**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 HP:** 1022103.1810.16.10

|  |  |  |  |
| --- | --- | --- | --- |
| **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. Kết hợp lọc trung vị để làm trơn kết quả thu được. 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 là không dễ vì thường gặp những lỗi như tín hiệu có nhiễu…*

***Từ khóa****: Tính/đo tần số cơ bản, tính/đo chu kỳ pitch, biểu diễn trên miền tần số, hàm tự tương quan, lọc trung vị*

## Mục lục

1. **Đặt vấn đề**
2. **Lý thuyết xử lý tín hiệu tiếng nói và các thuật toán tính F0 và làm trơn F0 thu được bằng lọc trung vị**
3. Lý thuyết
4. Thuật toán tự tương quan
5. Thuật toán trên miền tần số
6. **Cài đặt các thuật toán**
7. Thuật toán tự tương quan
8. Thuật toán trên miền tần số
9. **Kết quả thực nghiệm**
10. Hình vẽ
11. Bảng biểu
12. **Kết luận**
13. **Những điều đã học được**
14. **Tài liệu tham khảo**

## ĐẶT VẤN ĐỀ

Xử lý và phân tích tiếng nói hiện nay đóng vai trò quan trọng trong các ngành khoa học và kĩ thuật. Phân  
tích tiếng nói thực hiện giải quyết các vấn đề để tìm ra một dạng thức tối ưu biểu diễn được tiếng  
nói một các hiệu quả. Nó là cơ sở cho việc phát triển các kỹ thuật, công nghệ tổng hợp, nhận  
dạng và nâng cao chất lượng tín hiệu tiếng nói. Phân tích tiếng nói thường thực hiện việc trích  
chọn hoặc chuyển đổi tín hiệu tiếng nói sang một dạng thức biểu diễn khác sao cho có thể  
biểu diễn thông tin tiếng nói tốt hơn theo cách mà chúng ta cần [1]. Để làm được điều đó, việc xác định tần số cơ bản là rất quan trọng. Có nhiều phương pháp khác nhau như sử dụng hàm vi sai biên độ trung bình (AMDF), LPC, xử lý đồng hình (homomophic), hàm tự tương quan (autocorrelation), sử dụng bộ lọc đảo và hàm tự tương quan, phân tích phổ trên miền tần số… để xác định được tần số cơ bản của tiếng nói [1].

1. **Lý thuyết xử lý tín hiệu tiếng nói và các thuật toán tính F0 và làm trơn F0 thu được bằng lọc trung vị**
2. **Lý thuyết**

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.

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. F0 phụ thuộc vào giới tính, lứa tuổi của người phát âm, F0 thay đổi theo thanh điệu và F0 cũng ảnh hưởng đến ngữ điệu của câu nói [1]. Chu kỳ cơ bản là khoảng thời gian nhỏ nhất của tín hiệu tuần hoàn trên miền thời gian[3]. 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 để tìm được tần số cơ bản của tiếng nói thông qua 5 nguyên âm /u/ /e/ /o/ /a/ /i/. Lọc trung vị (median smoothing) sử dụng để làm trơn kết quả F0 thu được.

1. **Thuật toán tìm tần số cơ bản tự tương quan**
2. **Đặt vấn đề**

Việc xác định tần số cơ bản của tín hiệu là rất quan trọng trong việc phân tích và xử lý tín hiệu

### 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.

1. **Cơ sở lý thuyết**
2. **Phân tích tiếng nói trong miền thời gian**

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.

1. **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 [1]. Với tín hiệu tuần hoàn, ta có định nghĩa hàm tự tương quan:

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

rxx(l) =

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.

1. **Sơ đồ khối**

Tín hiệu gốc

Tính hàm tự tương quan cho mỗi khung tín hiệu

Chia khung cho tín hiệu (30ms)

Tìm cực đại của hàm tự tương quan tại mỗi khung

Tìm độ trễ của cực đại ứng với mỗi khung tương ứng

Tính tần số cơ bản F0

Hình 1. Sơ đồ khối thuật toán tìm F0 trên miền thời gian dùng hàm tự tương quan

1. **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.

1. **Vấn đề phát sinh**

* **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.

1. **Thuật toán tìm tần số cơ bản trên miền tần số**
2. **Đặt vấn đề**

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**
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.

1. **Sơ đồ khối**

Sơ đồ khối thuật toán trên miền tần được trình bày trên Hình ??

