УДК 004.021+534.781

МЕТОДЫ ПРЕДВАРИТЕЛЬНОГО АНАЛИЗА ГОЛОСОВОГО СИГНАЛА ДЛЯ ЗАДАЧ МЕДИЦИНСКОЙ ДИАГНОСТИКИ

Жолуд Е.И., магистрант гр.355701

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Вашкевич М.И. – д.т.н., доцент

Аннотация. *В работе рассматривается метод сегментации голосового сигнала на периоды основного тона для цельей последующего акустического анализа. Исследование фокусируется на процессе разделения голосового сигнала на отдельные периоды, соответствующие основным тональным компонентам, с использованием алгоритмов обработки сигналов - автокорреляции. Практическая реализация алгоритма включает подключение библиотек, загрузку и обработку аудиофайлов, вычисление автокорреляционной функции, а также фильтрацию сигнала и вывод результатов на экран.*

Ключевые слова. Акустический анализ, сегментация голосового сигнала, основной тон, автокорреляция.

Введение

Современные методы и алгоритмы акустического анализа голосового сигнала играют важную роль в системах медицинской диагностики. Они позволяют выявлять различные заболевания и состояния человека по его голосу. Методы акустического анализа голосового сигнала представляют собой мощный инструмент для нетравматичного обследования пациентов, обеспечивая более раннюю диагностику и эффективное лечение.

Анализ голосового сигнала позволяет выявить изменения в интонации, тембре, скорости речи и других параметрах, которые могут свидетельствовать о наличии определенных заболеваний, таких как болезни дыхательной системы, неврологические расстройства или психические состояния. Методы акустического анализа голоса становятся все более точными и надежными благодаря развитию вычислительных технологий и алгоритмов обработки сигналов.

Сегментация голосового сигнала на его на периоды основного тона

Если отобразить голосовой сигнал на временной диаграмме, то можно увидеть, что он представляет из себя практически периодическое колебание (рис. 1).

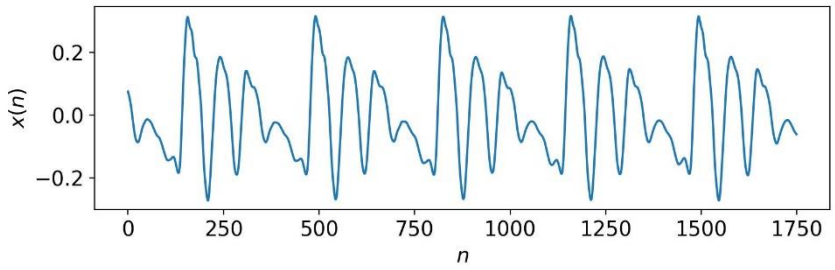

Рисунок 1 – Пример голосового сигнала

В действительности период сигнала не являются стабильными во времени, поэтому голосовой сигнал принято относить к квазипериодическим сигналам. Периодом основного тона принято считать минимальное значение T_0 для которого выполняется приблизительное равенство:

$$x(t) \approx x(t + T_0). \quad (1)$$

Величина обратная периоду основного тона называется частотой основного тона вычисляется по формуле (2):

$$F_0 = 1/T_0. \quad (2)$$

Сегментация голосового сигнала на периоды основного тона – это процесс разделения голосового сигнала на отдельные периоды, соответствующие частоте основного тона. Этот процесс может быть полезен при анализе речи, музыки или других звуковых сигналов.

Для сегментации голосового сигнала на периоды основного тона можно использовать алгоритмы обработки сигналов, такие как алгоритмы автокорреляции или методы, основанные на преобразовании Фурье. Эти методы позволяют определить периоды повторяющихся структур в сигнале, которые соответствуют основному тональному компоненту.

Практическая реализация сегментации сигнала

Первый шаг в реализации алгоритма сегментации голосового сигнала на его периоды основного тона - загрузка аудиофайла, проверка количества каналов и выбор канала.

