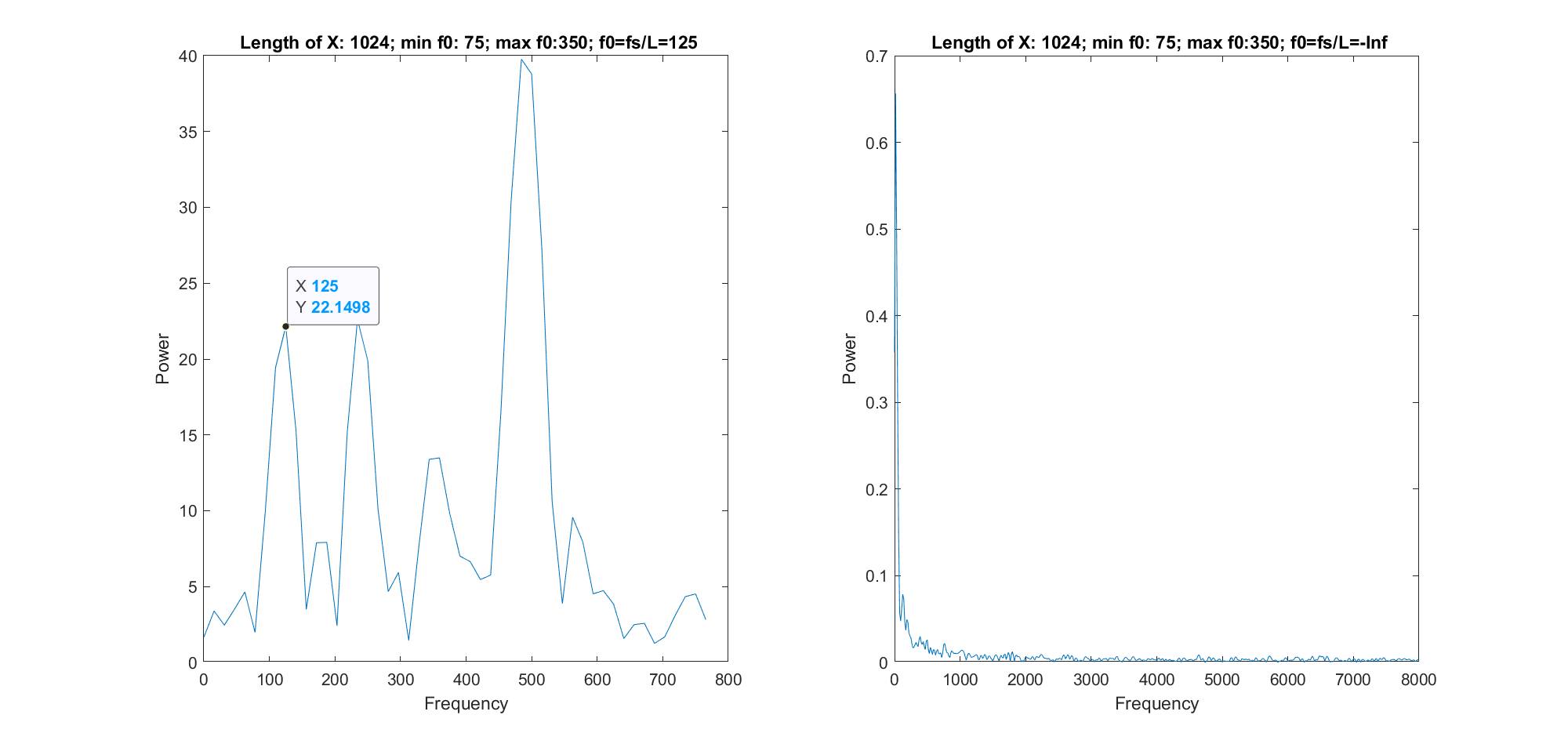
1. Hàm tự tương quan của 1 đoạn tín hiệu âm hữu thanh (cột trái) và âm vô thanh (cột phải) dài 20ms

