МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ ИМЕНИ ПАТРИСА ЛУМУМБЫ»

Факультет/учебный институт
_Физико-математических и естественных наук
Кафедра/учебный департамент Математический институт им. С. М. Никольского

«Допустить к защите»

Заведующий кафедрой/директор учебного департамента
Мат. институт им. С. М. Никольского
(название)
Муравник А.Б.
(Ф.И.О.)
«___» _____2024 г.

Выпускная квалификационная работа бакалавра

Направление/Специальность 01.03.02 «Прикладная математика и информатика»
(шифр направления/специальности) (наименование направления/специальности)

ТЕМА Разработка алгоритма для автоматической генерации мелодий на основе музыкальной гармонии

Выполнил студент Понкратова Христина Анатольевна
(Фамилия, имя, отчество)

Руководитель выпускной
квалификационной работы
Карандашев Я.М. доцент, к.ф.-м.н.__
(Ф.И.О., степень, звание, должность)

(подпись)

Автор_____
(подпись)

Группа НПМбд-01-20

Студ. билет №1032202476

г. Москва
2024г.

Федеральное государственное автономное образовательное учреждение высшего образования

“Российский университет дружбы народов имени Патриса Лумумбы”

АННОТАЦИЯ

выпускной квалификационной работы

Понкратовой Христины Анатольевны

на тему: **“Разработка алгоритма для автоматической генерации мелодий на основе музыкальной гармонии”**

Выпускная квалификационная работа посвящена исследованию и анализу алгоритмов генерации музыкальных композиций, имеющим большой потенциал для развития и применения в различных областях, таких как компьютерные игры, фильмы, музыкальная индустрия и искусство, а также разработке собственного программного решения.

Цель работы — создание посредством компьютерных технологий инструмента, который мог бы помочь композиторам и музыкантам в процессе создания новых музыкальных произведений за счет автоматизации процесса генерации мелодий.

В первой главе рассмотрены ключевые аспекты, касающиеся связи музыки и математики, а также основные понят

ия и определения, необходимые для понимания музыкальной теории и ее математической интерпретации.

Вторая глава представляет собой аналитический обзор математических методов, используемых для генерации музыкальных композиций, а также основные понятия и области применения звуковых волн.

Третья глава посвящена разработке алгоритма для автоматической генерации музыкальных мелодий. В результате работы были успешно сгенерированы музыкальные мелодии, которые демонстрируют эффективность и разнообразие разработанного алгоритма, подтверждая его способность создавать логичные и гармоничные музыкальные произведения.

Автор ВКР _____ ***Х.А. Понкратова***

(подпись)

Содержание

ВВЕДЕНИЕ..... 4
ГЛАВА 1. ТЕОРИЯ МУЗЫКИ И МАТЕМАТИЧЕСКИЕ МЕТОДЫ..... 5
1.1 Основные понятия и определения.....5
1.2 Как устроена музыкальная гармония. Пространство кратностей.....10
1.3 Музыкальные термины.....12
1.4 Выводы первой главы.....14
ГЛАВА 2. АНАЛИЗ МАТЕМАТИЧЕСКИХ МЕТОДОВ И МОДЕЛЕЙ ДЛЯ ГЕНЕРАЦИИ МУЗЫКАЛЬНЫХ КОМПОЗИЦИЙ..... 16
2.1 Методы генерации музыкальных композиций.....16
2.2 Звуковые волны: основные понятия и области применения..... 18
Виды звуковых волн..... 18
2.3 Математические модели основных типов звуковых волн (сигналов).. 20
2.3.1 Синусоидальная волна..... 20
2.3.2 Пилообразная волна..... 22
2.3.3 Квадратная волна.....24
2.3.4 Треугольная волна..... 26
2.4 Выводы второй главы.....30
ГЛАВА 3. РАЗРАБОТКА АЛГОРИТМА ДЛЯ АВТОМАТИЧЕСКОЙ ГЕНЕРАЦИИ МЕЛОДИЙ.....31
3.1 Описание алгоритма.....31
3.2 Описание алгоритма музыкальным языком.....33
Алгоритм написания песни..... 35
3.3 Используемые библиотеки, оборудование и инструменты разработки..... 37
3.4 Разработка и тестирование алгоритма.....39
3.5 Выводы третьей главы.....46
ЗАКЛЮЧЕНИЕ.....47
СПИСОК ЛИТЕРАТУРЫ И ИНТЕРНЕТ-ИСТОЧНИКОВ..... 49
ПРИЛОЖЕНИЕ..... 51

ВВЕДЕНИЕ

В музыке **гармония** — это концепция объединения различных звуков вместе для создания новых, отчетливых музыкальных идей. Другими словами, музыкальная гармония — это состояние, когда звуки в музыке звучат вместе и создают приятное и цельное звучание, которое может быть достигнуто через сочетание различных нот и аккордов, создающих музыкальную цельность и красоту, и играет важную роль, определяя тон и настроение музыкального произведения.

Исследование и анализ алгоритмов генерации музыкальных композиций имеют большой потенциал для развития и применения в различных областях, таких как компьютерные игры, фильмы, музыкальная индустрия и искусство.

Цель работы — создание посредством компьютерных технологий инструмента, который мог бы помочь композиторам и музыкантам в процессе создания новых музыкальных произведений за счет автоматизации процесса генерации мелодий.

Для достижения поставленной цели необходимо решить следующие **задачи:**

- выполнить анализ литературы и интернет-источников по теме ВКР;
- провести анализ существующих методов генерации мелодий;
- разработать инструменты для создания алгоритма для автоматической генерации мелодий;
- выполнить тестирование разрабатываемых решений;
- осуществить генерацию мелодий на основе гармонии;
- осуществить итоговое тестирование и оценку результатов;
- доработать и оптимизировать алгоритм.

ГЛАВА 1. ТЕОРИЯ МУЗЫКИ И МАТЕМАТИЧЕСКИЕ МЕТОДЫ

1.1 Основные понятия и определения

Связь музыки и математики (математической логики) подробно разбирает в своей работе [1] отечественный исследователь М.В. Андрейкина, где она прослеживает трансформацию этого феномена, разбирая работы великих мыслителей прошлого от учения Пифагора к теории А.Ф. Лосева, согласно которому: *«**Музыка** — это своеобразный феномен, идеальная субстанция, она есть время, в котором существует число, которое пронизывает Вселенную и бытие человека»*.

Сегодня развитие технологий искусственного интеллекта, появление новых алгоритмов машинного обучения и анализа данных, нейронных сетей и др. перспективных IT-технологий открывает широкие возможности для автоматизации генерации мелодий на основе музыкальной гармонии программными методами. Больше других внимания этой проблеме уделил другой отечественный ученый Н.А. Никитин [2-4], идеи которого нашли свое отражение в данной выпускной квалификационной работе (ВКР).

Далее будут приведены основные понятия и определения из теории музыки [5-8], необходимые для реализации задач ВКР.

Мелодия — это основной элемент музыкальной композиции, представляющий собой последовательность звуков, звучащих в определенной расстановке и ритме. Мелодия может быть тем, что мы поем или играем на музыкальном инструменте, она состоит из нот (рис. 1.1), звучащих в некоторой последовательности, определенной длительности (рис. 1.2).

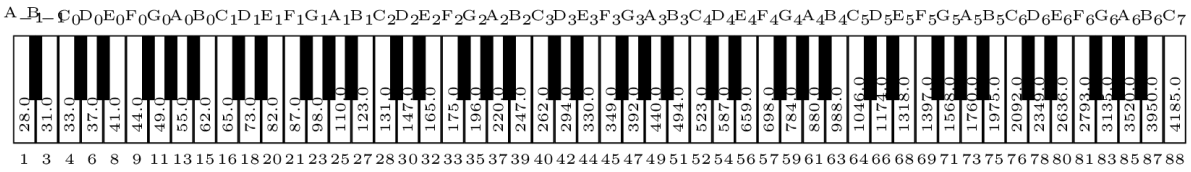


Рис. 1.1 Ноты в Герцах

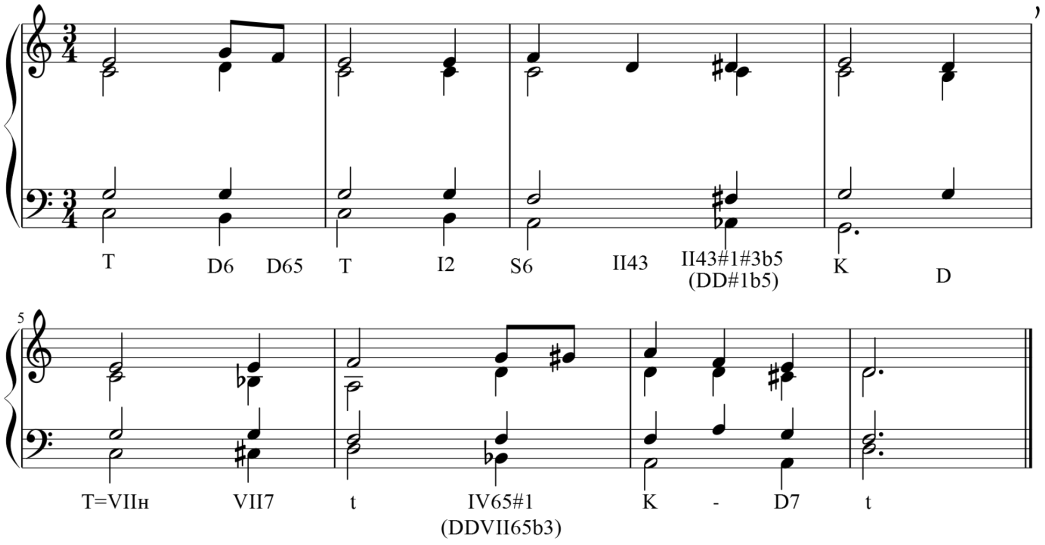


Рис. 1.2 Пример музыкальной мелодии, написанной на языке нотной грамоты

Характер звука определяется четырьмя основными свойствами, которые приведены в табл. 1.1.

Высота звука зависит от частоты источника звука. Слышимые человеком звуковые частоты имеют диапазон примерно от 20 Гц до 20 кГц.

Громкость в свою очередь зависит от размаха колебательных движений источника звука.

Длительность — от продолжительности колебаний источника.

Тембром же называется качественная характеристика звука, т. е. его окраска. Именно благодаря этой характеристике мы различаем огромное количество музыкальных инструментов, голоса и даже шумовые звуки.

Табл. 1.1 Четыре основных свойства звука, которые определяют его характер

№	Наименование параметра	Размерность (ед. измерения)	Комментарий
1	Высота	Высота звука, измеряется в герцах (Гц, Hz) 1 Гц = 1/Секунда	Зависит от частоты колебания звуковой волны
2	Громкость	Громкость выражается в децибелах (дБ).	Зависит от амплитуды колебания. Чем выше амплитуда, тем громче звук
3	Тембр	Безразмерный параметр	Окраска голоса, делающая его индивидуальным
4	Длительность	Длительности могут быть как относительными (выражают ритмические соотношения), так и абсолютными [Секунда]	Продолжительность колебаний

Кроме того, у звука существуют такие характеристики, как “форма звучания”, “основной тон” и “обертоны” (рис. 1.3).

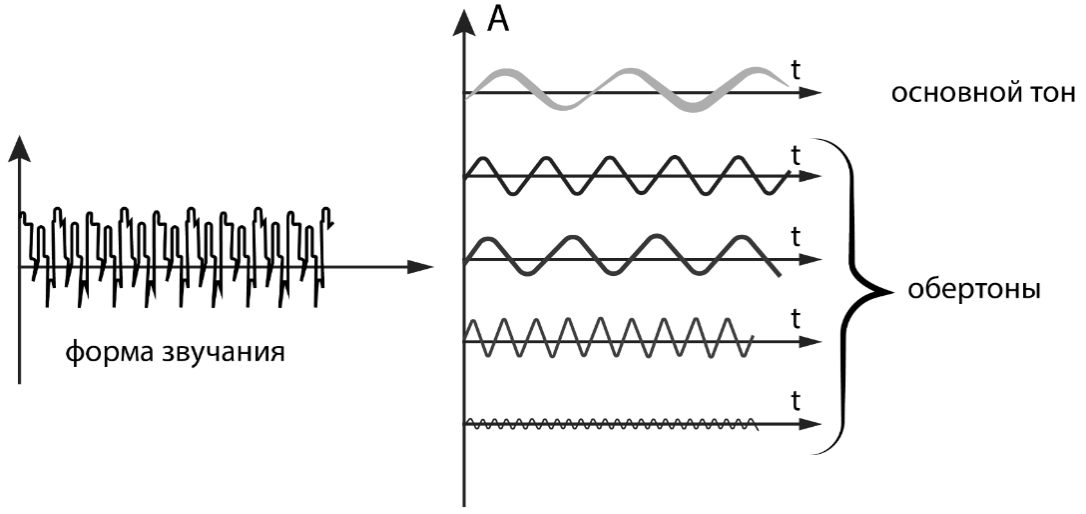


Рис. 1.3 Характеристики звука

Тембр звука зависит от наличия в нем "частичных" тонов или, иначе говоря, обертонов (**гармоник**). Гармоники по частоте всегда выше основного тона и выражаются рядом простых чисел:

1, 2, 3, 7, 11, 13, 17 и т. д.,

если за единицу измерения принять частоту основного тона. Существует визуализация пространства кратностей, в котором близлежащие тона всегда образуют консонанс (рис. 1.4).