Tín Hiệu Gốc

Phân tích phổ, biến đổi Fourier cho mỗi khung tín hiệu

Chia khung cho tín hiệu (30ms)

Tìm đỉnh đầu tiên của mỗi khung tín hiệu

Tính tần số cơ bản, là mảng F0 chứa các đỉnh của các khung đã chia

Hình 2. Sơ đồ khối thuật toán tìm F0 trên miền tần số

1. **Các tham số quan trọng của hàm miền**

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. **Các vấn đề phát sinh**

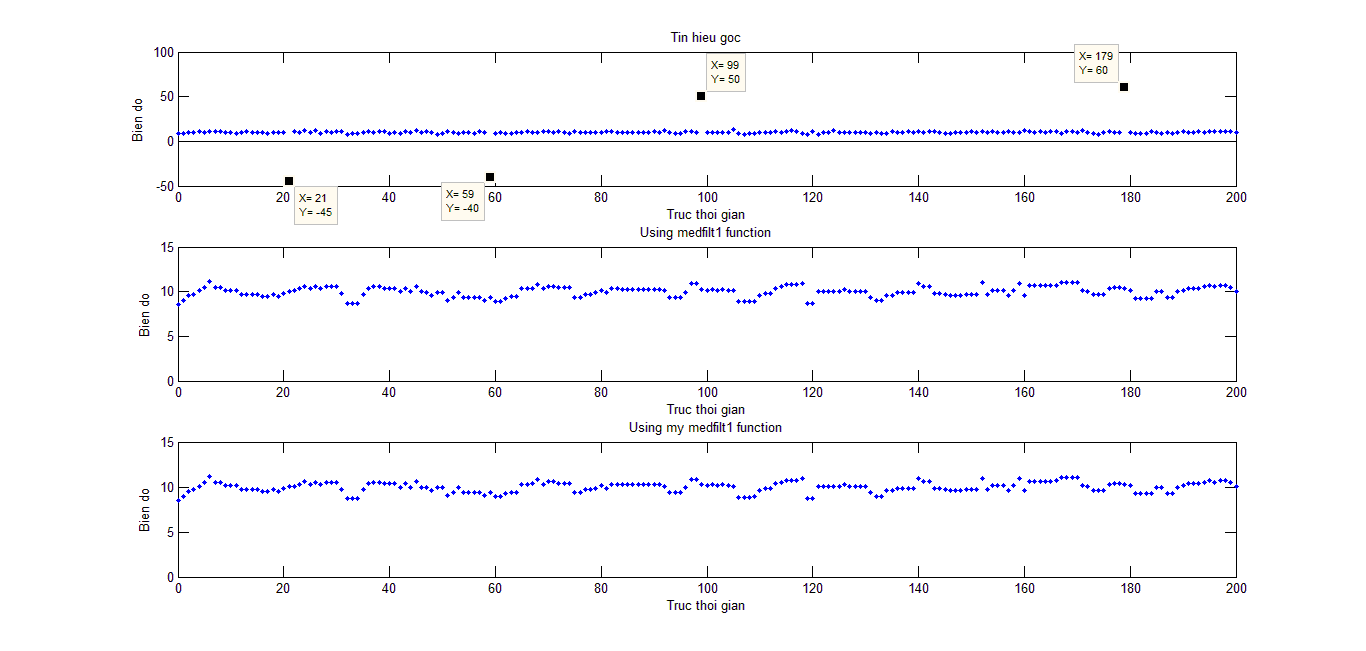
Một bài toán mở quan trọng về mặt lý thuyết là chứng minh chặn dưới cho độ phức tạp tính toán và số phép tính của biến đổi Fourier nhanh. Hiện vẫn chưa có chứng minh nào cho việc DFT có thực sự đòi hỏi Ω(N\*logN) phép tính, ngay cả trong trường hợp kích thước N là lũy thừa của hai, mặc dù không có thuật 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].

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.

1. **Thuật toán lọc trung vị để làm trơn F0 thu được**
2. **Đặt vấn đề**

Bộ lọc trung vị là một [kỹ thuật lọc số](https://en.wikipedia.org/wiki/Digital_filter) phi tuyến tính, thường được sử dụng để loại bỏ [nhiễu](https://en.wikipedia.org/wiki/Signal_noise) khỏi hình ảnh hoặc tín hiệu, [giảm tiếng ồn](https://en.wikipedia.org/wiki/Noise_reduction). Đây là một bước xử lý trước điển hình để cải thiện kết quả sau khi xử lý. Lọc trung vị được sử dụng rộng rãi trong [xử lý ảnh](https://en.wikipedia.org/wiki/Image_processing) kỹ thuật số bởi vì, trong điều kiện nhất định, nó bảo đảm việc loại bỏ nhiễu nhưng tín hiệu hay hình ảnh vẫn không bị mất nhiều thông tin. Lọc trung vị có nhiều ứng dụng trong [xử lý tín hiệu](https://en.wikipedia.org/wiki/Signal_processing). Bài báo này dùng lọc trung vị để làm trơn thu được sau khi trả qua quá trình tìm bằng hàm tự tương quan hay trên miền tần số [3]. Sau đây là một tính hiệu bất kì bị nhiễm nhiễu, có vài điểm có biên độ bất thường cần phải lọc.



Hình 3. Ví dụ cụ thể về tác dụng của lọc trung vị.

Nhận xét: Ta thấy những điểm có biên độ bất thường được làm trơn về phù hợp với tín hiệu gốc

Một tín hiệu giọng nói x(n) sẽ có công thức: x(n) = S[x(n)] + R[x(n)] [2].

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).