## Hàm vi sai biên độ trung bình của 1 đoạn tín hiệu âm hữu thanh và 1 đoạn tín hiệu âm vô thanh dài 20ms



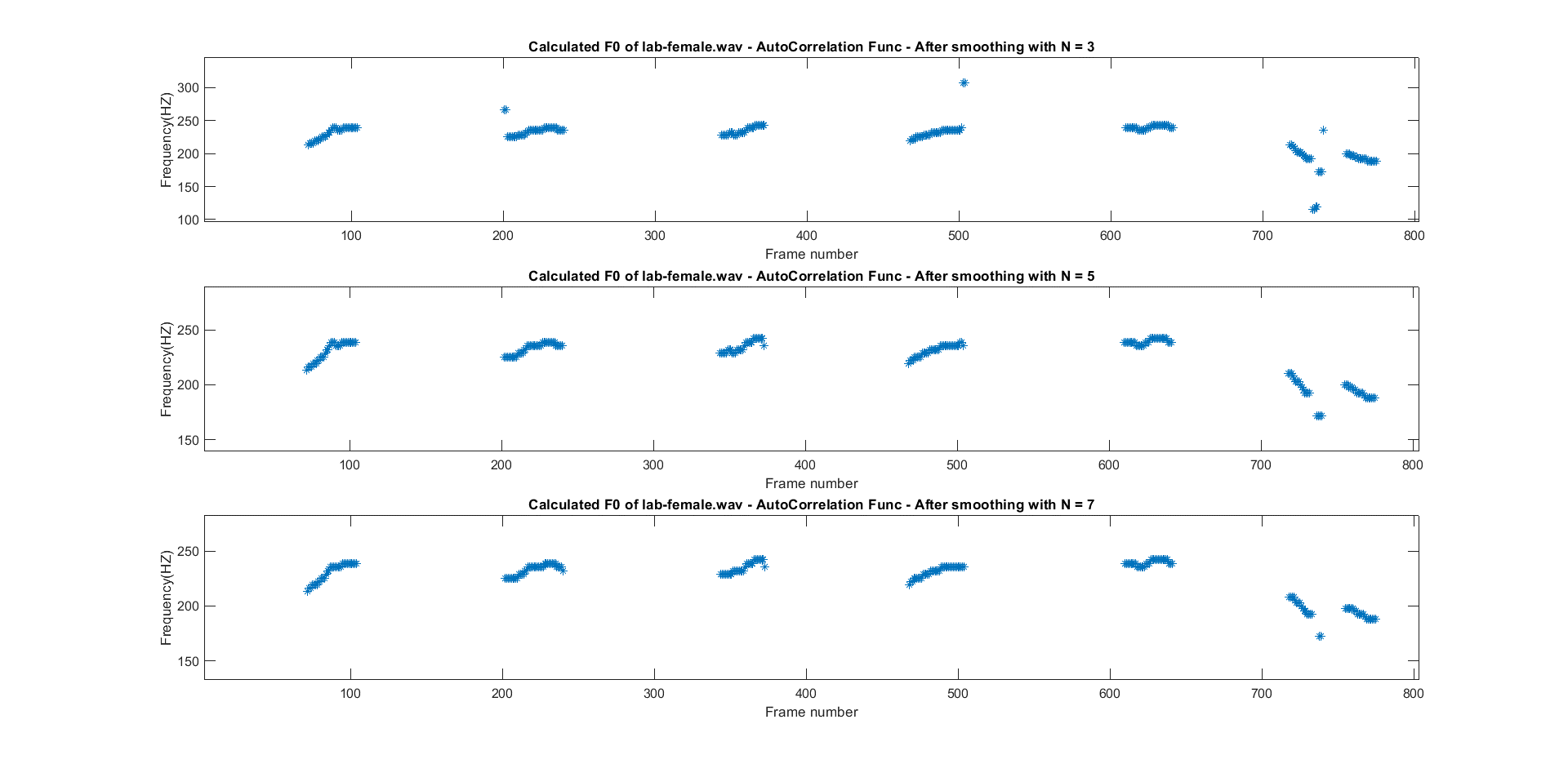
1. Hàm vi sai biên độ trung bình của 1 đoạn tín hiệu âm hữu thanh (cột trái) và âm vô thanh (cột phải) dài 20ms