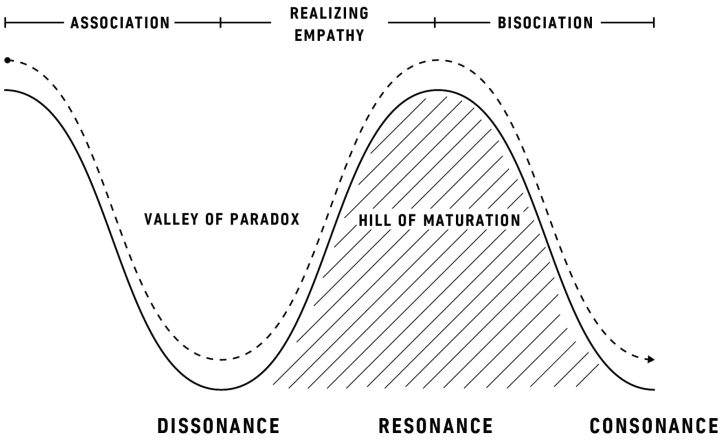


Рис. 1.4 Консонанс

Базовые звуки (существующие ноты) образуют консонанс при кратностях 3 и 5.

Октава — это диапазон колебаний звука, эквивалентом которого является изменение частоты колебаний в два раза.

Говоря о частоте звука здесь и далее будем иметь в виду частоту f_1 основного тона. Однако звуковые колебания содержат также гармоники, частоты f_n, которых отвечают соотношению:

$$f_n = n f_1, \ (n = 1, 2, 3, \ldots). \qquad \textbf{(1)}$$

Амплитуды A_n этих волн имеют зависимость от способа возбуждения музыкальной волны. Эти амплитуды необходимы для придания музыкальной окраски звуку (тембру).

Частоты музыкальных нот следуют логарифмической шкале (двенадцатитоновый темперированный строй). Основная формула для вычисления частоты нот:

$$f_n = f_0 \times (a)^n, \quad \textbf{(2)}$$

где f_n - частота n-й ноты, f_0 - частота опорной ноты, a - корень двенадцатой степени из двух, n - количество полутонов между опорной нотой и искомой нотой.

Многие звуки, которые мы слышим в музыке, можно аппроксимировать с помощью тригонометрических функций, таких как синусы и косинусы. Это позволяет представить музыкальные звуки как сумму различных гармонических компонент, каждая из которых имеет свой собственный частотный спектр.

Использование тригонометрического ряда Фурье [7] в музыке позволяет анализировать звучание инструментов, создавать эффекты и фильтры звука, а также синтезировать новые аудиоэффекты и инструменты. Тригонометрическим рядом Фурье функции $f \in L_2([-\pi, \pi])$ называют функциональный ряд вида:

$$f(x) = \frac{a_0}{2} + \sum_{n-1}^{\infty} (a_n cos\,(nx) + b_n sin\,(nx), \quad \textbf{(3)}$$

где

$$a_0 = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x)\,dx, \quad \textbf{(4)}$$

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\, cos\,(nx)\, dx, \quad \textbf{(5)}$$

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\, sin\,(nx)\, dx. \quad \textbf{(6)}$$

Числа a_0, a_n, b_n ($n = 1, 2, 3, \ldots$) называются коэффициентами Фурье функции f.

Таким образом, тригонометрический ряд Фурье играет важную роль в создании и обработке музыки, помогая музыкантам и звукорежиссерам достичь желаемых звуковых эффектов и улучшить качество звучания музыкальных произведений.

1.2 Как устроена музыкальная гармония. Пространство кратностей

Большое внимание вопросам о том, как устроена музыкальная гармония уделил математик и музыкальный эксперт Роман Олейников [7].

Музыкальная гармония — это основа музыкальной композиции, которая определяет звуковую структуру музыкального произведения. Гармония включает в себя сочетание звуков, их взаимоотношения и взаимодействие.

Исследования в области музыкальной гармонии [8] показывают, что она важна не только для создания приятного звучания, но и для передачи эмоций и настроения слушателя. Гармония определяет уровень напряженности и разрешения в музыкальной композиции, создавая плавные переходы и красивые мелодии.

Гармоники возникают из-за того, что большинство реальных звуковых источников не создают идеальные синусоидальные волны. Например, при игре на струне или в воздушной колонне возникает

основной тон, а также множество гармоник, которые являются результатом сложного поведения системы. Каждая гармоника добавляет к основной волне, изменяя её форму и создавая уникальный звуковой отпечаток.

Микрохроматика — особый род интервальной системы музыки. Её выделил и описал известный российский музыкант-теоретик и выдающийся музыковед Юрий Холопов. Ключевым понятием микрохроматики является микроинтервал, то есть интервал, размер которого составляет менее полутона (рис. 1.5).

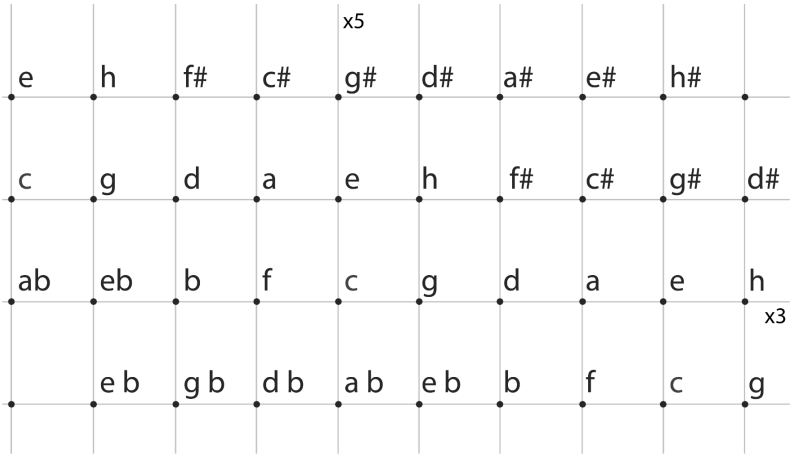


Рис. 1.5 Характеристики звука (микрохроматика)

Так, существуют микроинтервалы четвертитон, трететон, шестинатон и т. д. Примечательно, что они являются стабильными элементами звуковой системы (рис. 1.6).

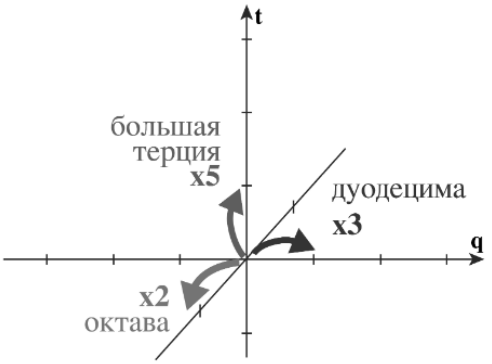


Рис. 1.6 Способ составления пространства кратностей

Только вот необученный слух практически не способен их различить, поэтому воспринимает как фальшь или негармоничное изменение структуры лада.

Чтобы лучше понять и использовать гармоники в рамках ВКР мы попробуем создать функцию, которая будет генерировать синусоидальную волну и добавлять к ней указанное количество гармоник. Эта функция будет принимать параметры, такие как основная частота, амплитуда, количество гармоник, длительность сигнала и частота дискретизации. с гармониками, помогая лучше понять, как гармоники влияют на звук.

1.3 Музыкальные термины

Сольфеджио – это дисциплина, направленная на изучение нотной грамоты, интонирования и ритма. В контексте данной работы, основы сольфеджио обеспечивают теоретическую базу для алгоритмов, которые анализируют и генерируют музыкальные структуры, помогая создавать гармоничные и ритмически корректные мелодии.

В процессе разработки алгоритма необходимо четко определить ряд ключевых понятий, которые будут последовательно описаны ниже, чтобы обеспечить необходимое понимание.

Нота представляет собой базовую единицу музыкального языка, обозначающую конкретную звуковую высоту. Каждая нота определяется своей частотой и может быть записана на нотоносце.

Длительность ноты указывает на продолжительность звучания конкретной ноты в музыкальной композиции. Длительности бывают различными (целая, половинная, четвертная и т.д.) и обозначаются специальными символами.

Ритмический рисунок мелодии описывает последовательность длительностей нот и пауз, из которых состоит мелодия. Это соотношение ритмических единиц создаёт определённый ритм и пульсацию в музыкальном произведении.

Мелодия – это последовательность нот, организованная по высоте и ритму таким образом, что образует осмысленное и эстетически завершенное музыкальное высказывание. Мелодия является основным носителем музыкальной идеи и эмоционального содержания произведения.

Аккомпанемент – это музыкальное сопровождение мелодии, обычно состоящее из гармонических и ритмических элементов, которые поддерживают и обогащают основную мелодическую линию.

Темп определяет скорость исполнения музыкального произведения и измеряется в ударах в минуту (BPM). Темп влияет на общее восприятие композиции и её динамическую характеристику.

Аккорд представляет собой сочетание нескольких нот, звучащих одновременно и образующих гармоническое единство. Аккорды являются основой гармонии в музыке.

Трезвучие – это специфический вид аккорда, состоящий из трёх нот, которые расположены по терциям. Трезвучия могут быть мажорными или минорными и являются фундаментальными гармоническими элементами.

Тоника – это главная нота или аккорд в тональности, к которому стремится музыкальная фраза. Тоника является отправной точкой и конечной целью многих мелодических и гармонических движений.

Тональность обозначает систему взаимоотношений между нотами и аккордами, организованными вокруг тоники. Тональность задаёт основную гармоническую структуру произведения и может быть мажорной или минорной.

Мажор и **минор** – это две основные модальности, характеризующиеся различными интервалами между нотами. Мажорная тональность обычно ассоциируется с яркими и жизнерадостными эмоциями, в то время как минорная – с более тёмными и меланхолическими настроениями.

Введение и рассмотрение этих понятий позволит сформировать основу для разработки алгоритма.

1.4 Выводы первой главы

В первой главе рассмотрены ключевые аспекты, касающиеся связи музыки и математики, а также основные понятия и определения, необходимые для понимания музыкальной теории и ее математической интерпретации.

Современные достижения в области искусственного интеллекта, машинного обучения и анализа данных открывают новые возможности для автоматизации музыкальной композиции. Вклад Н.А. Никитина в эту область подчеркивает перспективность использования IT-технологий для генерации мелодий на основе музыкальной гармонии.

Были детально рассмотрены основные музыкальные понятия, такие как нота, длительность ноты, ритмический рисунок мелодии, мелодия, аккомпанемент, темп, аккорд, трезвучие, тоника, тональность, мажор и минор. Эти понятия составляют основу музыкальной теории и являются фундаментальными для разработки алгоритма анализа и синтеза музыкальных произведений.

Определение и понимание данных понятий позволяют сформировать прочную теоретическую базу для дальнейшего развития алгоритмов, способных эффективно обрабатывать музыкальные данные, создавая гармоничные и эстетически привлекательные музыкальные произведения.

Таким образом, первая глава заложила основу для последующих исследований и разработок, направленных на интеграцию музыкальной теории с математическими методами и технологиями искусственного интеллекта, что позволяет двигаться к автоматизированному созданию музыки с помощью программных средств.

ГЛАВА 2. АНАЛИЗ МАТЕМАТИЧЕСКИХ МЕТОДОВ И МОДЕЛЕЙ ДЛЯ ГЕНЕРАЦИИ МУЗЫКАЛЬНЫХ КОМПОЗИЦИЙ

2.1 Методы генерации музыкальных композиций

Исследование алгоритмов генерации музыкальных композиций актуально в области компьютерной музыки и искусственного интеллекта. Этот процесс включает создание программ для автоматической генерации музыки по заданным правилам.

В разработке алгоритма генерации мелодий планируется использовать собственный метод, основанный на личных знаниях и опыте написания песен. Однако анализ других методов все равно необходим для понимания текущих подходов и их преимуществ и ограничений.

Одним из основных подходов является использование генетических алгоритмов, позволяющих комбинировать и эволюционировать музыкальные фрагменты. Другие методы включают алгоритмы на основе нейронных сетей и обработки больших данных.

Скрытые марковские модели (СММ) предоставляют математический аппарат для моделирования последовательностей музыкальных данных. Сети Петри, как математическая модель, могут описывать и анализировать музыкальные композиции.

Искусственные нейронные сети (ИНС) обучаются на примерах и могут генерировать музыку, имитирующую стили и жанры. Системы Линденмайера, или L-системы, применимы для создания музыкальных структур на основе алгоритмических правил.

Генетические алгоритмы основаны на процессах естественного отбора и генетической эволюции, позволяя создавать новые музыкальные идеи и структуры. Клеточные автоматы, в свою очередь, моделируют

динамические системы по определенным правилам, предлагая уникальный подход к генерации музыкального материала.