Загрузка аудиофайла выполняется с помощью функции sf.read() из библиотеки soundfile.

```
# Загрузка аудиофайла
data, samplerate = sf.read('D:/ 010.wav', dtype='float32')
if len(data.shape) > 1:
    data = data[:, 0]  # Выбор первого канала
```

Результат загрузки аудиофайла сохраняется в переменные data и samplerate. Переменная data содержит аудиоданные, а переменная samplerate содержит частоту дискретизации аудиофайла.

Далее происходит проверка количества каналов в аудиоданных. Если len(data.shape) (размерность данных) больше 1, значит в данных есть несколько каналов (например, стерео), и необходимо выбрать только один канал для дальнейшей обработки.

Если в данных есть несколько каналов, то строка data = data[:, 0] выбирает только первый канал из всех доступных каналов. Таким образом, после выполнения этой строки переменная data будет содержать только данные из первого канала аудиофайла.

Второй шаг в реализации алгоритма сегментации голосового сигнала на его периоды основного тона - вычисление автокорреляционной функции.

Автокорреляционная функция (*aucorrelation function – ACF*) является мощным инструментом для анализа периодических компонентов в сигнале. Она позволяет оценить степень корреляции между сигналом и его запереженными копиями. В контексте поиска периодических компонентов, автокорреляционная функция помогает найти период сигнала. Если автокорреляционная функция имеет ярко выраженные пики при определенных значениях задержки, это может указывать на наличие периодичности или регулярной структуры в сигнале. Вычисляется автокорреляционная функция по формуле (3):

$$r_{xx}(k) = \frac{1}{N}\sum_{n=0}^{N-1} x\,[n] \times [n+k] \qquad (3)$$

где $r_{xx}(k)$ - значение автокорреляции для задержки k, N – длина сигнала, $x[n]$ - анализируемый сигнал.

Результат выполнения представлен на рис. 2.

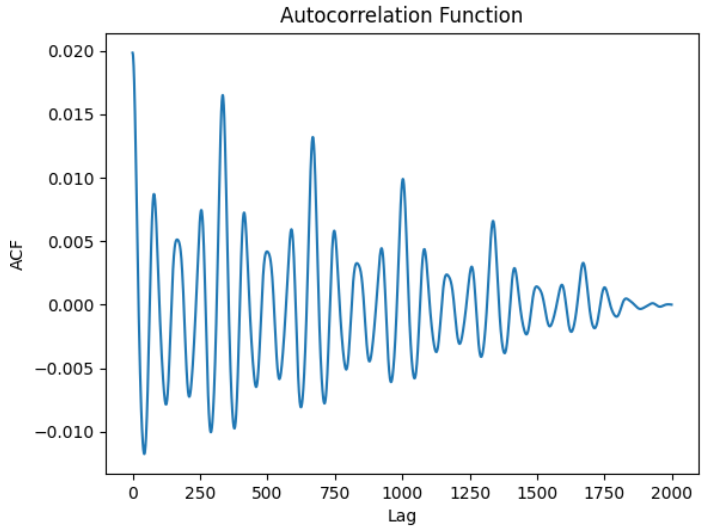

Рисунок 2 – Автокорреляционная функция

Третий шаг алгоритма – нахождение пика в автокорреляционной функции, соответствующего частоте основного тона. Период основного тона соответствует обратной частоте этого пика.

Для начала необходимо найти все локальные максимумы: find_peaks(r_xx) находит все локальные максимумы в массиве значений автокорреляции r_xx и возвращает их индексы в

переменной peaks. Далее найти максимальный пик в автокорреляционной функции. Из всех найденных пиков выбирается первый пик как начальное предположение о максимальном пике. Затем происходит итерация по остальным пикам, и если значение автокорреляции в текущем пике больше, чем значение автокорреляции в текущем максимальном пике, то этот пик становится новым максимальным. По завершении цикла получается индекс max_peak, соответствующий максимальному значению автокорреляции.