1. **Cơ sở lý thuyết**

Trong [lý thuyết xác suất](https://vi.wikipedia.org/wiki/L%C3%BD_thuy%E1%BA%BFt_x%C3%A1c_su%E1%BA%A5t) và [thống kê](https://vi.wikipedia.org/wiki/Khoa_h%E1%BB%8Dc_Th%E1%BB%91ng_k%C3%AA), **số trung vị** ([tiếng Anh](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Anh): *median*) là một số tách giữa nửa lớn hơn và nửa bé hơn của một mẫu, một quần thể, hay một [phân bố xác suất](https://vi.wikipedia.org/wiki/Ph%C3%A2n_ph%E1%BB%91i_x%C3%A1c_su%E1%BA%A5t). Nó là giá trị giữa trong một phân bố, mà số các số nằm trên hay dưới con số đó là bằng nhau. Điều đó có nghĩa rằng 1/2 quần thể sẽ có các giá trị nhỏ hơn hay bằng số trung vị, và một nửa quần thể sẽ có giá trị bằng hoặc lớn hơn số trung vị [4].

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 2, 5, 8, 10, 14, 16, 22 thì số trung vị sẽ là 10.

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 5, 6, 9, 11, 15, 17, 20, 24 thì số trung vị sẽ là trung bình cộng của 11 và 15 là 13.

1. **Các tham số quan trọng của hàm lọc trung vị**

Trong matlab, đã có sẵn hàm thư viện để lọc trung vị của tín hiệu cần lọc. Đó là hàm medfilt1 (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. Bài báo này sẽ sử dụng hàm my\_medfilt(x, N) có chức năng giống như hàm thư viện medfilt1(x, N).

1. **Cài đặt các thuật toán**
2. **Thuật toán tự tương quan**

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.

Đoạn code sau đây lấy âm /o/ của sinh viên Nguyễn Xuân Lực làm ví dụ

[y,Fs] = audioread('oluc.wav');

t=1/Fs:1/Fs:(length(y)/Fs);

subplot(3,1,3);

plot(t,y);

title('Tin hieu input');

xlabel('Chi so thoi gian');

ylabel('Bien do');

frame\_leng=0.03 \* Fs;

half = round(frame\_leng/2);

number\_frame = round(length(y)/half) - 1;

result = [];

F0 = [];

k = 1;

for i = 1 : number\_frame - 1

range = ((i-1)\*half + 1 ): (frame\_leng + (i-1)\*half); % vi tri khung k chay

frame = y(range);

[rxx, lags] = xcorr(frame); % tu tuong quan cua fram

lags1 = lags((length(lags) - 1 )/2 + 1: length(lags));

rxx1 = rxx((length(lags) - 1 )/2 + 1: length(lags));

%lay doan tu (100:(length(lags11)-1)) de tim cuc dai

lags11 = lags1(1 : length(lags1) - 1); %((i-1)\*lags1 +1 ):i\*(length(lags1)-1)

rx1 = rxx1(1 : length(rxx1) - 1);

[pksa , locs] = findpeaks(rx1);

[a,x] = max(pksa);

result(i) = Fs/locs(x);

if(result(i) > 400 || result(i) < 80)

result(i) = 0;

else

F0(k) = result(i);

k= k+1;

end

end

%F0 tren ca tin hieu truoc khi loc

subplot(3,1,2);

stem(result, '.','LineStyle','none','MarkerFaceColor','w');

title('Truoc khi loc');

xlabel('n');

ylabel('F');

F0\_mean = mean(F0); %F0 trung bình trong doan tuan hoan

F0\_std = std(F0); %F0 do lech chuan trong doan tuan hoan

%TIEN HANH LOC TRUNG VI VOI N = 5

%Loc F0 tren ca tin hieu sau khi loc voi N = 5

F2\_result = my\_medfilt1(result, 5);

subplot(3, 1, 1);

%ve F0 tren ca tin hieu sau khi loc voi N = 5

stem(F2\_result,'.','LineStyle','none','MarkerFaceColor','w');

title('Sau khi loc voi N = 5');

xlabel('n');

ylabel('F');

%Loc F0 doan tuan hoan

F2 = my\_medfilt1(F0, 5);

F2\_mean = mean(F2); %F2 trung bình trong doan tuan hoan

F2\_std = std(F2); %F2 do lech chuan trong doan tuan hoan

%Sai so tren F0 so voi F0 thu cong

SaiSo = RMSE(F2, 141.35);

Kết quả F0 sẽ được trình bày trong phần IV

1. **Thuật toán trên miền tần số**

clear;clc;clf;

[y,Fs] = audioread('D:\codeIT\matlab\Thithuchanh\ukhanh.wav');

%Do dai khung tin hieu (30ms)

frame\_len = round(0.1\*Fs);

half = round(frame\_len/2);

temp = [];

F0 = [];

i=1;

%Vong lap chay tu 1 -> so khung

for k = 1 : (length(y)/half) - 1

range = ((k-1)\*half + 1):(frame\_len + (k-1)\*half);

frame = y(range);