Эти методы открывают новые возможности в исследовании и создании музыки, способствуя разнообразию и инновациям в музыкальной композиции.

Основные проблемы, с которыми сталкиваются исследователи в области генерации музыки, включают в себя сложность определения критериев качества музыкальной композиции, необходимость учета стилевых особенностей различных жанров музыки, а также необходимость обеспечения разнообразия и оригинальности созданных композиций.

Подробный обзор основных математических методов для генерации музыкальных композиций приведет в работе [2]. Помимо методов, озвученных выше, существует еще ряд моделей, к которым, в частности, относятся Скрытые марковские модели**,** Сети Петри, Клеточные автоматы, Системы Линденмайера (L-системы) и генетические алгоритмы.

Генетические алгоритмы (рис. 2.1) — это методы поиска и оптимизации, вдохновленные процессами естественного отбора и генетической эволюции [9]. В музыкальной композиции они могут использоваться для создания новых музыкальных идей и структур путем итеративного процесса отбора, кроссовера и мутации.

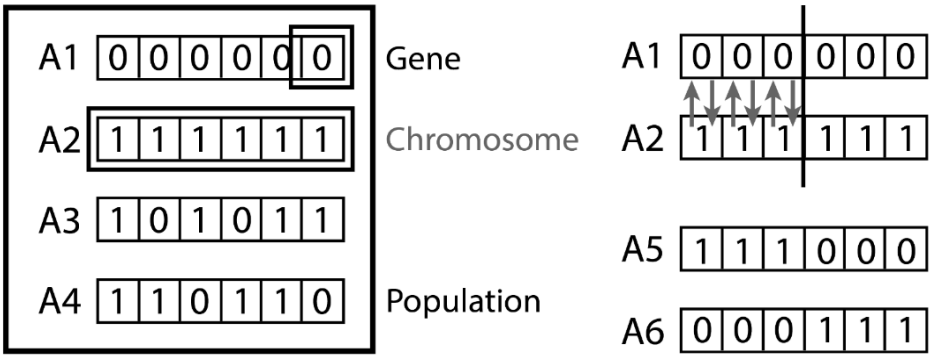


Рис. 2.1 Пример генетического алгоритма

Тем не менее, озвученные выше алгоритмы в силу своей природы не всегда могут создавать музыкальные мелодии, учитывающие классические правила теории музыки, поэтому в данной ВКР была поставлена задача формализовать основные правила музыкальной гармонии и на их основе разработать собственный алгоритм для автоматической генерации мелодий.

В дальнейшем, для решения данной задачи предполагается использовать искусственные нейронные сети (ИНС), которые способны обучаться на примерах и генерировать музыку, которая имитирует стили и жанры, на которых они были обучены.

2.2 Звуковые волны: основные понятия и области применения

Для того, чтобы перейти к непосредственной генерации мелодий необходимо более подробно остановиться на способах математического моделирования звуковой волны.

Звуковые волны — это колебания давления, которые распространяются через среду, такую как воздух, вода или твердые материалы. Эти колебания возникают в результате движений, например, вибрации струн гитары или колебания голосовых связок.

Виды звуковых волн

- ***Синусоидальная*** — простейшая форма волны, используемая для описания чистых тонов.
- ***Пилообразная*** — волна, которая звучит более резко и используется в синтезаторах.

- ***Квадратная*** — характеризуется резкими переходами между уровнями амплитуды, часто используется в электронике и цифровых синтезаторах.
- ***Треугольная*** — содержит гармоники и имеет более мягкое звучание по сравнению с квадратной волной.

Форма звуковой волны сильно влияет на характер и качество звука. Например, синусоидальные волны создают чистые тона, а пилообразные и квадратные волны добавляют более богатые обертоны, что делает их популярными в электронных музыкальных жанрах.

Рассмотрим еще несколько важное понятий: “**Интерференция**”, “**Резонанс**” и “**Гармоники**”.

Интерференция происходит, когда две или более звуковых волны пересекаются, усиливая или ослабляя друг друга. Она может быть:

Конструктивная — волны усиливают друг друга, увеличивая амплитуду.

Деструктивная — волны ослабляют друг друга, снижая амплитуду. *Пример:* Музыкальные инструменты, такие как гитары, часто используют интерференцию для создания сложных звуков.

Резонанс возникает, когда объект колеблется с частотой, совпадающей с естественной частотой другой системы. Это явление может значительно увеличивать амплитуду колебаний. *Пример:* Струнные музыкальные инструменты используют резонанс для усиления звука.

Гармоники — это дополнительные частоты, которые появляются при звучании основной частоты. Они добавляют богатство и глубину

звуку. *Пример:* Удар по струне гитары создает основной тон и его гармоники, формируя уникальный тембр инструмента.

Звуковые волны играют важную роль в нашей жизни, от создания музыки до медицинских приложений и технологий. Понимание их природы и характеристик позволяет нам использовать их более эффективно и безопасно.

2.3 Математические модели основных типов звуковых волн (сигналов)

В данной части рассматриваются основные типы звуковых волн и их применение. Основное различие между типами волн заключается в их форме и спектральных характеристиках, определяющих уникальные свойства каждой волны, такие как яркость и насыщенность звука.

Разработанный алгоритм, будет позволять выбирать тип звуковой волны для генерации звука. Это позволит изменять звучание в зависимости от выбранной формы волны, создавая разнообразные звуковые эффекты и тембры. Доступные для выбора волны включают синусоидальную, треугольную, квадратную и пилообразную, они и будут рассмотрены далее.

2.3.1 Синусоидальная волна

Синусоида (или синусоидальная волна) — это математическая кривая, описывающая гладкие периодические колебания. Она выражается как функция синуса и имеет форму, повторяющуюся через равные интервалы времени. Синусоида является фундаментальной формой

периодических колебаний и широко используется в различных областях науки и техники.

Синусоида используется для моделирования многих природных и искусственных процессов благодаря своим уникальным свойствам:

- ***Периодичность и гладкость*** (синусоидальные волны являются непрерывными и дифференцируемыми, что делает их идеальными для описания плавных и повторяющихся процессов).
- *Гармоничность.* Синусоиды служат основой для описания гармонических колебаний, которые являются наиболее простыми и фундаментальными типами колебаний.
- *Аналитическая простота.* Синусоидальные функции обладают простыми математическими свойствами, что упрощает их анализ и применение в различных задачах.

Синусоидальные волны часто встречаются в физике, в частности при описании колебаний струн музыкальных инструментов. Когда струна натягивается и отпускается, она начинает колебаться, создавая звуковые волны. Эти колебания можно аппроксимировать синусоидальной функцией, что позволяет точно моделировать и анализировать их поведение.

Математическое выражение синусоидальной волны задано формулой:

$$y(t) = A\,sin\,(2\pi f + \varphi), \qquad \textbf{(7)}$$

где:

($y(t)$) — значение синусоиды в момент времени (t),

(A) — амплитуда, максимальное отклонение от среднего значения,

(f) — частота, количество колебаний в единицу времени,

(φ) — начальная фаза, сдвиг волны относительно начала координат.

Синусоиды (рис. 2.2) играют ключевую роль в математическом описании и анализе периодических процессов. Их простота и универсальность делают их незаменимыми в различных научных и инженерных приложениях, от анализа звуковых волн до моделирования электромагнитных колебаний.

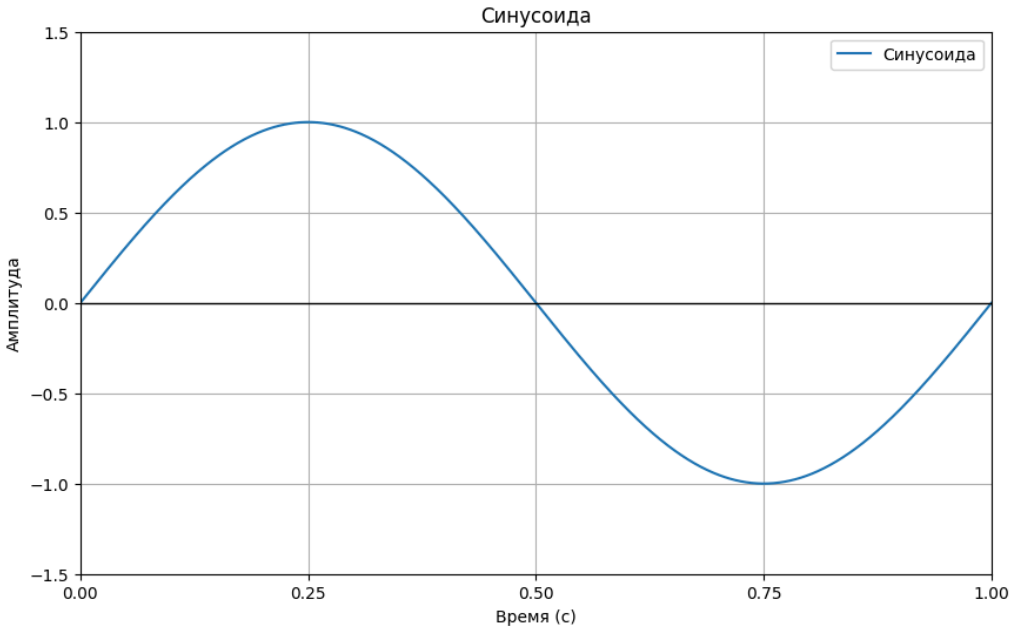


Рис. 2.2 Графическое представление синусоидальной волны

2.3.2 Пилообразная волна

Пилообразная волна (или пилообразный **сигнал**) — это математическая кривая, характеризующаяся линейным, возрастающим (или убывающим) поведением в течение одного периода, за которым следует резкий возврат к начальной точке. Эта форма сигнала напоминает зубцы пилы, откуда и происходит её название. Пилообразные волны широко используются в различных областях техники и науки, особенно в звуковой и визуальной обработке.

Пилообразная волна находит применение в моделировании многих процессов благодаря своим уникальным свойствам:

- *Резкие переходы.* Пилообразная волна имеет линейное увеличение и резкий спад, что делает её идеальной для представления сигналов с моментальными изменениями.
- *Аналитическая простота.* Несмотря на свою неидеальную форму, пилообразные волны могут быть легко описаны и сгенерированы математически.
- *Широкий спектр применений.* Из-за своей простой и предсказуемой структуры, пилообразные волны используются в генерации сигналов, синтезе звука и других приложениях, где важны четкие переходы и периодичность.

Пилообразные волны часто встречаются в музыке и звуковом синтезе. Они являются основой для многих электронных музыкальных инструментов, таких как синтезаторы, благодаря своему богатому спектру гармоник, который придает звуку яркий и насыщенный характер. Пилообразная волна содержит все гармонические составляющие, что делает её звук сложным и интересным для слуха.

Математическое выражение пилообразной волны (рис. 2.3) задано следующей формулой

$$y(t) = A\,(1 - 2\left|\frac{t}{T} - round\,(\frac{t}{T})\right|), \qquad \textbf{(8)}$$

где:

$y(t)$ — значение пилообразной волны в момент времени (t),

A — амплитуда, максимальное отклонение от среднего значения,

t — частота, количество повторений волны в единицу времени,

T — номер гармоники в ряду Фурье.

Пилообразные волны, благодаря своей форме и спектральным характеристикам, имеют множество применений в цифровой обработке сигналов, генерации синтетических звуков и моделировании периодических процессов. Их способность обеспечивать чёткие и резкие изменения делает их незаменимыми в тех областях, где требуются сложные и насыщенные сигналы.

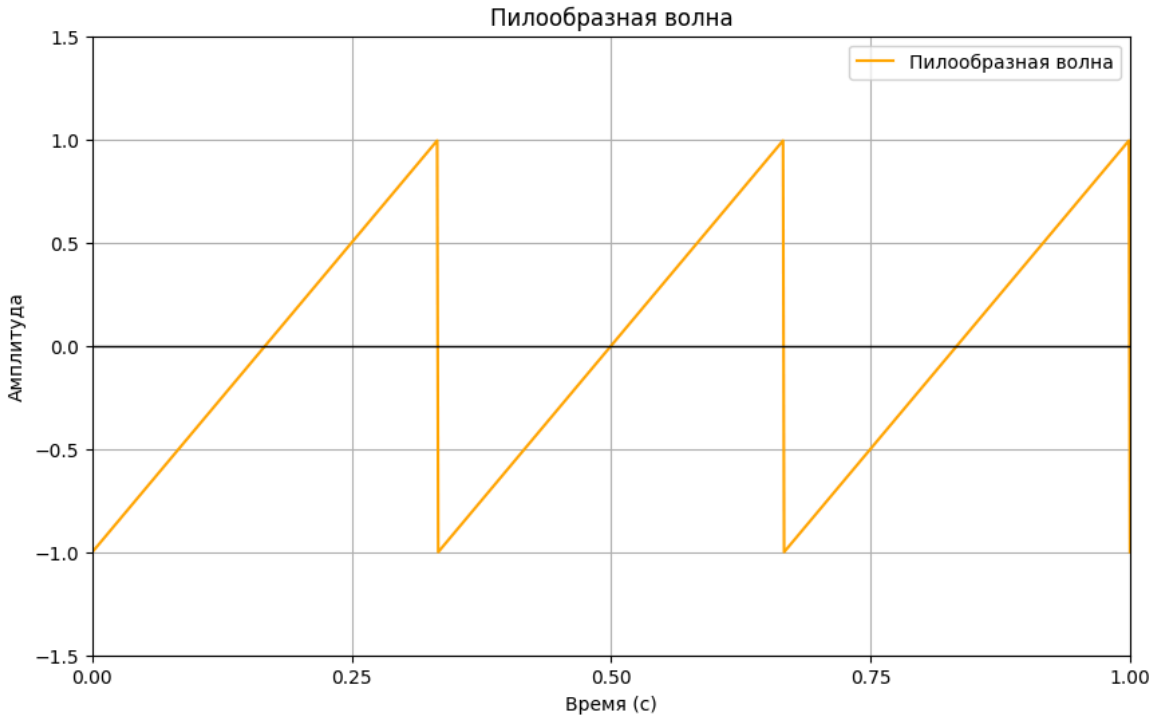


Рис. 2.3 Графическое представление пилообразной волны

Далее рассмотрим квадратный тип музыкальных сигналов.