Чтобы найти пик в автокорреляционной функции, мы ищем максимальное значение этой функции. Пусть m_{peak} - лаг, соответствующий пику автокорреляционной функции. Пик автокорреляционной функции вычисляется по формуле (4):

$$m_{peak} = max_m\, r_{xx}(k) \tag{4}$$

Программная реализация выглядит следующим образом:

```
# Находим первый пик АКФ
peaks, _ = find_peaks(r_xx)
# Находим максимальный пик АКФ
max_peak = peaks[0]
for i in range(1, len(peaks)):
    if r_xx[peaks[i]] > r_xx[max_peak]:
        max_peak = peaks[i]
print(f"Max peak at index {max_peak}")
```

Результат выполнения представлен на рис. 3.

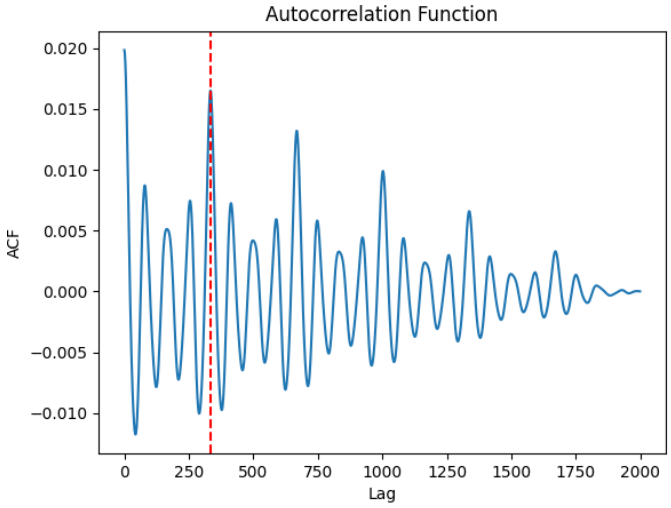

Рисунок 3 – Максимальный пик автокорреляционной функции

Четвертый шаг в реализации алгоритма сегментации голосового сигнала на его периоды основного тона - создание КИХ-фильтра.

КИХ-фильтр создается по следующему алгоритму:

1) Вычисляется частота дискретизации (*sampling rate*). Частота дискретизации (fs) определяется как частота, с которой сигнал или данные были дискретизированы. Частота дискретизации представляет собой скорость, с которой аналоговый сигнал преобразуется в цифровой формат путем выборки значений сигнала на определенных интервалах времени. Обозначается как fs и измеряется в герцах (Гц). Математически это можно представить формулой (5):

$$f_s = \frac{1}{T_s} \tag{5}$$

где T_s – период дискретизации.

2) Рассчитывается частота Найквиста

$$nyq = \frac{f_s}{2} \tag{6}$$

3) Задается основной период в отсчетах. Он представляет собой количество дискретных отсчетов между повторяющимися экземплярами сигнала и является ключевой временной характеристикой периодического сигнала и позволяет определить основной период повторения сигнала. В контексте сигналов и систем фундаментальный период в отсчетах помогает определить частоту основной составляющей сигнала. Фундаментальный период (T0) задается как максимальный пик, найденный в автокорреляционной функции ранее. В аудио обработке, фундаментальный период обычно указывает на частоту основной составляющей звукового сигнала (формула (7)).

$$T_0 = m_{peak} \tag{7}$$

4) Основной период T_0 пересчитывается в секунды для того, чтобы иметь представление о периоде в удобном для интерпретации временном формате. Для этого необходимо воспользоваться формулой (8):

$$T_0 = \frac{T_0}{f_s} \tag{8}$$

5) Вычисляется основная частота (F0) как обратная величина основного периода T0. Основная частота — это частота основной гармоники периодического сигнала, соответствующая его основному периоду:

$$F_0 = \frac{1}{T_0} \tag{9}$$

6) Вычисляется нормализованная частота среза фильтра (cutoff). Она позволяет удобно задавать параметры фильтров, не зависимо от конкретной частоты дискретизации и позволяет установить, какие частоты будут подавлены или пропущены фильтром, используя относительные показатели в виде нормализованных частот:

$$cutoff = \frac{1{,}5\ F_0}{F_s} \tag{10}$$

7) Определяется длина фильтра определяет количество коэффициентов фильтра, которые будут использоваться для его создания (N_filter). Чаще всего это число выбирается на основе эмпирических знаний о конкретной задаче или настройках фильтра. В данном случае оно равно 201.