%Hien thi tin hieu dau vao la mau xanh va khung duoc chon mau do

subplot(3,1,1)

plot(y);

title('Tin Hieu');

xlabel('Chi so thoi gian');

ylabel('Bien Do');

set(gca, 'xtick',[],'position',[ 0.05 0.85 0.91 0.12])%gca,

hold on;

yn = ones(size(y))\*NaN;

yn(range) = frame;

plot(yn,'r');

hold off

%Hien thi tung khung

subplot(3,1,2)

plot(frame);

title('Pho bien do');

xlabel('Tan So');

ylabel('Bien Do');

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

%Su dung fft() de tim tan so cua khung

P2 = abs(fft(frame,1024));

P1 = P2(1:length(P2)/2+1);

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

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

%Su dung ham findpeak de tim cac dinh

[pks,locs] = findpeaks(P1,freq, 'MinPeakDistance',80);

temp(k) = locs(1);

if(temp(k) > 400 || temp(k) < 80)

temp(k) =0;

else

subplot(3,1,3)

findpeaks(P1,freq,'MinPeakDistance',80);

title('Tim cac dinh vua tim duoc');

xlabel('Tan So');

ylabel('Bien Do');

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

F0(i) = temp(k);

i= i+1;

end

pause;

end

figure(2);

subplot(4,1,1);

stem(temp, '.','LineStyle','none','MarkerFaceColor','w');

title('Truoc khi loc');

xlabel('n');

ylabel('F');

F0\_mean = mean(F0); %F0 trung bình trong doan tuan hoan

F0\_std = std(F0); %F0 do lech chuan trong doan tuan hoan

%tien hanh loc trung vi

%Loc F0 tren ca tin hieu sau khi loc voi N = 5

F2\_temp = my\_medfilt1(temp, 5);

subplot(4, 1, 3);

%ve F0 tren ca tin hieu sau khi loc voi N = 5

stem(F2\_temp,'.','LineStyle','none','MarkerFaceColor','w');

title('Sau khi loc voi N = 5');

xlabel('n');

ylabel('F');

%Loc F0 doan tuan hoan

F2 = my\_medfilt1(F0, 5);

F2\_mean = mean(F2); %F2 trung bình trong doan tuan hoan

F2\_std = std(F2); %F2 do lech chuan trong doan tuan hoan

1. **Thuật toán lọc trung vị để làm trơn F0 thu được**

* **Hàm sắp xếp tăng dần Sort mảng 1 chiều để phục vụ cho hàm my\_medfilt1**

function y = Sort(y)

for i = 1:length(y)-1

for j = i+1:length(y)

if (y(i) > y(j))

temp = y(i);

y(i) = y(j);

y(j) = temp;

end

end

end

end

%y là mảng cần sắp xếp, hàm Sort trả về một mảng đã sắp xếp

* **Hàm lọc trung vị 1 chiều my\_medfilt1 giống với hàm thư viện của matlab medfilt1**

function z = my\_medfilt1(y, m)

z = y;

temp = 1:m;

for x = 1:length(y)

for i = 1:m

if(x < ceil(m/2))

if(i < ceil(m/2)-x+1)

temp(i) = 0;

else

temp(i) = y(x+i-ceil(m/2));

end

else

if(x > length(y) - ceil(m/2) + 1)

if(i < length(y) + ceil(m/2) - x + 1)

temp(i) = y(x-ceil(m/2) + i);

else

temp(i) = 0;

end

else

temp(i) = y(x-ceil(m/2)+i);

end

end

end

temp = Sort(temp);

z(x) = temp((length(temp)+1)/2);

end

end

* **Ví dụ cụ thể về lọc trung vị 1 chiều**

Xét đoạn code sau dùng hàm my\_medfilt1 để lọc với N bằng 5

n = -5:5;

x = [ 6,5,50,-9,-4,--3,10,1,12,10,-42];

y = medfilt1(x,5);

z = my\_medfilt1(x,5);

figure(1)

subplot(3,1,1)

stem(n,x, 'filled');

title('Tinh hieu goc');

xlabel('n');

ylabel('x');

subplot(3,1,2)

stem(n,y, 'filled');

title('Using medfilt1 function');

xlabel('n');

ylabel('x');

subplot(3,1,3)

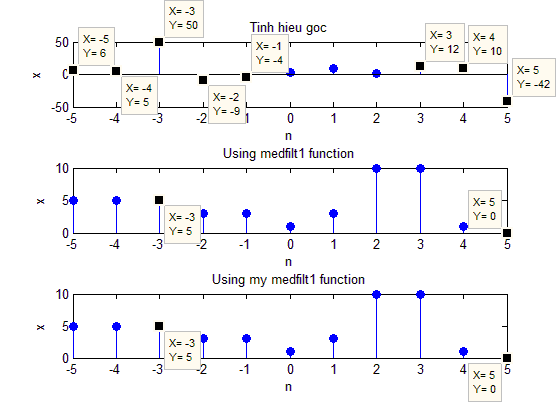
stem(n,z, 'filled');

title('Using my medfilt1 function');

xlabel('n');

ylabel('x');

Kết quả của lọc trung vị được thể hiện ở Hình 4.



Hình 4. Tín hiệu trước và sau khi đi qua lọc trung vị.