2.3.3 Квадратная волна

Квадратная волна (или квадратный **сигнал**) представляет собой математическую кривую, характеризующуюся резкими переходами между двумя фиксированными значениями в течение периода. Эта форма сигнала напоминает последовательность прямоугольных импульсов, отсюда и происходит её название. Квадратные волны находят широкое применение в

различных областях, включая цифровую обработку сигналов, телекоммуникации и управление.

Квадратные волны используются для моделирования и анализа различных процессов благодаря следующим свойствам:

- ***Резкие переходы.*** Квадратная волна характеризуется мгновенными переходами между двумя уровнями, что делает её идеальной для представления сигналов с двоичной природой и для управления сигналами.
- ***Простота генерации.*** Квадратные волны могут быть сгенерированы простыми логическими схемами, что делает их популярным выбором для цифровых приложений.
- ***Широкий диапазон применений.*** Благодаря своей простой структуре и чётким переходам, квадратные волны используются в цифровой электронике, включая генерацию тактовых сигналов, модуляцию и демодуляцию, а также в системах управления и синхронизации.

Квадратные волны широко используются в цифровых системах, таких как цифровая обработка сигналов, компьютерные сети и телекоммуникации. Они представляют собой базовый элемент в цифровой логике, используемый для передачи информации в виде последовательности битов.

Математическое выражение квадратной волны может быть представлено следующим образом:

$$y(t) = \left\{ \frac{A, \text{ если } 0 \le t < \frac{T}{2}}{-A, \text{ если } \frac{T}{2} \le t < T} \right\}, \quad \textbf{(9)}$$

где:

$(y(t))$ — значение квадратной волны в момент времени (t),

(A) — амплитуда, константное значение во время активного уровня,

(T) — период квадратной волны.

В телекоммуникационных системах квадратные волны (рис. 2.4) используются для генерации и демодуляции цифровых сигналов. Они обеспечивают быстрые и чёткие переходы между уровнями сигнала, что позволяет эффективно передавать и интерпретировать информацию.

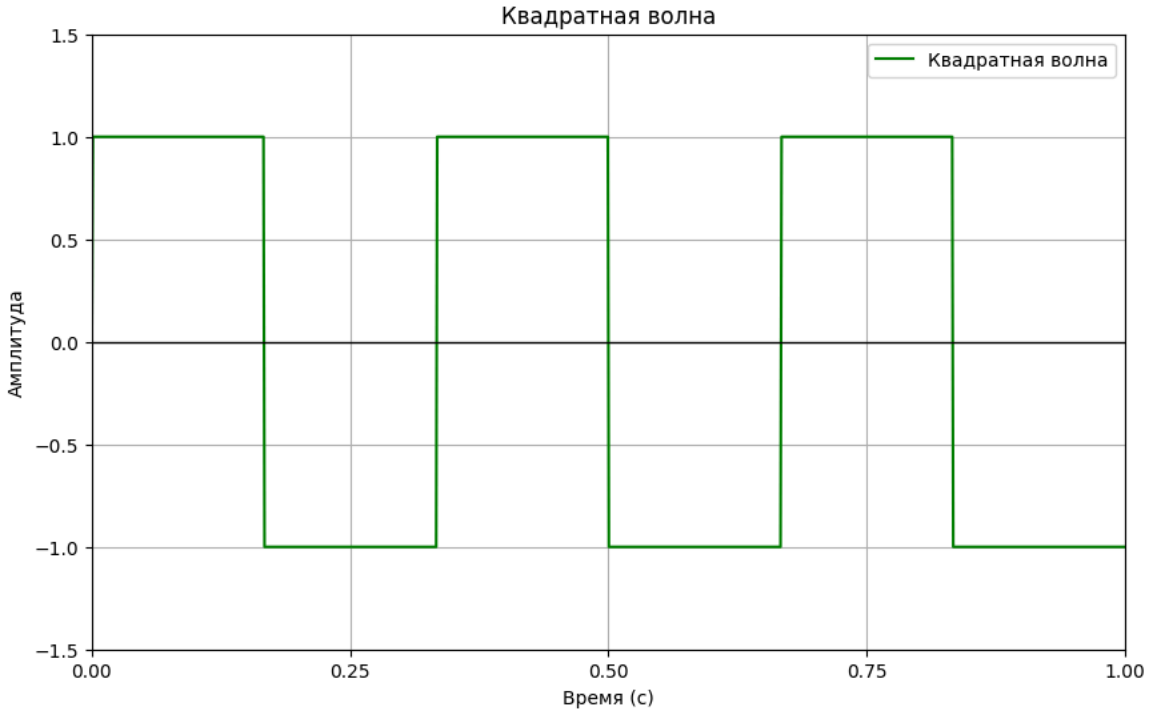


Рис. 2.4 Графическое представление квадратной волны

Квадратные волны представляют собой важный элемент в цифровой электронике и телекоммуникациях. Их простота и эффективность делают их незаменимыми для различных цифровых приложений, где важна точность и скорость обработки сигналов.

2.3.4 Треугольная волна

Треугольная волна (или треугольный **сигнал**) представляет собой форму сигнала, характеризующуюся линейным возрастанием (или

убыванием) амплитуды в течение одного периода, за которым следует резкое изменение направления движения. Этот сигнал получил своё название благодаря своему сходству с формой треугольника. Треугольные волны находят применение в различных областях техники и науки, особенно в обработке сигналов и электронике.

Треугольные волны используются для моделирования различных процессов благодаря их уникальным характеристикам:

- ***Плавные переходы.*** Треугольная волна характеризуется линейным изменением амплитуды, что делает её удобной для представления сигналов с плавными переходами.
- ***Простота в обработке.*** Математическое описание треугольной волны просто и понятно, что облегчает её анализ и использование в различных приложениях.
- ***Широкий спектр применений.*** Благодаря своей предсказуемой структуре, треугольные волны используются в генерации сигналов, тестировании оборудования и других областях, где важны плавные изменения сигнала.

Треугольные волны часто используются в звуковом синтезе для создания различных звуковых эффектов. Они могут послужить основой для синтезаторов и других музыкальных инструментов, добавляя звукам особый характер и теплоту. Треугольная волна обладает богатым спектром гармоник, что делает её звук насыщенным и интересным для слушателя.

Математически треугольная волна может быть описана следующим образом:

$$y(t) = A\,(1 - 2\left|\frac{t}{T} - round\,(\frac{t}{T})\right|), \qquad \textbf{(10)}$$

где:

($y(t)$) — значение треугольной волны в момент времени (t),

(A) — амплитуда, максимальное отклонение от среднего значения,

(T) — период волны.

В графике и видео треугольные волны могут использоваться для создания различных эффектов. Например, они могут служить для генерации анимаций с плавными переходами или для синхронизации временных сигналов в видеопотоке.

Треугольные волны (рис. 2.5) представляют собой важный инструмент в обработке сигналов и звуковой инженерии, обеспечивая плавные и естественные звуковые эффекты.

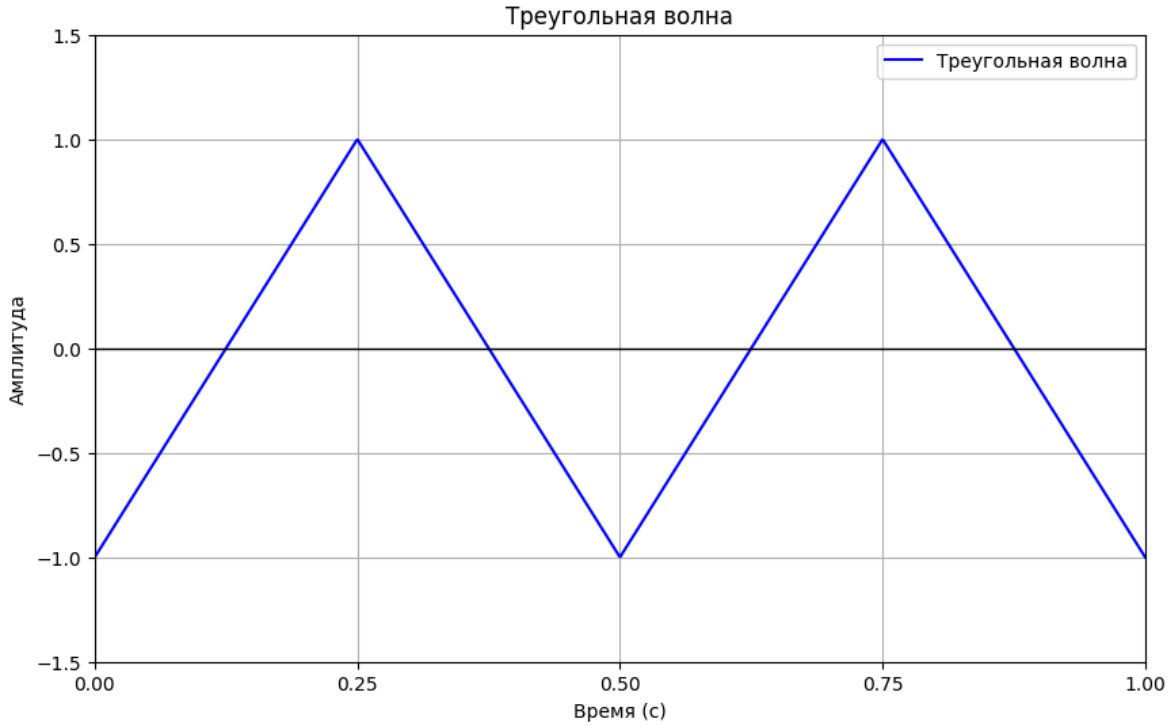


Рис. 2.5 Графическое представление треугольной волны

Сравнивая графики и звучание всех четырех типов волн - синусоидальной, пилообразной, квадратной и треугольной, мы можем сделать следующие выводы:

1. **Графическое представление.** Синусоидальная волна имеет плавные, гладкие кривые синусоиды, без резких переходов или углов. Пилообразная волна характеризуется линейным, возрастающим или убывающим поведением в течение одного

периода, с резкими переходами в начале и конце периода. Квадратная волна имеет резкие переходы между двумя постоянными значениями, образуя квадратные импульсы. Треугольная волна образует плавные переходы между двумя крайними значениями через линейное изменение, напоминающее форму треугольника.

2. **Звучание.** Синусоидальная волна имеет чистый, мягкий звук, часто ассоциируемый с природными звуками или музыкальными инструментами. Пилообразная волна создает звук с насыщенными высокими частотами из-за резких переходов, что делает его ярким и насыщенным. Квадратная волна звучит более резко и насыщенно из-за резких переходов между уровнями, что делает её заметной и энергичной. Треугольная волна имеет богатый гармонический спектр, создавая звук с насыщенностью, сходной с квадратной волной, но с более гладкими переходами.
3. **Применение.** Синусоидальная волна широко используется в аудиосигналах, синусоидальных генераторах, световой технике и многих других областях. Пилообразная волна применяется в синтезе звука, генерации сигналов и в тех областях, где важны резкие изменения. Квадратная волна используется в электронике, цифровой обработке сигналов и в приложениях, где требуются четкие переходы. Треугольная волна находит применение в звуковой обработке, аудиосинтезе и других областях, где нужны плавные переходы с высокой насыщенностью звука.

Итак, каждый тип волны имеет свои уникальные характеристики, которые делают их подходящими для различных приложений в аудио, электронике, обработке сигналов и других областях.

2.4 Выводы второй главы

Глава 2 представляет аналитический обзор математических методов, используемых для генерации музыкальных композиций, а также основные понятия и области применения звуковых волн.

В разделе 2.1 рассматриваются различные методы генерации музыкальных композиций, такие как генетические алгоритмы, нейронные сети, скрытые марковские модели и другие. Каждый из этих методов предоставляет уникальные возможности для создания и эволюции музыкальных фрагментов. Однако, подходы не всегда учитывают классические правила теории музыки, что подталкивает к необходимости создания собственного алгоритма, основанного на формализации основных правил музыкальной гармонии.