8) Создается КИХ-фильтр с помощью функции (firwin) с параметрами N_filter, cutoff и fs.

Результатом этой операции являются коэффициенты фильтра, которые будут использоваться для фильтрации входных данных.

Программная реализация выглядит следующим образом:

```
# Задаем параметры для КИХ-фильтра
fs = samplerate # Вычисляется частота дискретизации (sampling frequency) fs и выводится
print(f"Sampling frequency (rate) is fs = {fs} Hz")
nyq = 0.5*fs    # Рассчитывается частота Найквиста
print(f"Nyqust frequency is Fn = {nyq} Hz")
T0 = max_peak   # Задается фундаментальный период T0 в отсчетах
print(f"Fundamental period is T0 = {T0} samples")
T0 = T0/fs      # Фундаментальный период T0 пересчитывается в секунды
print(f"Fundamental period is T0 = {T0} seconds")
F0 = 1.0/(T0)   # Вычисляется фундаментальная частота F0
print(f"The fundamental frequency is F0 = {F0} Hz")
cutoff = 1.5*F0/nyq # Вычисляется нормализованная частота среза фильтра
print(f"Filter cutoff normilized frequency is fc={cutoff}")  # Частота среза фильтра в Гц
# N_filter = int(fs / F0)  # Длина фильтра
N_filter = 201  # Определяется длина фильтра N_filter
print(f"Filter length is N_filter = {N_filter}")

# Создание КИХ-фильтра
taps = firwin(N_filter, cutoff, fs=fs) # Создается FIR фильтр с помощью функции firwin с параметрами N_filter, cutoff и fs
```

Результат выполнения представлен на рис. 4.