Xét tín hiệu ban đầu, tại n = -3 thì giá trị của x là 50. Áp dụng lọc trung vị cho tín hiệu x với bậc là 5, tức là ta xét các giá trị của x tại n = -5 -4,-3,-2,-1. Mỗi giá trị tại đó tương ứng với 6, 5 , 50, -9, -4. Sau khi sắp xếp tăng dần ta được -9, -4, 5, 6, 50 và chọn trung vị, ta được số trung vị là 5. Và 5 sẽ được chọn là giá trị cho tín hiệu x sau khi lọc tại n = -3.

Một ví dụ khác, tại n = 5, giá trị của x sẽ là -42, sau khi lọc tín hiệu x với bậc là 5 (xét các giá trị n = 3, -4, 5, 6, 7 tương ứng với các giá trị là -9 , 4, -50, 10, 1) thì giá trị của x tại n = 0 là -4.

Nhận xét: 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 là quá mịn dẫn đến sự sai lệch so với kết quả thực tiễn.

1. **Thuật toán tính sai số RMSE**

function z = RMSE(y,f)

temp = y;

for i = 1:length(y)

temp =abs(y(i)-f)\*abs(y(i)-f);

end

z = sqrt( mean(temp) );

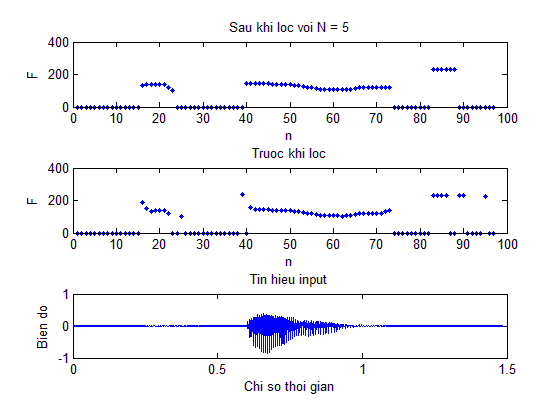
end

% y la mang tan so chua cac tan so cua cac khung, f la tan so chuan tinh thu cong

% z la gia tri sai so

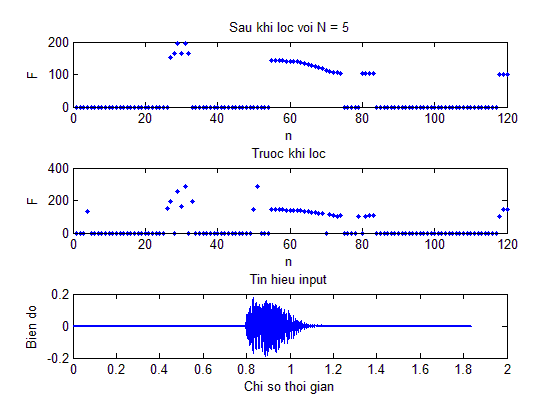
1. **Kết quả thực nghiệm**
2. **Hình vẽ**
3. **Hàm tự tương quan**

* Trường hợp xấu nhất /u/ của Đinh Quang Duy



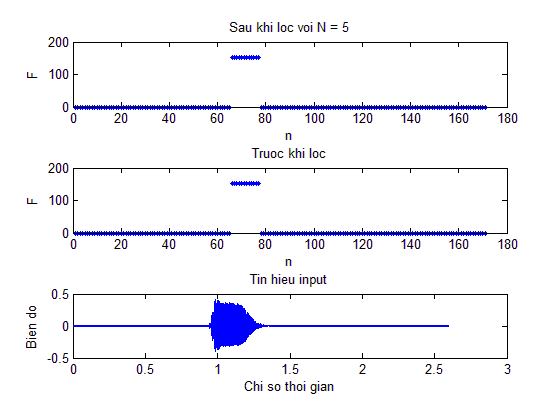
Hình 5. F0 của trường hợp xấu nhất (.bmp)

* Trường họp trung bình /o/ của Đinh Quang Duy



Hình 6. F0 của trường hợp trung bình (.bmp)