В разделе 2.2 обсуждаются основные понятия звуковых волн и их важность в различных областях, начиная от музыкальной индустрии и заканчивая медицинскими приложениями. Здесь представлен обзор различных форм звуковых волн, их характеристик и областей применения.

В разделе 2.3 представлены математические модели основных типов звуковых волн, таких как синусоидальная, пилообразная, квадратная и треугольная. Каждый тип волны описан с математической точки зрения, предоставляя понимание их структуры и особенностей. Также обсуждается их использование в различных приложениях и областях.

Обобщая, глава 2 позволяет получить обзор современных методов генерации музыки и понимание основных понятий и характеристик звуковых волн. Это предоставляет фундаментальные знания для дальнейшего изучения и развития в области компьютерной музыки и искусственного интеллекта.

ГЛАВА 3. РАЗРАБОТКА АЛГОРИТМА ДЛЯ АВТОМАТИЧЕСКОЙ ГЕНЕРАЦИИ МЕЛОДИЙ

3.1 Описание алгоритма

Цель данной программы заключается в разработке эффективного алгоритма для автоматической генерации музыкальных мелодий на основе заданных пользователем параметров, включая тон, мажор/минор, длительность мелодии, тип волны и темп.

Программа будет начинать работу с ввода пользовательских данных, включая выбор тона, указание мажора или минора, определение длительности мелодии, выбор типа звуковой волны и указание темпа. Затем, используя известную частоту (например, 440 Гц для ноты ля), программа будет преобразовывать выбранный пользователем тон в соответствующую частоту, используя формулу для вычисления частоты ноты.

Следующим этапом будет формирование цепочки аккордов на основе выбранного тона. Цепочка аккордов будет представлять собой последовательность ступеней тональности, где каждая ступень будет соответствовать звуку в аккорде. Важным шагом в этом процессе будет проверка и при необходимости транспонирование аккордов для обеспечения их соответствия одной октаве.

После формирования цепочки аккордов программа будет создавать звуковую волну аккомпанемента, где каждый аккорд будет представлен звуковой волной, а длительность аккорда будет определяться исходя из выбранного темпа.

Следующим шагом будет генерация мелодии. Для этого программа будет использовать массив частот аккордов и формировать массив ступеней для каждого аккорда. Затем, с помощью функции случайной

генерации ритмического рисунка, программа будет определять длительность каждой ноты в мелодии. Каждой ноте будет случайным образом сопоставляться частота одной из ступеней выбранного аккорда.

Наконец, аккорды и мелодия будут объединяться в один звуковой файл формата .wav, который будет сохраняться для последующего воспроизведения.

Общий алгоритм основного программного решения ВКР приведен на рис. 3.1.

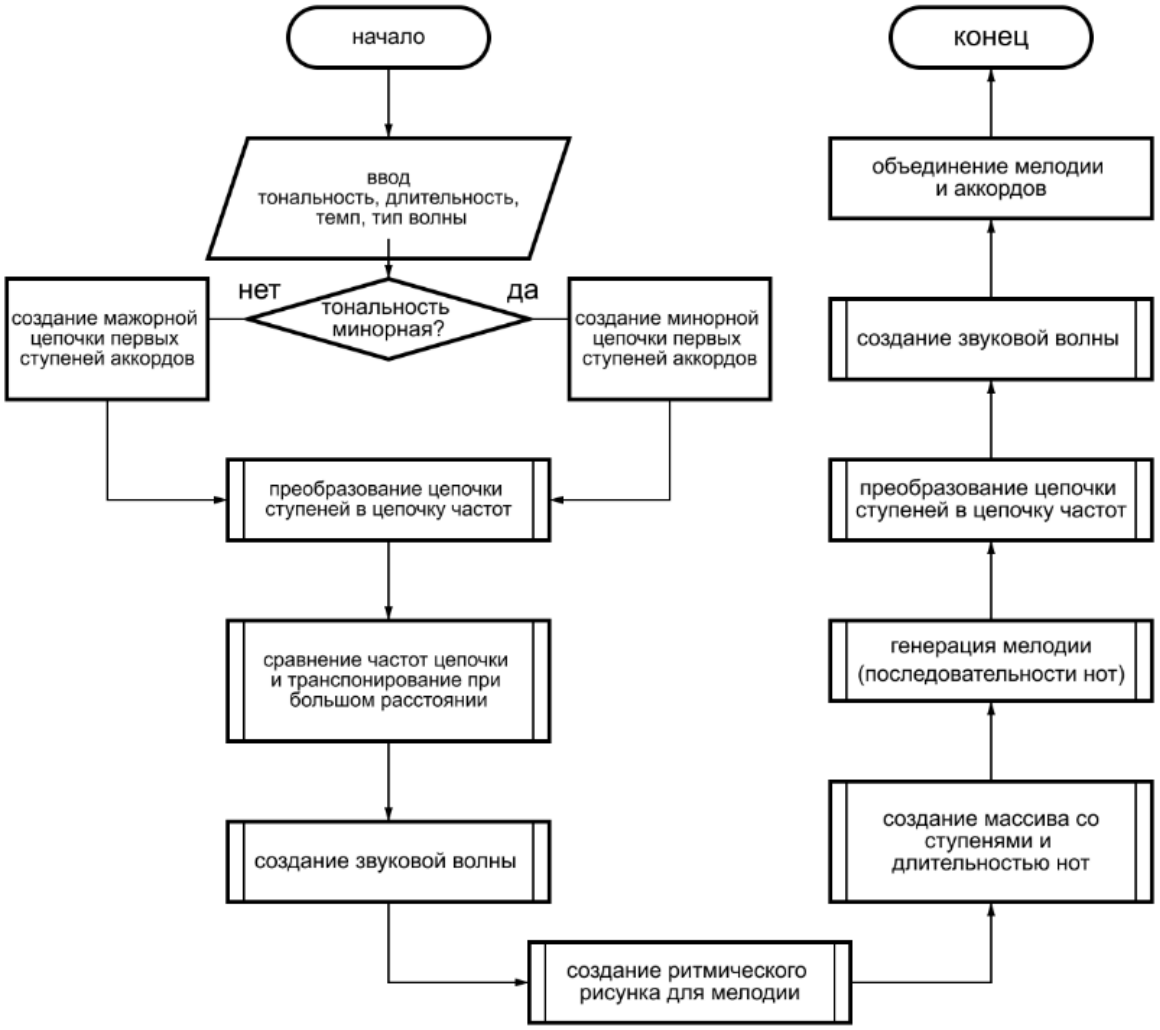


Рис. 3.1 Алгоритм генерации мелодий на основании музыкальных гармоник

Описанный алгоритм учитывает различные пользовательские параметры, такие как тон, мажор/минор, длительность мелодии, тип звуковой волны и темп, а также включает преобразование частот, формирование цепочки аккордов, создание звуковых волн аккомпанемента и генерацию мелодии.

Теперь рассмотрим алгоритм написания песни с использованием музыкальных понятий и подходов. В следующем разделе будет представлен более детализированный подход к выбору аккордов и построению мелодии на основе общепринятых принципов гармонии.

3.2 Описание алгоритма музыкальным языком

В данной работе представлен алгоритм написания песни, основанный на базовых методах выбора аккордов и построения мелодии. Этот алгоритм использует общепринятые принципы гармонии, позволяя создавать музыкальные произведения с внутренней логикой и выразительностью.

Для разработки алгоритма используются базовые теории музыкальной гармонии и тональности. Эти теории описывают, как аккорды взаимодействуют между собой и как создаются гармонические прогрессии, которые определяют структуру и движение музыкальной композиции. В основе алгоритма лежат следующие принципы:

- **Тональная центрированность** — использование аккордов, связанных с одной тональностью, чтобы создать устойчивость и логичность в композиции.
- **Функциональная гармония** (рис. 3.2) — аккорды распределяются по их функциям в тональной системе (**тоника, доминанта, субдоминанта** и т.д.), что помогает создавать напряжение и разрешение в музыке.

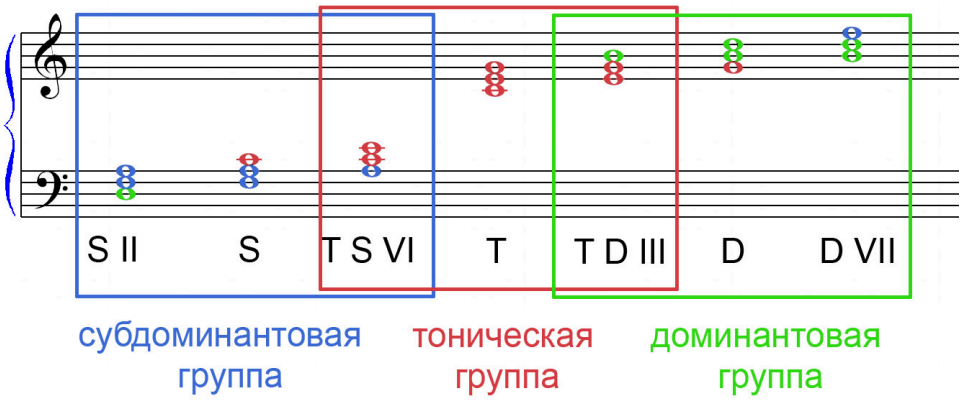


Рис. 3.2 Функциональная гармония

При анализе гармонического движения используется ряд методов и подходов, которые будут упомянуты в дальнейшем (рис. 3.3):

- **Аккордовые последовательности** — обозначение аккордов с указанием их функции и положения в тональной системе (например, T, S, D).
- **Нумерация аккордов** — обозначение аккордов/ступеней римскими цифрами в зависимости от их положения в тональности, например, I, IV, V.

Рис. 3.3 Обозначение аккордов

Алгоритм написания песни

Шаг 1. Выбор последовательности аккордов

Начальный аккорд (1 аккорд) — тоника (I ступень). Тоника задает основную тональность композиции и создает ощущение устойчивости. Например, в тональности C мажор это будет аккорд C мажор.

Последний аккорд (4 аккорд)— доминанта (V ступень). Доминанта создает напряжение и требует разрешения, что делает её идеальным аккордом для завершения фразы или произведения. Например, в тональности C мажор это будет аккорд G мажор.

Средние аккорды (2-3 аккорды)— II, III, IV, VI, или VII ступени тональности. Эти аккорды выбираются по правилам гармонической функции и добавляют разнообразие в прогрессию. Например:

- II ступень (D минор в тональности C): добавляет движение.
- IV ступень (F мажор в тональности C): подготавливает к доминанте.
- VI ступень (A минор в тональности C): добавляет меланхоличный оттенок.

Пример последовательности аккордов:
I (C) - VI (Am) - IV (F) - V (G)

Шаг 2. Создание мелодии

Выбор нот для мелодии. Под каждый аккорд подбираются ноты, входящие в его состав. Например, для аккорда C мажор это ноты C, E, G.

Вариативность мелодии. Мелодия может варьироваться за счет выбора разных нот из аккорда. Под каждый аккорд можно использовать от

одной до нескольких нот (в зависимости от длительности), что позволяет создавать разнообразные мелодические линии.

Пример:

Под аккорд I (C мажор: C, E, G) можно использовать ноты C, E или G.

Под аккорд VI (A минор: A, C, E) можно использовать ноты A, C или E.

Под аккорд IV (F мажор: F, A, C) можно использовать ноты F, A или C.

Под аккорд V (G мажор: G, B, D) можно использовать ноты G, B или D.

Описание и пример алгоритма показаны на рис. 3.4 и рис. 3.5.