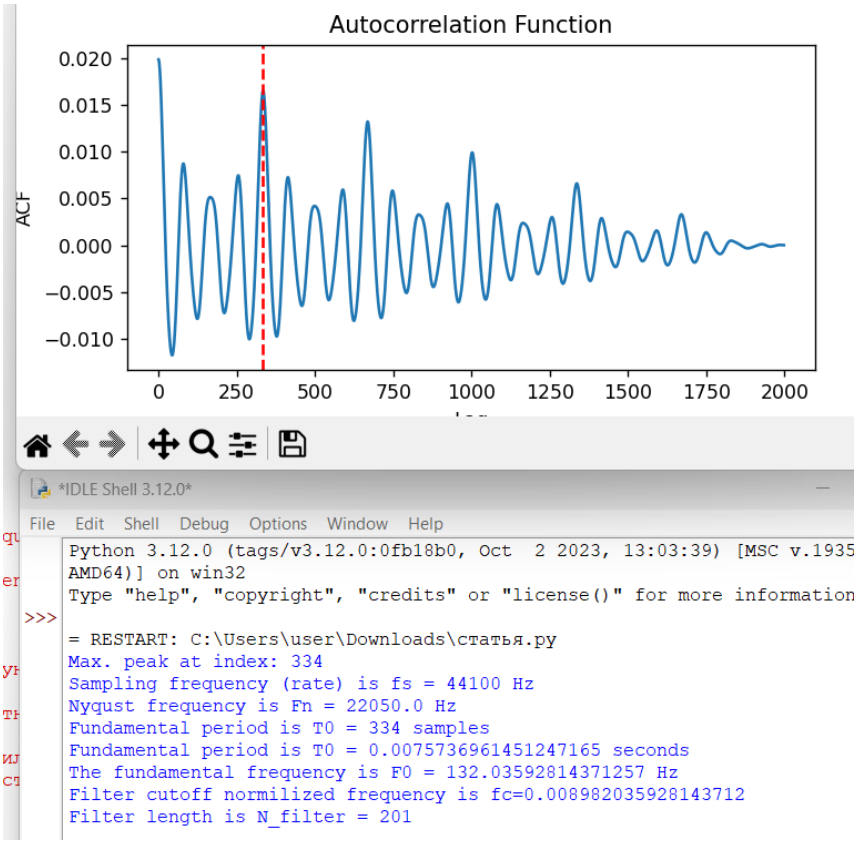

Рисунок 4 – Характеристики фильтра для обработки голоса

Пятый шаг в реализации алгоритма сегментации голосового сигнала на его периоды основного тона - фильтрация входного сигнала с помощью КИХ-фильтра.

В данном случае используется линейная свертка сигнала с коэффициентами фильтра для получения отфильтрованного сигнала. Математическое описание: пусть data - входной сигнал, taps - коэффициенты фильтра, filtered_signal - отфильтрованный сигнал. np.convolve(data, taps, mode='full') производит линейную свертку входного сигнала с коэффициентами фильтра. filtered_signal[(N_filter-1)//2:] извлекает правильную часть отфильтрованного сигнала, учитывая задержку свертки.

Программная реализация выглядит следующим образом:

```
# Фильтрация сигнала
filtered_signal = np.convolve(data, taps, mode='full') # Фильтрация
filtered_signal = filtered_signal[(N_filter-1)//2:]
```

Результат выполнения представлен на рис. 5.

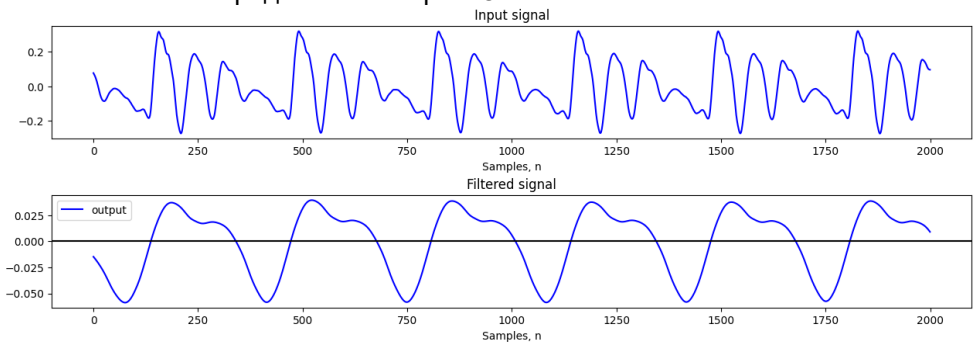

Рисунок 5 – Фильтрация сигнала

Шестой шаг в реализации алгоритма сегментации голосового сигнала на его периоды основного тона – нахождение всех пиков в отфильтрованном сигнале и вывод их на экран.

Поиск пиков позволяет выделить значимые точки изменения сигнала: максимальные, положительные и отрицательные пики могут быть важны для дальнейшего анализа и интерпретации данных, что помогает отслеживать изменения в сигнале и их влияние на поведение системы.

Процесс поиска пиков в сигнале заключается в нахождении локальных максимумов (положительных или отрицательных) в сигнале с определенными параметрами выделения. Программно максимальные пики в отфильтрованном сигнале находятся с использованием функции find_peaks, учитывая высоту (prominence) пика и значения максимальных пиков сохраняются. Также для проверки правильности работы кода можно выводить позиции и значения максимальных пиков в сигнале. Таким же образом ищутся и выводятся данные положительных и отрицательных пиков.