## Phân tích phổ tín hiệu dùng FFT của 1 đoạn tín hiệu âm hữu thanh và 1 đoạn tín hiệu âm vô thanh dài 20ms



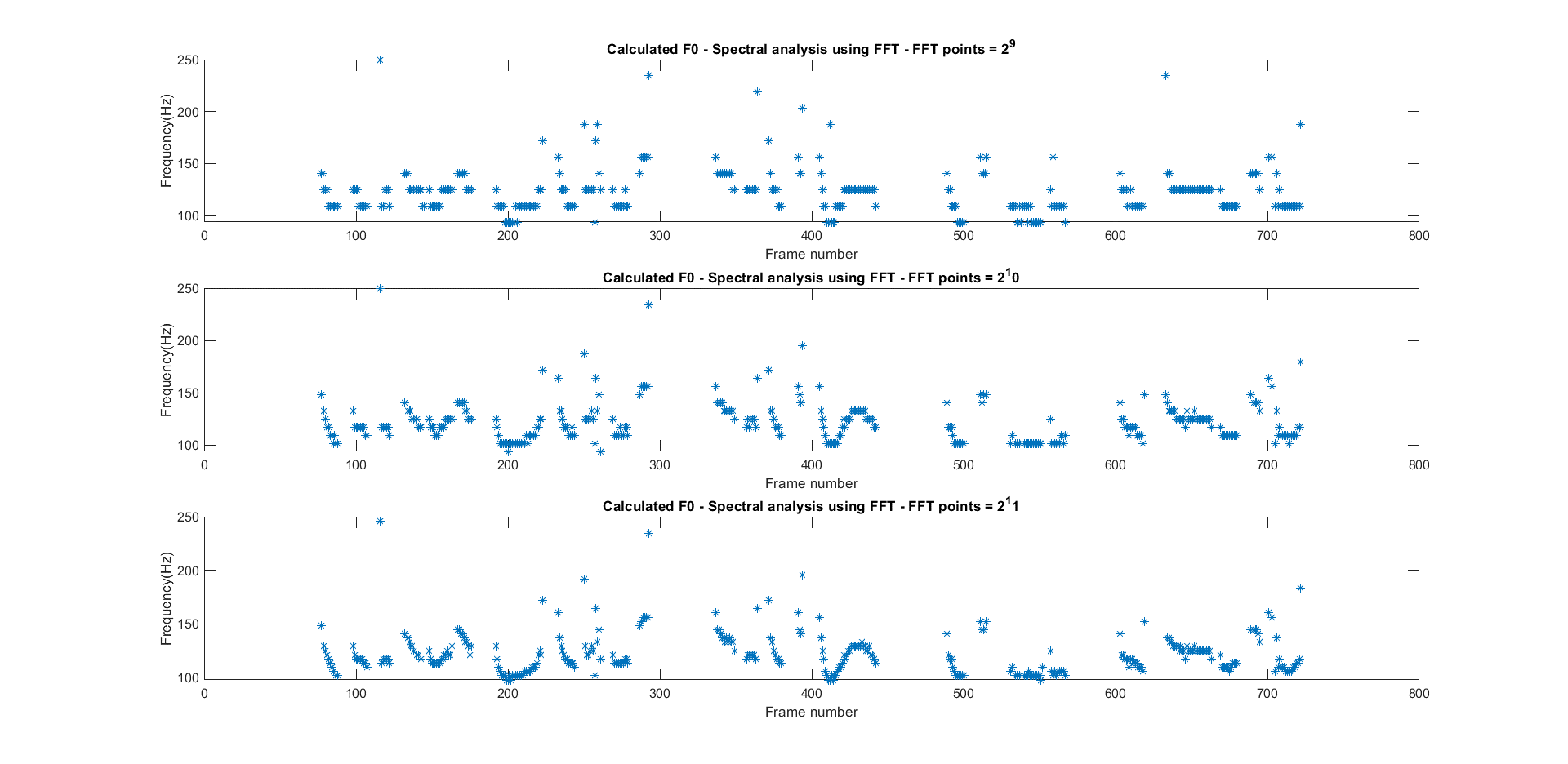
1. Phân tích phổ tín hiệu dùng FFT của 1 đoạn tín hiệu âm hữu thanh (cột trái) và âm vô thanh (cột phải) dài 20ms

## Khảo sát ảnh hưởng của độ dài của bộ lọc trong công thức lọc trung vị tới kết quả F0 thu được



1. Kết quả lọc trung vị với kết quả F0 của hàm tự tương quan từ file lab\_female.wav (N = 3, 5, 7)

## Khảo sát ảnh hưởng của các FFT points khác nhau tới kết quả F0 thu được



1. Kết quả F0 thu được từ phân tích phổ tín hiệu dùng FFT với các FFT points khác nhau (29, 210, 211) trên file studio\_male.wav