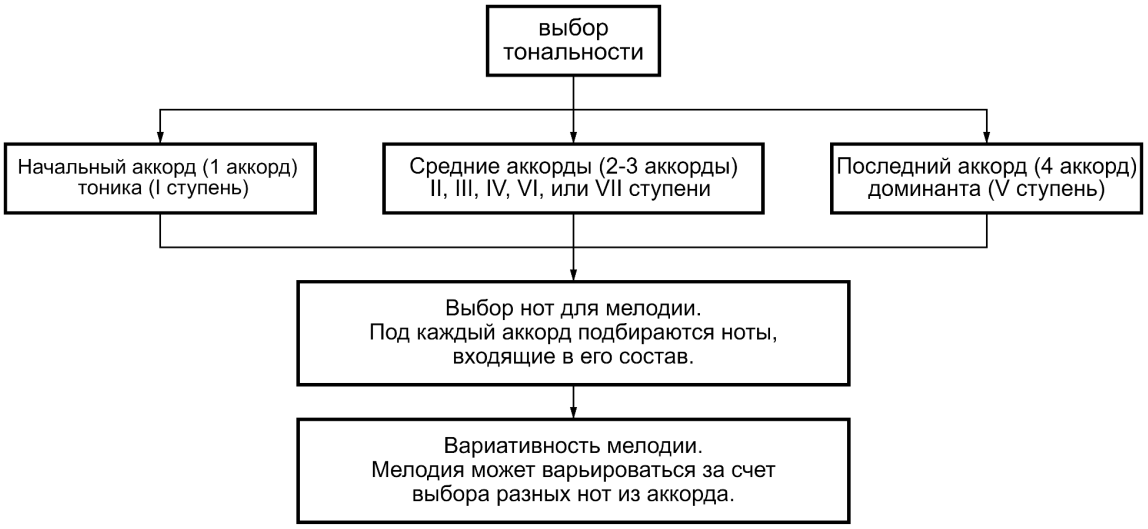


Рис. 3.4 Выбор тональности

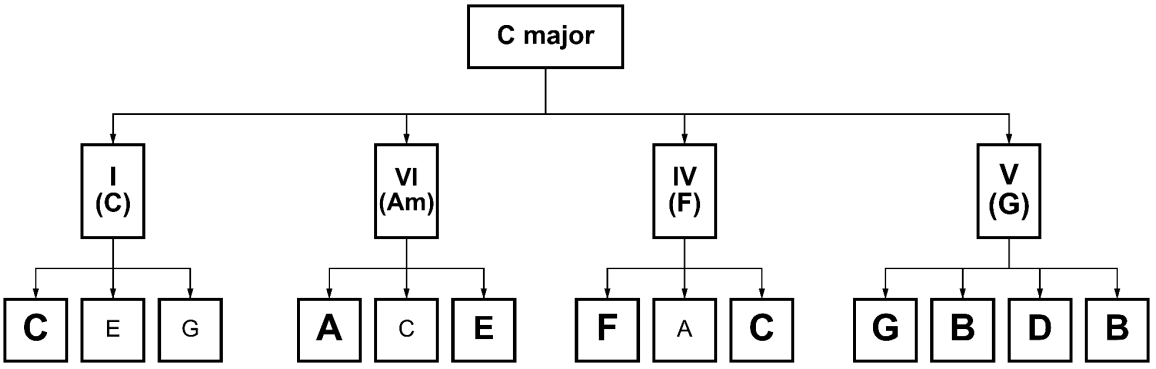


Рис. 3.5 Пример работы алгоритма (аккорды + мелодия)

Последовательность **C-A-E-F-C-B-D-B** — в данном примере является итоговой мелодией. Добавление нескольких нот под один аккорд происходит за счет сокращения их длительностей. Пример длительностей в секундах можно увидеть на рис. 3.6. Последовательность длительностей называется ритмическим рисунком.

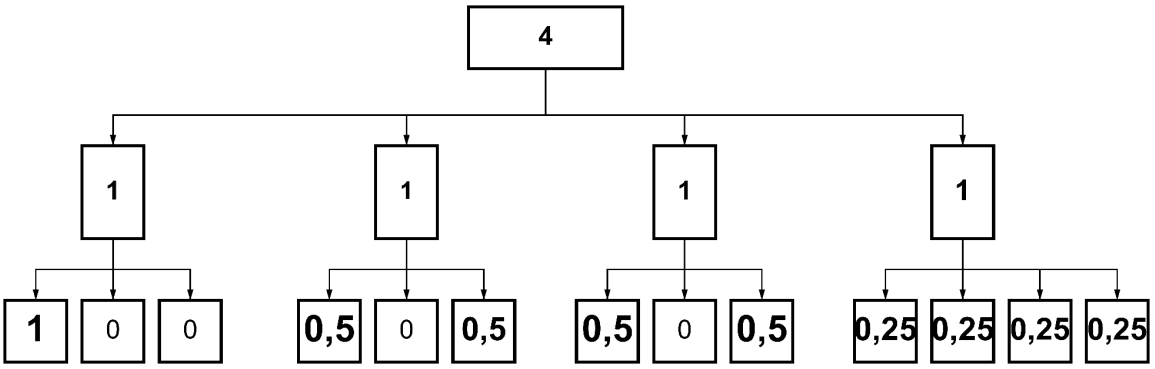


Рис. 3.6 Ритмический рисунок

Используя описанный алгоритм, можно создать аккомпанемент и мелодию для песни. Сочетание выбранных аккордов и соответствующих нот обеспечивает логичность и гармоничность композиции, что является общепринятым подходом в музыкальной практике. Базовые теории гармонии и тональности, используемые в алгоритме, помогают структурировать музыкальное произведение, создавая устойчивую и выразительную музыкальную форму.

3.3 Используемые библиотеки, оборудование и инструменты разработки

Для разработки использовался язык программирования “***Python***”. Для реализации программного решения использовались следующие библиотеки:

NumPy. Используется для работы с массивами и матрицами, предоставляет высокоуровневые математические функции.

Matplotlib. Используется для визуализации данных, в том числе для построения графиков, диаграмм и изображений.

SciPy. Используется для чтения и записи аудиофайлов, так как включает модули для работы с сигналами и обработки звука.

IPython.display. Используется для отображения аудиофайлов и других интерактивных элементов в блокнотах Jupyter. Может использоваться для прослушивания аудиофайлов, отображения графиков и других мультимедийных данных прямо в блокноте.

Random. Библиотека для работы с случайными числами. В данном контексте может быть использована для генерации случайных данных или для случайного выбора элементов из массивов.

Math. Предоставляет математические функции и константы. Может быть использована для выполнения различных математических операций, если они не предоставляются в NumPy или других библиотеках.

Для написания и тестирования решения использовались персональные компьютеры со следующими характеристиками:

– операционная система: MS Windows 10, MS Windows 8.1;

– 64-разрядная версия операционной системы (ОС);

– многоядерный процессор (4 ядра и больше) с тактовой частотой 3 ГГц и выше;

– 16 ГБ оперативной памяти и более;

– видеокарта с поддержкой OpenGL 4.5, с 2 ГБ видеопамяти и более, пропускная способность видеопамяти — 80 ГБ/с и более;;

– монитор с разрешением 1920x1080 пикселей или более.

Используемый web-браузер - “Google Chrome”.

3.4 Разработка и тестирование алгоритма

Программа начинает работу с ввода пользовательских данных, включающих в себя:

- выбор тона,
- указание мажора или минора,
- определение длительности мелодии,
- выбор типа звуковой волны и указание темпа.

Затем, используя известную частоту (например, 440 Гц для ноты ля), программа преобразует выбранный пользователем тон в соответствующую частоту, используя формулу для вычисления частоты ноты (2) в частном виде (11):

$$f_n = f_0 \times 2^{1/12}, \quad \textbf{(11)}$$

где f_n - частота n-й ноты, f_0 - частота опорной ноты, n - количество полутонов между опорной нотой и искомой нотой.

Функция, вычисляющая частоту ноты:

```
def note_frequency(base_freq, step):
    a = 2 ** (1 / 12)
    return base_freq * (a ** step)

note_frequency(440, 2)
493.8833012561241
```

Следующим этапом является формирование цепочки аккордов на основе выбранного тона.

```
# Создание цепочки аккордов для минорной тональности (по ступеням):
if key == 'minor':
    second, third = random.sample([3, 4, 6, 7], 2)
    chords_chain = [1, second, third, 5]
                                    ## Пример: chords_chain = [1, 3, 7, 2]
```

Цепочка аккордов представляет собой последовательность ступеней тональности, где каждая ступень соответствует звуку в аккорде. Для воспроизведения мелодии ступени нужно перевести в частоты (рис 3.7).

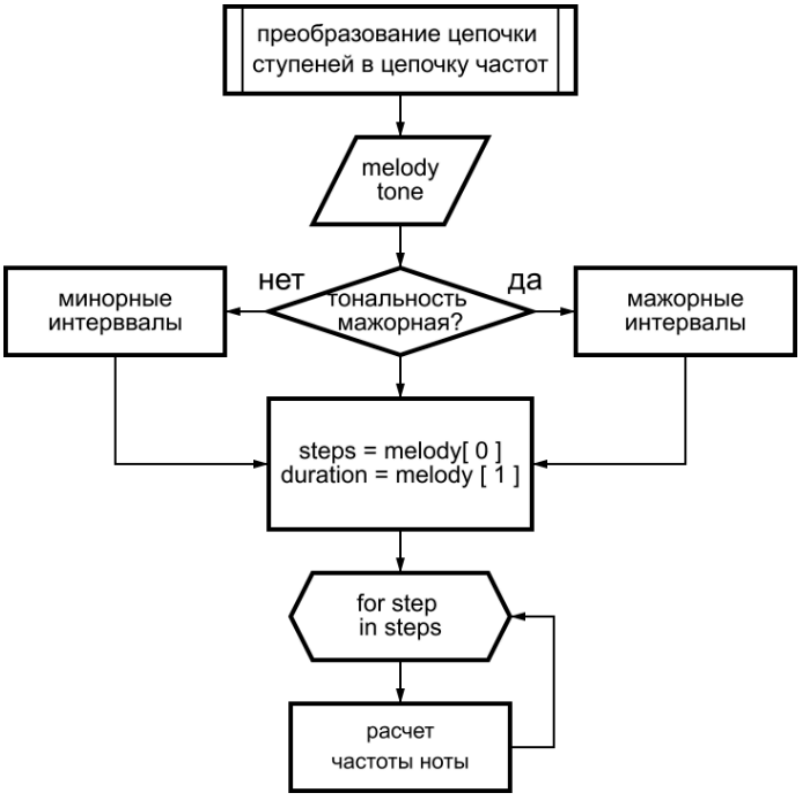


Рис. 3.7 Модуль (подпрограмма) преобразования ступеней в частоты

Важным шагом в этом процессе является проверка и при необходимости транспонирование аккордов для обеспечения их соответствия одной октаве (рис. 3.8).

Пример до транспонирования:

frequencies = [164.81, 293.6580354734188, 219.99495636576341, 246.93598934004595]

после:

frequencies = [164.81, 146.8290177367094, 109.99747818288171, 123.46799467002297]

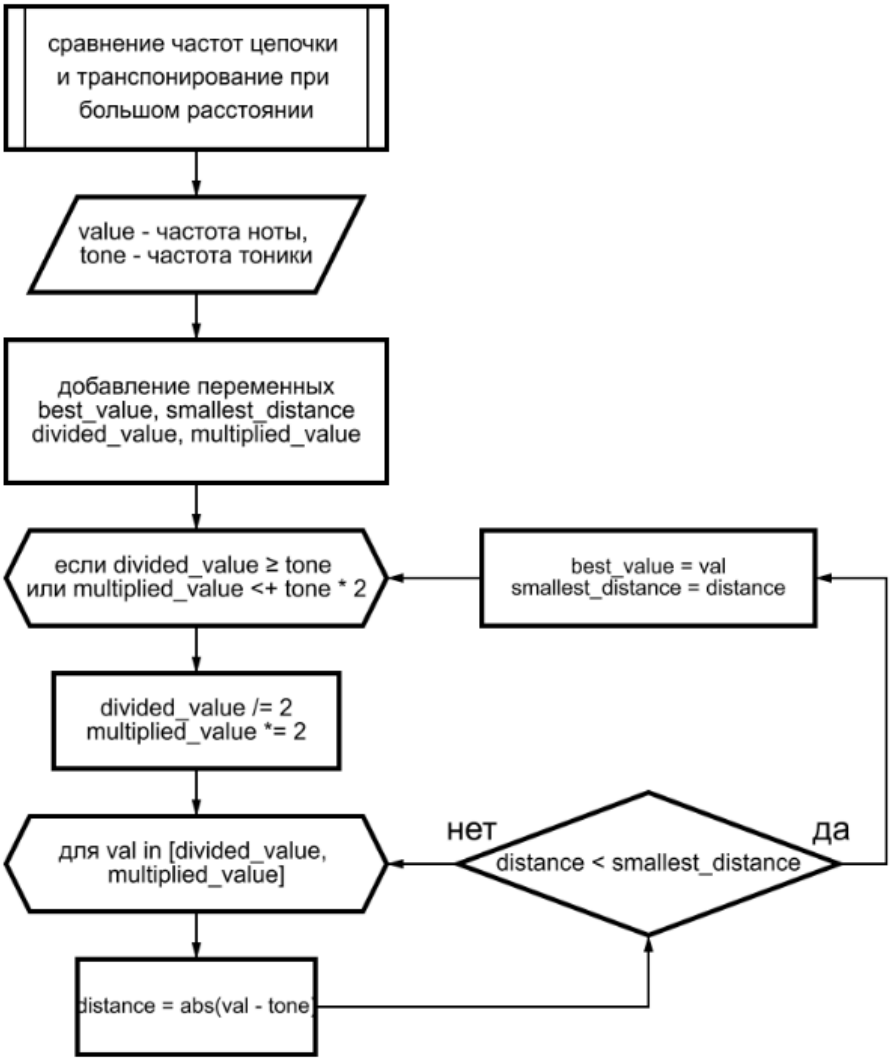


Рис. 3.8 Модуль (подпрограмма) сравнения частот цепочки и транспонирование при большом расстоянии