Программная реализация выглядит следующим образом:

```
max_peaks, _ = find_peaks(filtered_signal[:2000], prominence= 0.1*np.max(filtered_signal[:2000]))
max_peak_values = filtered_signal[max_peaks][:2000]
print(f"Max peaks at indices {max_peaks}")
print(f"Max peak values: {max_peak_values}")

# Находим все пики в отфильтрованном сигнале
positive_peaks, _ = find_peaks(filtered_signal[:2000],
prominence=0.1*np.max(filtered_signal[:2000]))
negative_peaks, _ = find_peaks(-filtered_signal[:2000],
prominence=0.1*np.max(filtered_signal[:2000]))
print(f"Positive peaks at indices: {positive_peaks}")
print(f"Negative peaks at indices: {negative_peaks}")
# Находим значения пиков
positive_peaks_values = filtered_signal[positive_peaks]
negative_peaks_values = filtered_signal[negative_peaks]
print(f"Values of positive peaks: {positive_peaks_values}")
print(f"Values of negative peaks: {negative_peaks_values}")
```

Результат выполнения представлен на рис. 6.

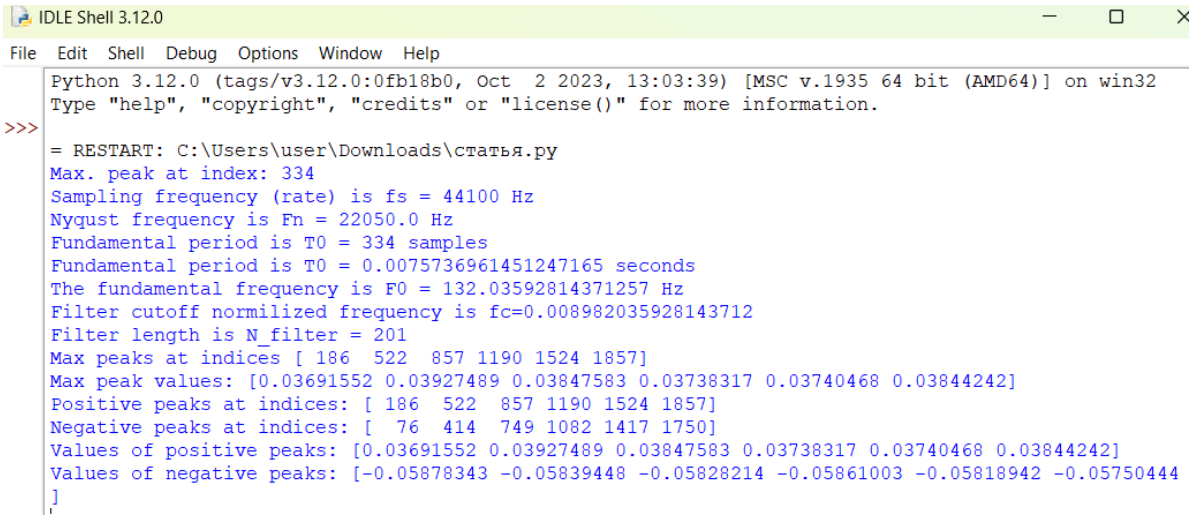

Рисунок 6 – Нахождение пиков в отфильтрованном голосовом сигнале

Седьмой шаг в реализации алгоритма - нахождение точек перехода функции из отрицательного значения в положительные. Это необходимо для дальнейшего определения значения периода.

Данный шаг позволяет выявить ключевые моменты изменения поведения сигнала, когда он пересекает нулевую линию или меняет знак. Нахождение точек пересечения или изменения знака сигнала может быть важным для дальнейшего анализа и выделения интересующих участков сигнала.

Программная реализация выглядит следующим образом:

```
# Находим точки, где график пересекает ось X в точке 0
zero_crossings = np.where(np.diff(np.sign(filtered_signal[:2000])))[0]

# Находим точки перехода из «-» в «+», которые находятся слева от максимальных пиков
positive_zero_crossings_left = []
for peak in positive_peaks:
    crossings = zero_crossings[zero_crossings < peak]
    if len(crossings) > 0:
        last_crossing = crossings[-1]
        positive_zero_crossings_left.append(last_crossing)
print(f"Positive zero crossings on the left: {positive_zero_crossings_left}")
```

```
negative_to_positive_crossings_right = []
for crossing in positive_zero_crossings_left:
    next_negative_peak_idx = np.argmax(negative_peaks > crossing)
    if next_negative_peak_idx < len(negative_peaks):
        negative_to_positive_crossings_right.append(negative_peaks[next_negative_peak_idx])

negative_to_positive_peaks_values = filtered_signal[negative_to_positive_crossings_right]
```

Результат выполнения представлен на рис. 7.