# KẾT LUẬN

## Hàm tự tương quan (AutoCorrelation)

Hàm tự tương quan được đánh giá là thuật toán tốt nhất trong cả 3 thuật toán trong việc xác định tần số cơ bản của tín hiệu. Nhờ vào đặc trưng cơ bản của đồ thị hàm tự tương quan có các đỉnh cực đại rất rõ, dễ dàng xác định được đối với các đoạn tín hiệu âm hữu thanh. Hơn nữa độ trễ (lag) của các đỉnh rất gần với bội chu kỳ cơ bản của tín hiệu nên kết quả tần số cơ bản F0 cũng sẽ rất sát. Đối với đoạn tín hiệu âm vô thanh, đồ thị hàm tự tương quan sẽ rất khác so với đồ thị của tín hiệu âm hữu thanh. Các đặc trưng này của đồ thị cũng sẽ dễ dàng được phân biệt. Điều này giúp hàm tự tương quan trở thành 1 trong những phương pháp phổ biến nhất để xác định tần số cơ bản của giọng nói và phân biệt tín hiệu tiếng nói với khoảng lặng.

Nhược điểm của thuật toán này là thời gian chạy và số lượng vòng lặp (hay độ phức tạp của thuật toán) khá đáng kể nhưng độ chính xác của thuật toán khá cao.

## Hàm vi sai biên độ trung bình (AMDF)

So với hàm tự tương quan, AMDF không có ưu điểm gì vượt trội rõ rệt. Thời gian chạy của chương trình và độ phức tạp như nhau nhưng lại có độ chính xác thấp hơn so với hàm tự tương quan. Với tín hiệu có tần số cơ bản cao (giọng nữ), AMDF trả về kết quả F0 đúng hơn so với tín hiệu có tần số cơ bản thấp (giọng nam).

Việc giới hạn đúng khoảng xác định tần số cơ bản sẽ giúp tăng độ chính xác của thuật toán này đáng kể. Cụ thể đối với giọng nữ, 2 tham số truyền vào sẽ là min\_F0 = 120 Hz và max\_F0 = 350 Hz (120 Hz F0 350 Hz). Còn đối với giọng nam, 2 tham số truyền vào sẽ là min\_F0 = 75 Hz và max\_F0 = 250 Hz (75 Hz F0 250 Hz).

## Phân tích phổ tín hiệu dùng biến đổi Fourier nhanh (FFT)

Kết quả tính tần số cơ bản F0 của phương pháp phân tích phổ tín hiệu phụ thuộc vào số điểm FFT (FFT points), với FFT points càng lớn thì sẽ cho ra kết quả càng chính xác. Nhưng đồng thời, nếu FFT points quá lớn thì cũng sẽ gây lãng phí dung lượng. Chọn FFT points bằng 210 hoặc 211 sẽ vừa cho kết quả đúng lẫn tối đa độ tối ưu của thuật toán.

So với 2 thuật toán hàm tự tương quan và hàm vi sai biên độ trung bình, thuật toán này có độ phức tạp bé hơn nhờ dùng hàm chuẩn fft() của thư viện Matlab.

## Lọc trung vị

Dựa vào các đồ thị trong phần III, mục H, ta có thể nhận thấy sự ảnh hưởng của bậc N tới kết quả lọc tần số cơ bản F0 thu được, với mỗi bậc lọc sẽ cho ra kết quả với độ mịn khác nhau. Chọn N = 7 sẽ cho kết quả lọc trung vị mịn nhất.

# TÀI LIỆU THAM KHẢO

1. Luận văn Thạc sĩ kỹ thuật Luận Văn Tâm, 2019 - Xác định tần số cơ bản của tín hiệu tiếng nói dùng hàm tự tương quan.
2. Average Magnitude Difference Function (AMDF). Retrieved from link: http://mirlab.org/jang/books/audiosignalprocessing/ptTimeDomainAmdf.asp?title=7-4%20Time-domain:%20PDF:%20AMDF
3. Biến đổi Fourier rời rạc. Retrieved from link: https://vi.wikipedia.org/wiki/Biến\_đổi\_Fourier\_rời\_rạc
4. Biến đổi Fourier nhanh. Retrieved from link: https://en.wikipedia.org/wiki/Biến\_đổi\_Fourier\_nhanh
5. Lawrence R. Rabiner, Ronald W. Schafe, “Prentice Hall - Digital Processing Of Speech Signals 1978”, p.158-159.
6. Window function. Retrieved from link: https://en.wikipedia.org/wiki/Window\_function