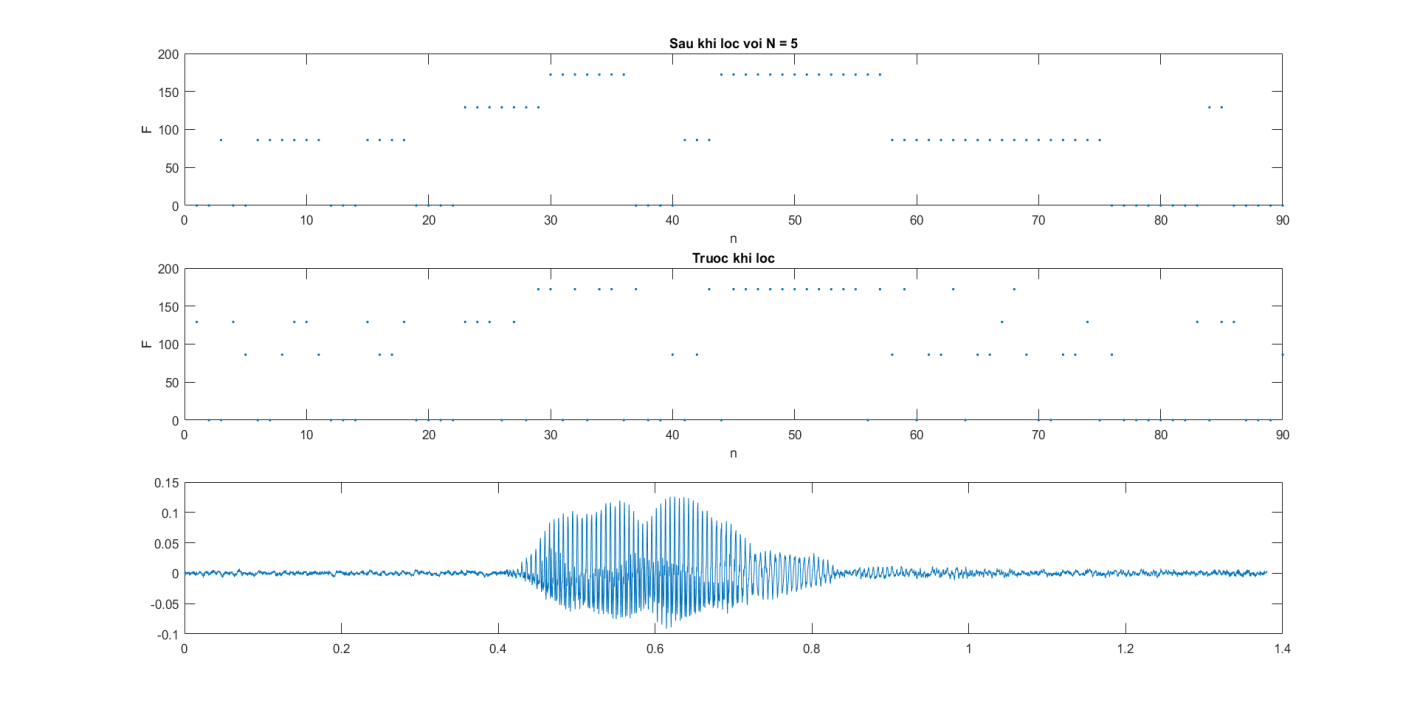
* Trường họp tốt nhất /o/ của Nguyễn Xuân Lực



Hình 7. F0 của trường hợp tốt nhất (.bmp)