После формирования цепочки аккордов программа создает звуковую волну аккомпанемента, где каждый аккорд представлен звуковой волной, а длительность аккорда определяется исходя из выбранного темпа (рис. 3.9).

```
# Пример создания волны для минорной тональности:
    chord_1 = minor_triad(tone, duration/i)
    if chain[1] == 4:
        chord_2 = minor_triad(chords[1], chain_duration/i)
    else:
        chord_2 = major_triad(chords[1], chain_duration/i)
```

```
    if chain[2] == 4:
        chord_3 = minor_triad(chords[2], chain_duration/i)
    else:
        chord_3 = major_triad(chords[2], chain_duration/i)
    chord_4 = major_triad(chords[3], chain_duration/i)

    chain_wave = np.concatenate((chord_1, chord_2, chord_3, chord_4))
    accompaniment = np.tile(chain_wave, duration // chain_duration)

    if duration % chain_duration > 0:
        accompaniment = np.concatenate((accompaniment,
chain_wave[:remaining_samples]))

        write('accompaniment.wav', sample_rate, accompaniment)
```

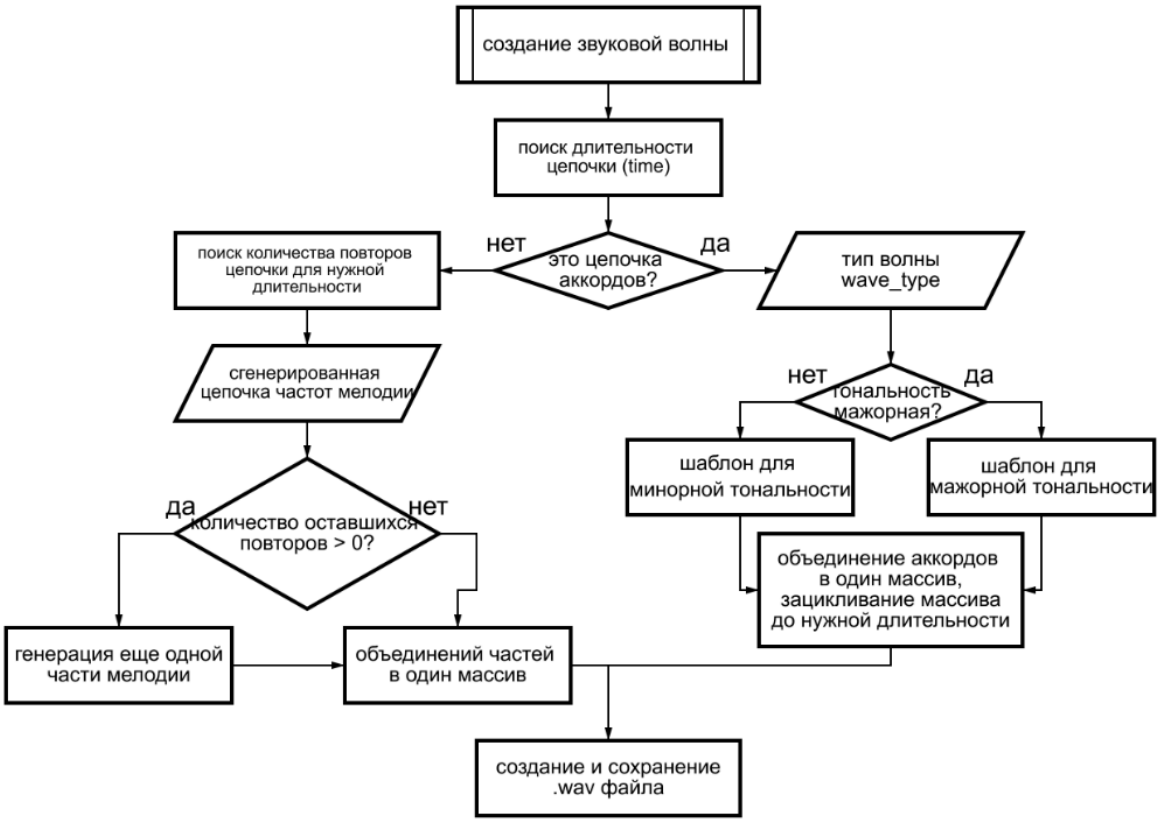


Рис. 3.9 Модуль (подпрограмма) для формирования звуковой волны из частот

Следующий шаг заключается в генерации мелодии, начиная с создания ритмического рисунка для мелодии (рис. 3.10).

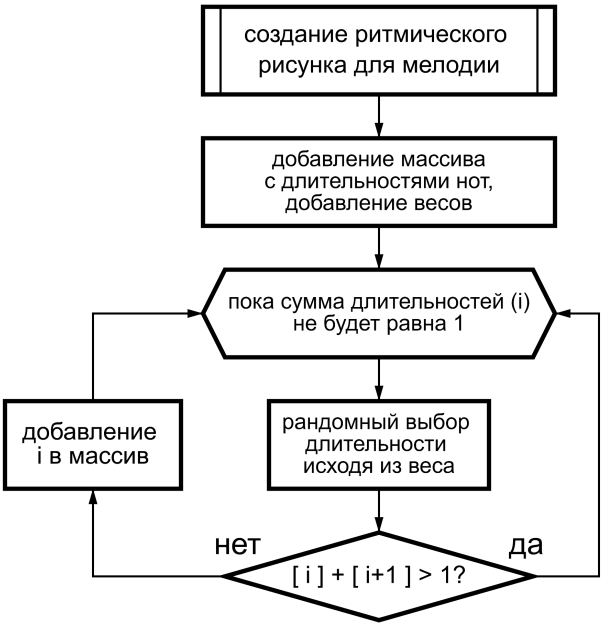


Рис. 3.10 Модуль (подпрограмма) для создания ритмического рисунка

```
  # Функция для создания ритмического рисунка для мелодии (0.125-1 это
длительность ноты):
    def rhythmic_pattern():
        values = [0.125, 0.25, 0.5, 0.75, 1]
        weights = [5, 4, 3, 2, 1]
        pattern = []
        current_sum = 0

        while current_sum < 1:
            value = random.choices(values, weights)[0]
            if current_sum + value <= 1:
                pattern.append(value)
                current_sum += value
        random.shuffle(pattern)
        return pattern
  # Пример: pattern = [0.125, 0.375, 0.375, 0.125]
```

Затем создается функция (рис. 3.11), позволяющая генерировать последовательность нот из выбранного аккорда в соответствии с ритмическим рисунком.

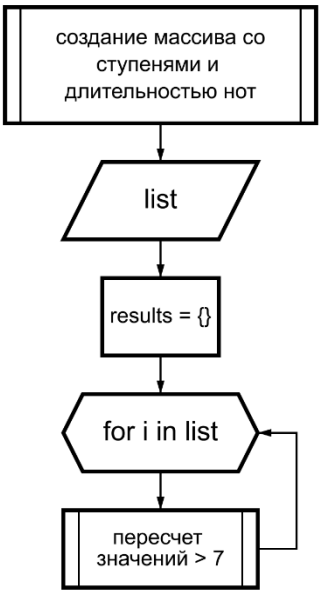


Рис. 3.11 Модуль (подпрограмма) создания массива со ступенями и длительностью нот

После генерации отрывков мелодии, необходимо объединить их в целостную мелодию. Следующая функция позволяет генерировать целостную мелодию (рис. 3.12).

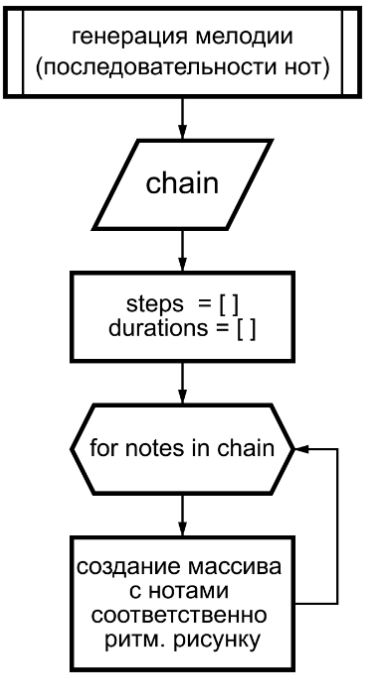


Рис. 3.12 Модуль (подпрограмма) для генерации мелодий

После генерации цепочки со ступенями, их необходимо перевести в цепочку частот для последующего создания звуковой волны (рис. 3.13).

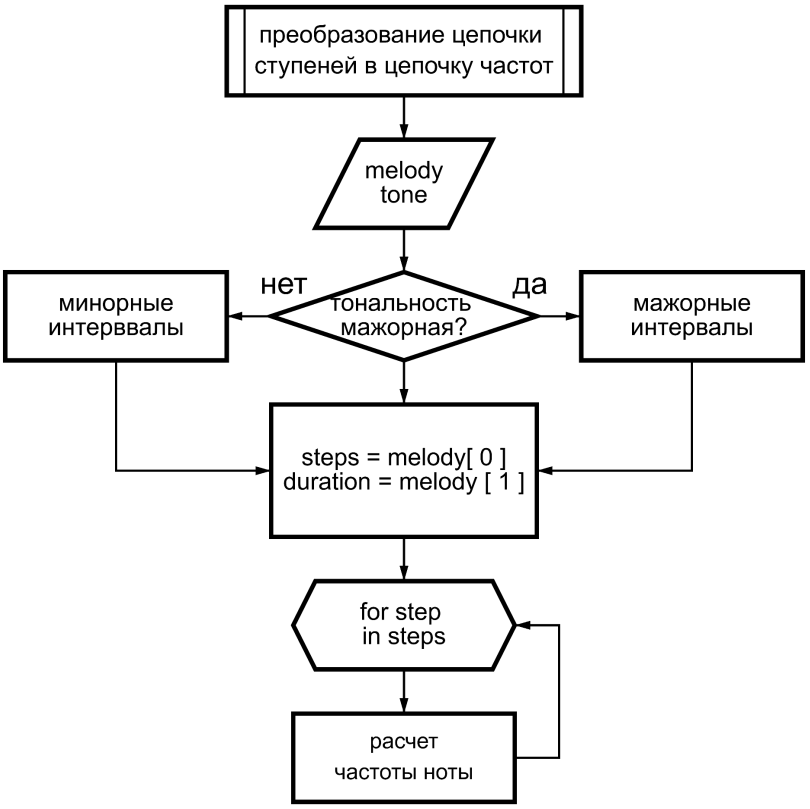


Рис. 3.13 Модуль (подпрограмма) преобразования цепочки ступеней в цепочку частот

Затем созданный массив частот преобразуется в звуковую волну, подстраиваясь под длительность аккомпанемента (рис. 3.9).

Наконец, аккорды и мелодия объединяются в один звуковой файл формата .wav, который сохраняется для последующего воспроизведения, код:

```
## Объединение мелодии и аккомпанемента:
final_wave = melody + accompaniment
final_wave = np.int16(final_wave / np.max(np.abs(final_wave)) * 32767)
write('music.wav', sample_rate, final_wave)
```

Полный текст кода реализованного в данной ВКР алгоритма по генерации мелодий на основании гармоник опубликован по ссылке [12] и приведен в приложении 1.

3.5 Выводы третьей главы

Глава 3 посвящена разработке алгоритма для автоматической генерации музыкальных мелодий. Алгоритм успешно генерирует логичные мелодии, используя принципы музыкальной гармонии и тональности.

Программа учитывает различные параметры, такие как тон, мажор/минор, длительность мелодии, тип звуковой волны и темп, создавая разнообразные композиции.

Алгоритм формирует аккорды и их последовательности, основываясь на функциональной гармонии, обеспечивая логичное движение и напряжение в мелодиях.

Создание мелодии происходит на основе ритмического рисунка и частот аккордов, что обеспечивает разнообразие и оригинальность.

В результате работы были успешно сгенерированы музыкальные мелодии, которые можно прослушать в репозитории по адресу [14].

Эти мелодии демонстрируют эффективность и разнообразие разработанного алгоритма, подтверждая его способность создавать логичные и гармоничные музыкальные произведения.

ЗАКЛЮЧЕНИЕ

Главы 1, 2 и 3 данной выпускной квалификационной работы представляют собой последовательные этапы исследования, направленного на создание алгоритма для автоматической генерации музыкальных мелодий. В первой главе представлен обзор существующих методов и подходов к созданию музыки с использованием компьютерных технологий. Это включает основные принципы и подходы к созданию музыкальных произведений при помощи алгоритмов и программного обеспечения, а также обзор существующих исследований и проектов в этой области.

Во второй главе подробно рассмотрен процесс анализа и выбора методов для реализации алгоритма. Описаны используемые библиотеки и инструменты разработки, обосновывается их выбор и применимость к поставленным задачам. Также представлены основные теоретические концепции и принципы, которые лежат в основе разрабатываемого алгоритма.

Третья глава описывает сам алгоритм генерации мелодий на основе музыкальных гармоник. Здесь представлен алгоритм как с использованием традиционного программирования, так и с использованием музыкального языка. Подробно описываются этапы алгоритма, начиная с выбора параметров пользователем и заканчивая объединением аккомпанемента и мелодии в единый звуковой файл.

В заключении можно отметить, что разработанный алгоритм представляет собой рабочий инструмент для автоматической генерации музыкальных мелодий. Он учитывает различные пользовательские параметры и применяет основные принципы музыкальной гармонии для создания логичных и выразительных композиций. Результаты

исследования показывают, что алгоритм способен генерировать разнообразные мелодии, демонстрируя его потенциал в области компьютерного создания музыки.