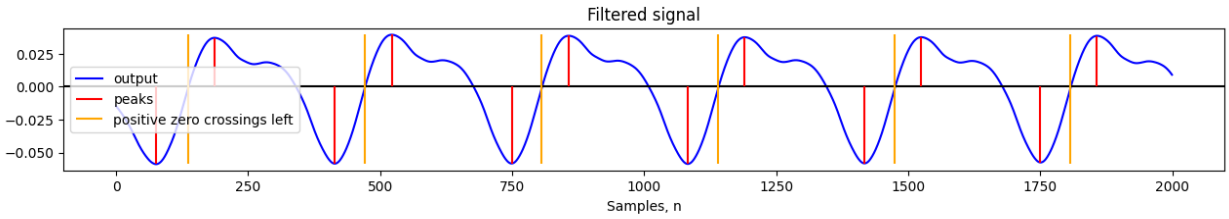

Рисунок 7 – Нахождение границ периодов

Восьмой шаг в реализации алгоритма - вычисление значений периодов.

Для этого в python есть np.diff(positive_zero_crossings_left). Эта функция diff из библиотеки NumPy используется для вычисления разницы между соседними элементами массива. В данном случае, positive_zero_crossings_left - это массив, и np.diff() вычисляет разность между каждой парой соседних элементов этого массива. Результат сохраняется в переменной periods.

Программная реализация выглядит следующим образом:

```
periods = np.diff(positive_zero_crossings_left)
print(f"Periods: {periods}")
```

Вывод.

В работе описана программная реализация процедуры сегментации голосового сигнала на периоды основного тона для целей последующего акустического анализа голоса.

Список использованных источников:

1. Кривошеев В.И. Современные методы цифровой обработки сигналов (цифровой спектральный анализ) / В.И. Кривошеев // Учебно-методические материалы по программе повышения квалификации «Современные системы мобильной цифровой связи, проблемы помехозащищенности и защиты информации» - 2006 г. -117 стр.

2. Первичный анализ речевых сигналов [Электронный ресурс] – Режим доступа: https://alphacephei.com/ru/lecture1.pdf – Дата доступа 11.04.2024.

3. ДПФ Лекции по курсу "Дискретные преобразования сигналов" 7 мая 2019 ФРТК МФТИ, 2019 [Электронный ресурс] – Режим доступа: https://kprf.mipt.ru/attachments/article/66/слайды_лекции_7_мая_2019.pdf – Дата доступа 11.04.2024.

4. Вашкевич М.И., Лихачёв Д.С., Азаров И.С. Система анализа и классификации голосовго сигнала на основе пертурбационных параметров и кепстрального представления в психоакустических шкалах [Электронный ресурс] – Режим доступа: https://libeldoc.bsuir.by/bitstream/123456789/46764/1/Vashkevich_Metodika.pdf – Дата доступа 11.04.2024.

UDC 004.021+534.781

METHODS AND ALGORITHMS OF ACOUSTIC ANALYSIS OF VOICE SIGNAL FOR MEDICAL DIAGNOSTIC SYSTEMS

Zholud L.I.

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Vashkevich M.I. – PhD in Technology

Annotation. This paper discusses a method for segmenting voice signals into periods of fundamental frequency for acoustic analysis. The research focuses on the process of dividing the voice signal into individual periods corresponding to fundamental tonal components using signal processing algorithms - autocorrelation. The practical implementation of the algorithm includes connecting libraries, loading and processing audio files, calculating the autocorrelation function, as well as signal filtration and displaying results on the screen.

Keywords. Acoustic analysis, voice signal segmentation, fundamental frequency, autocorrelation.