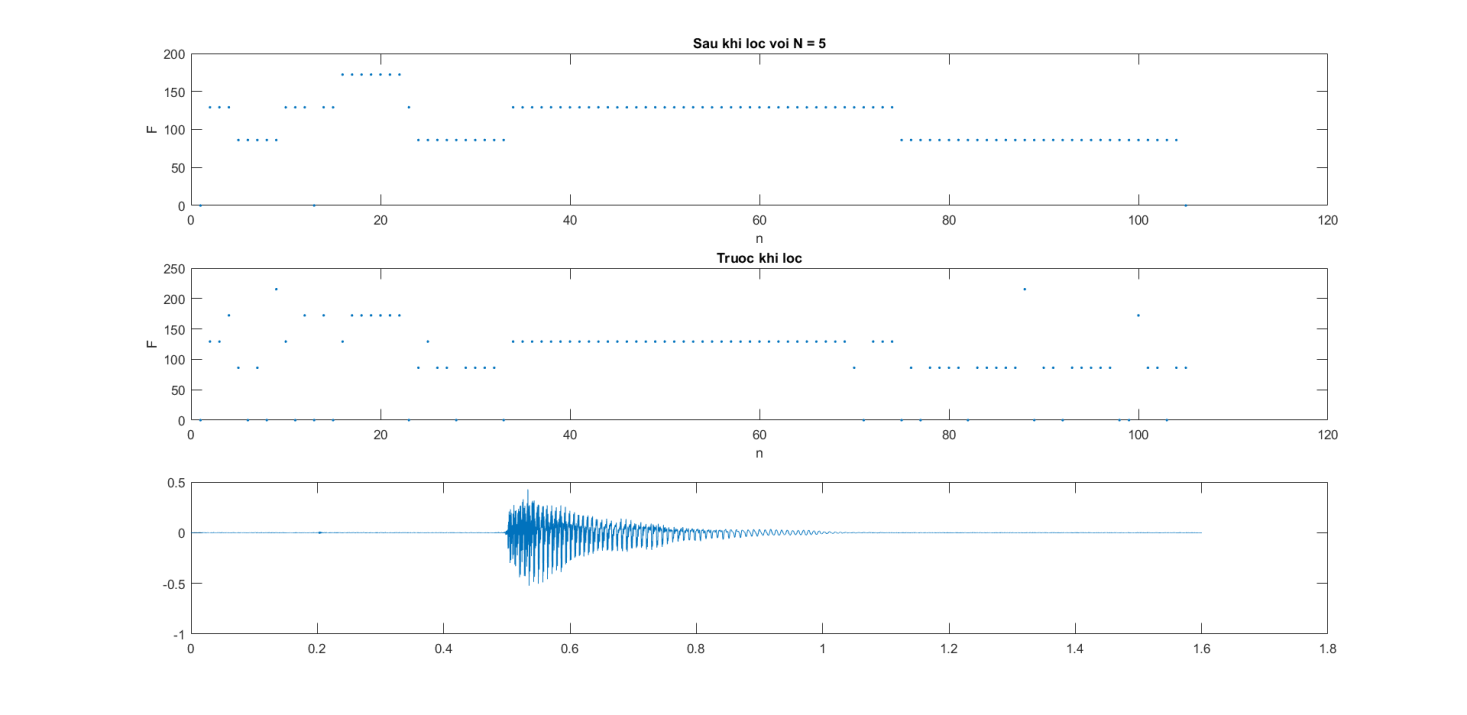
1. **F0 trên miền tần số**

* Trường hợp xấu /u/ của Hồ Gia Khánh



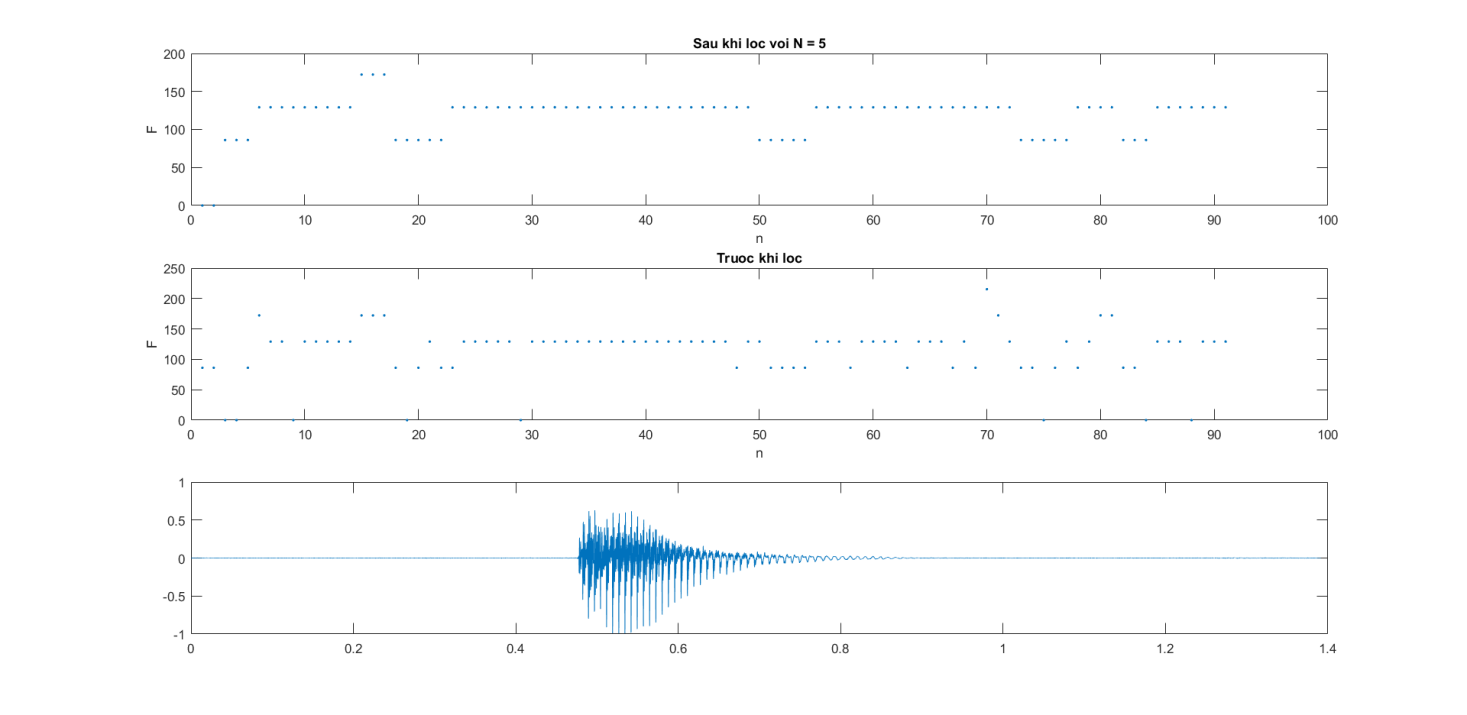
Hình 8. F0 của trường hợp xấu nhất (.bmp)

* Trường hợp trung bình /e/ của Đinh Quang Duy



Hình 9. F0 của trường hợp trung bình (.bmp)

* Trường hợp tốt nhất /a/ của Đinh Quang Duy



Hình 10. F0 của trường hợp tốt nhất (.bmp)

1. **Bảng biểu**
2. **Hàm tự tương quan**

* **Âm của Hồ Gia Khánh**

Bảng 1: kết quá đo được của Hồ Gia Khánh

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Âm | F0 chuẩn | F0 TTQ trước lọc | F0 TTQ sau lọc | Sai số RMSE | F0 FFT trước lọc | F0 FFT sau lọc | Sai số RMSE |
| /i/ | 170,3 | 165,71 | 166,82 | 16,64 | 135,07 | 134,09 | 84,18 |
| /u/ | 175 | 182,29 | 183,66 | 71,23 | 133.19 | 133.19 | 88.87 |
| /o/ | 164,6 | 167,13 | 167,04 | 6,36 | 124,8 | 122,19 | 78,48 |
| /a/ | 162,7 | 178,53 | 178,72 | 43,83 | 139,97 | 139,97 | 76,57 |
| /e/ | 173,6 | 188,99 | 183,66 | 64,17 | 112,64 | 104,35 | 87,47 |

* **Âm của Đinh Quang Duy**

Bảng 2: kết quá đo được của Đinh Quang Duy

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Âm | F0 chuẩn | F0 TTQ trước lọc | F0 TTQ sau lọc | Sai số RMSE | F0 FFT trước lọc | F0 FFT sau lọc | Sai số RMSE |
| /i/ | 134,86 | 154,5 | 152,84 | 11,65 | 140,26 | 139,87 | 5,25 |
| /u/ | 143,18 | 144,79 | 142,42 | 79,55 | 123,88 | 121,46 | 57,05 |
| /o/ | 138,68 | 145,56 | 136,47 | 39,13 | 123,77 | 121,83 | 9,48 |
| /a/ | 128,95 | 117,68 | 117,18 | 18,42 | 122,45 | 122,97 | 0,25 |
| /e/ | 134,45 | 130,65 | 126,64 | 44,9 | 120,78 | 119,3 | 48,72 |

* **Âm của Nguyễn Xuân Lực**

Bảng 3: kết quá đo được của Nguyễn Xuân Lực

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Âm | F0chuẩn | F0 TTQ trước lọc | F0 TTQ sau lọc | Sai số RMSE | F0 FFT trước lọc | F0 FFT sau lọc | Sai số RMSE |
| /i/ | 154,19 | 133,8 | 133,44 | 42,54 | 161,44 | 157,41 | 68,06 |
| /u/ | 160,95 | 152,36 | 145,16 | 60,95 | 149,62 | 138,52 | 31,75 |
| /o/ | 153,66 | 152,78 | 152,78 | 2,63 | 138,74 | 132,89 | 67,53 |
| /a/ | 141,35 | 133,03 | 132,96 | 27,69 | 114,41 | 100,27 | 55,22 |
| /e/ | 146,03 | 130,25 | 130,25 | 62,66 | 172,27 | 162,64 | 59,9 |

1. **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 tần số tính được từ thuật toán tìm tần số cơ bản F0 trên miền thời gian dùng hàm tự tương quan có kết quả gần giống với tần số chuẩn đo thủ công hơn thuật toán tìm tần số cơ bản F0 trên miền tần số.

1. **Những điều đã học được**

Qua bài thực hành trên chúng em một phần hiểu được cách sử dụng phần mềm Matlab cho việc tính toán và xử lý tín hiệu, chúng em còn biết thêm được 2 cách để tìm tần số cơ bản của một tín hiệu tiếng nói bất kì đó là: thuật toán tìm tần số cơ bản F0 trên miền thời gian sử dụng hàm tự tương quan và thuật toán tìm tần số cơ bản F0 trên miền tần số, và cách cài đặt chúng trong Matlab.

Ngoài ra chúng em còn học được các kỹ năng làm việc nhóm sao cho hợp lý.

1. **Lời cảm ơn**

## Cảm ơn TS. Ninh Khánh Duy, giảng viên bộ môn Hệ thống nhúng, khoa Công nghệ thông tin, trường Đại học Bách Khoa, Đại học Đà Nẵng đã hướng dẫn và hỗ trợ chúng em trong quá trình học tập và tìm hiểu cách làm bài thực hành và báo cáo.

1. **Tài liệu tham khảo**

[1] Bài giảng Xử lý tiếng nói, Học viện bưu chính viễn thông, 2010.

[2] Prentice Hall –Digital Processing Of Speech Signals

[3] Median filter, Wikipedia: <https://en.wikipedia.org/wiki/Median_filter>

[4] Số trung vị, Wikipedia: <https://vi.wikipedia.org/wiki/S%E1%BB%91_trung_v%E1%BB%8B>

[5]

## IX. TÀI LIỆU THAM KHẢO của nhóm kia

[5] 1-D median filtering – MATLAB medfilt1 – Mathworks, https://www.mathworks.com/help/signal/ref/medfilt1.html

[2] 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.

1. [1] Tần số âm cơ bản, Wikipedia https://vi.wikipedia.org/wiki/Tần\_số\_âm\_cơ\_bản
2. [2] Nguyen Binh Thien, Ninh Khanh Duy, Bài Báo Fair 2017

Prentice Hall - Digital Processing Of Speech Signals 1978

1. Abigail Lira, “Implementing a Pitch Detection Algorithm to Tune a Bass Guitar”, Technical report, El Paso Community College, 2015.
2. Cubic Spline Interpolation. Retrieved 2016-11-05, from https://en.wikiversity.org/wiki/Cubic\_Spline\_Interpolation.
3. 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.
4. 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.
5. 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.