СПИСОК ЛИТЕРАТУРЫ И ИНТЕРНЕТ-ИСТОЧНИКОВ

1. Андрейкина М.В. Музыка и математическая логика: от учения Пифагора к теории А. Ф. Лосева // Вестник МГУКИ. 2019. №6 (92). URL: https://clck.ru/3Ae3Bk (дата обращения: 13.05.2024).

2. Никитин, Н.А. Обзор математических методов для генерации музыкальных композиций / Н. А. Никитин, В. Л. Розалиев, Ю. А. Орлова. — Текст : непосредственный // Молодой ученый. — 2020. — № 52 (342). — С. 36-39. — URL: https://moluch.ru/archive/342/77077/ (дата обращения: 11.05.2024).

3. Никитин Н.А., Розалиев В.Л., Орлова Ю.А., Заболеева-Зотова А.В. ПРИМЕНЕНИЕ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ЗАДАЧИ ГЕНЕРАЦИИ МУЗЫКАЛЬНЫХ КОМПОЗИЦИЙ // Прикаспийский журнал: управление и высокие технологии. 2018. №2 (42). URL: https://clck.ru/3Ae3DP (дата обращения: 11.05.2024).

4. **Никитин, Н.А.** Алгоритм генерации музыкальных композиций с использованием интуитивного и эмоционального подходов / Н. А. Никитин, Ю. А. Орлова, В. Л. Розалиев. — Текст : непосредственный // Молодой ученый. — 2021. — № 24 (366). — С. 35-39. — URL: https://moluch.ru/archive/366/82308/ (дата обращения: 11.05.2024).

5. Благовещенская Е.А.Алгебраические структуры в теории музыки // SAEC.2019.№2. URL: https://clck.ru/3Ae3GL (дата обращения: 13.05.2024).

6. Маццола, Гуэрино (2018), Музыкальная топология: геометрическая логика понятий, теории и исполнения URL: https://clck.ru/3Ae3Kq (дата обращения: 11.05.2024).

7. **Мартюгин С.А. , Поршнев С.В.** Быстрое дробное преобразование Фурье // International Journal of Open Information

Technologies. 2024. №1. URL: https://clck.ru/3Ae3Ps (дата обращения: 15.05.2024).

8. Бершадская Т. С. Гармония в звуковысотной системе музыки как материальная категория // Opera musicologica. 2015. №2 (24). URL: https://cyberleninka.ru/article/n/garmoniya-v-zvukovysotnoy-sisteme-muzyki-kak-materialnaya-kategoriya (дата обращения: 15.05.2024).

9. Холодков Д.В. АНАЛИЗ ОСОБЕННОСТЕЙ ПРИМЕНЕНИЯ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ // Вестник науки. 2024. №4 (73). URL: https://clck.ru/3Ae3RR (дата обращения: 15.05.2024).

10. Саруханян С.К., Масловская А.Г. ПРОСТЕЙШИЙ КЛЕТОЧНЫЙ АВТОМАТ ДЛЯ МОДЕЛИРОВАНИЯ ПОВЕРХНОСТНОГО РОСТА БАКТЕРИЙ // Вестник Амурского государственного университета. Серия: Естественные и экономические науки. 2022. №99. URL: https://clck.ru/3Ae3TW (дата обращения: 15.05.2024).

11. Анализ аудиоданных (дата обращения: 15.05.2024).

Часть 1 URL: https://habr.com/ru/articles/668518/

Часть 2/ URL: https://habr.com/ru/amp/publications/670676/

12. Programming a Guitar Tuner with Python. URL: https://www.chciken.com/digital/signal/processing/2020/05/13/guitar-tuner.html (дата обращения: 15.05.2024).

13. GraduateWork URL: https://github.com/cristinka-p/GraduateWork (дата обращения: 15.05.2024).

14. Ссылка на итоговый результат **[final music examples].** URL: https://github.com/cristinka-p/GraduateWork/tree/main/final%20music%20examples (дата обращения: 15.05.2024).

ПРИЛОЖЕНИЕ

FINAL CODES/Final_Melody_Generation.ipynb

```
def melody_generation(tone = 440, key = 'minor', duration = 5, bpm = 120,
wave_type = 'sine', sample_rate = 44100):

    # Функции:

    ## Функция для поиска частоты:
    def note_frequency(tone, step):
        a = 2 ** (1 / 12)
        return tone * (a ** step)

    ## Сравнение частот цепочки и транспонирование при большом расстоянии
    def adjust_to_tone(value, tone):
        best_value = value
        smallest_distance = abs(value - tone)
        divided_value = value
        multiplied_value = value

        while divided_value >= tone or multiplied_value <= tone * 2:
            divided_value /= 2
            multiplied_value *= 2

            for val in [divided_value, multiplied_value]:
                distance = abs(val - tone)
                if distance < smallest_distance:
                    best_value = val
                    smallest_distance = distance
        return best_value

    ## Создание мажорного и минорного трезвучий:
    def major_triad(tone, chord_duration):

        mediant = note_frequency(tone, 4)
        dominant = note_frequency(tone, 7)
        frequencies = [tone, mediant, dominant]

        waves = [wave_functions[wave_type](freq) for freq in frequencies]

        wave = np.sum(wave, axis=0)
        wave /= np.max(np.abs(wave))
        wave = np.int16(wave * 32767)
        return wave

    def minor_triad(tone, chord_duration):
```

```
        mediant = note_frequency(tone, 4)
        dominant = note_frequency(tone, 7)
        frequencies = [tone, mediant, dominant]

        wave = [wave_functions[wave_type](freq) for freq in frequencies]

        wave = np.sum(wave, axis=0)
        wave /= np.max(np.abs(wave))
        wave = np.int16(wave * 32767)
        return wave


    ## Функции, генерирующие разные типы волн:
    def sine(frequency, amplitude=0.5):
        wave = amplitude * np.sin(2 * np.pi * frequency * time)
        return wave
    def saw(frequency, amplitude=0.5):
        period = 1 / frequency
        phase = (time / period) % 1
        wave = (amplitude * 2 * phase) - amplitude
        return wave
    def square(frequency, amplitude=0.5):
        wave = amplitude * np.sign(np.sin(2 * np.pi * frequency * time))
        return wave
    def triangle(frequency, amplitude=0.5):
        wave = amplitude * (2 * np.abs(2 * (time * frequency - np.floor(time *
frequency + 0.5))) - 1)
        return wave


    ## Функция для создания звуковой волны аккомпанемента
    def create_chord_wave(tone, duration, key, chords_chain):
        i = len(chain)
        chords = chords_chain

        if key == 'major':
            chord_1 = mor_triad(tone, duration/i)
            chord_2 = mor_triad(chords[1], chain_duration/i)
            chord_3 = mor_triad(chords[2], chain_duration/i)
            chord_4 = mor_triad(chords[3], chain_duration/i)

        else:
            chord_1 = minor_triad(tone, duration/i)
            if chain[1] == 4:
                chord_2 = minor_triad(chords[1], chain_duration/i)
            else:
                chord_2 = major_triad(chords[1], chain_duration/i)
            if chain[2] == 4:
                chord_3 = minor_triad(chords[2], chain_duration/i)
            else:
                chord_3 = major_triad(chords[2], chain_duration/i)
```

```
            chord_4 = major_triad(chords[3], chain_duration/i)

        chain_wave = np.concatenate((chord_1, chord_2, chord_3, chord_4))

        accompaniment = np.tile(chain_wave, duration // chain_duration)

        if duration % chain_duration > 0:
            accompaniment = np.concatenate((accompaniment,
chain_wave[:remaining_samples]))

        write('accompaniment.wav', sample_rate, accompaniment)
        return (chain_wave, accompaniment)
                                                ## Результат в конце
create_chord_wave это сохранение файла с аккомпанементом,
                                                ## сохранение волны всего
аккомпанемента и волны цепочки аккордов

    ## Функция для создания ритмического рисунка для мелодии (0.125-1 это
длительность ноты):
    def rhythmic_pattern():
        values = [0.125, 0.25, 0.5, 0.75, 1]
                                                                    ##[0.125,
0.25, 0.375, 0.5, 0.625, 0.75, 0.875, 1]
        weights = [5, 4, 3, 2, 1]
        pattern = []
        current_sum = 0

        while current_sum < 1:
            value = random.choices(values, weights)[0]
            if current_sum + value <= 1:
                pattern.append(value)
                current_sum += value
        random.shuffle(pattern)
        return pattern
                                                ## Пример: pattern = [0.125,
0.375, 0.375, 0.125]


    ## Функция для создания массива с ритмическим рисунком и соответственными
ступенями нот

    def chord_step(start):
        values = []
        current_value = start
        for _ in range(3):
            values.append(current_value)
            current_value += 2
            if current_value > 7:
                current_value = current_value % 7
                if current_value == 0:
                    current_value = 7
```

```
        return values

    def chord_st_chain(input_list):
        results = {}
        for i, value in enumerate(input_list):
            results[f"ch_{i+1}"] = chord_step(value)
        return results

    ## Функция для генерации мелодии:
    def generate_melody(chord_st_chain):
        all_steps = []
        all_durations = []

        for notes in chord_st_chain.values():
            rythm = rhythmic_pattern()
            melody_notes = [random.choice(notes) for _ in rythm]
            all_steps.extend(melody_notes)
            all_durations.extend(rythm)

        return [all_steps, all_durations]

    ## Функция для преобразования ступеней в частоты
    def melody_freq(melody, tone):

        if key == 'major':
            intervals = [0, 2, 3, 5, 7, 8, 10]
        else:
            intervals = [0, 2, 3, 5, 7, 8, 10]

        steps = melody[0]
        durations = melody[1]

        frequencies = [note_frequency(tone, intervals[step - 1]) for step in
steps]


        return [frequencies, durations]

    ## Функция для создания звуковой волны мелодии
    def create_melody_wave():
        num_repeats = duration // chain_duration
        remaining_time = duration % base_duration
        remaining_samples = int(remaining_time * semple_rate)

        base_melody = generate_melody(result=None)
        melody = np.tile(base_melody, int(num_repeats))

        if remaining_samples > 0:
            melody = np.concatenate((melody, base_melody[:remaining_samples]))
        write('melody.wav', sample_rate, melody)
        return melody
```

```
                                                ## Результат в конце
create_melody_wave это сохранение файла с мелодией
                                                ## и сохранение волны мелодии

    ## Создание цепочки аккордов для минорной и мажорной тональностей (по
тонам):
    def maj_min (key):
        if key == 'minor':
            second, third = random.sample([3, 4, 6, 7], 2)
            chords_chain = [1, second, third, 5]
                                                ## Пример: chords_chain = [1,
3, 7, 2]

        if key == 'major':
            second, third = random.sample([3, 4, 6, 7], 2)
            chords_chain = [1, second, third, 5]
                                                ## Пример: chords_chain = [1,
3, 7, 2]

        return chords_chain


    # Генерация музыки:

    chords_chain = maj_min(key)                 ## Определение тональности

        ## Преобразование цепочки ступеней в частоты
    major_intervals = [0, 2, 4, 5, 7, 9, 11]
    minor_intervals = [0, 2, 3, 5, 7, 8, 10]

    second = chords_chain[1] - 1
    third = chords_chain[2] - 1
    dominant = note_frequency(tone, 7)

    if key == 'major':
        second = major_intervals[second]
        third =  major_intervals[third]

    elif key == 'minor':
        second = minor_intervals[second]
        third =  minor_intervals[third]

    second = note_frequency(tone, second)
    third = note_frequency(tone, second)
    frequencies = [tone, second, third, dominant]    ## Пример: frequencies =
[164.81, 293.6580354734188, 219.99495636576341, 246.93598934004595]

    ## Сравнение частот и транспонирование при большом расстоянии
    for i in range(1, len(frequencies)):
        frequencies[i] = adjust_to_tone(frequencies[i], tone)
                                                ## Пример до: frequencies =
```

```
[164.81, 293.6580354734188, 219.99495636576341, 246.93598934004595]
                                                ##     после: frequencies =
[164.81, 146.8290177367094, 109.99747818288171, 123.46799467002297]

    ## Создание звуковой волны для цепочки аккордов:
    chain_duration = 4 / (bpm / 60)
    time = np.linspace(0, chain_duration, int(44100 * chain_duration),
endpoint=False)

    ## Генерация мажорного и минорного трезвучий:
    chord_duration = chain_duration/4
    wave_functions = {
        'sine': sine,
        'saw': saw,
        'square': square,
        'triangle': triangle
    }

    result = chord_st_chain(chords_chain)

    melody = generate_melody(result)

    melody_freq = melody_freq(melody, tone)

    ## Объединение мелодии и аккомпанемента:
    final_wave = melody + accompaniment
    final_wave = np.int16(final_wave / np.max(np.abs(final_wave)) * 32767)
    write('music.wav', sample_rate, final_wave)
    return (ipd.Audio('music.wav'))
```

waveforms/waveforms.ipynb

Звуковые волны

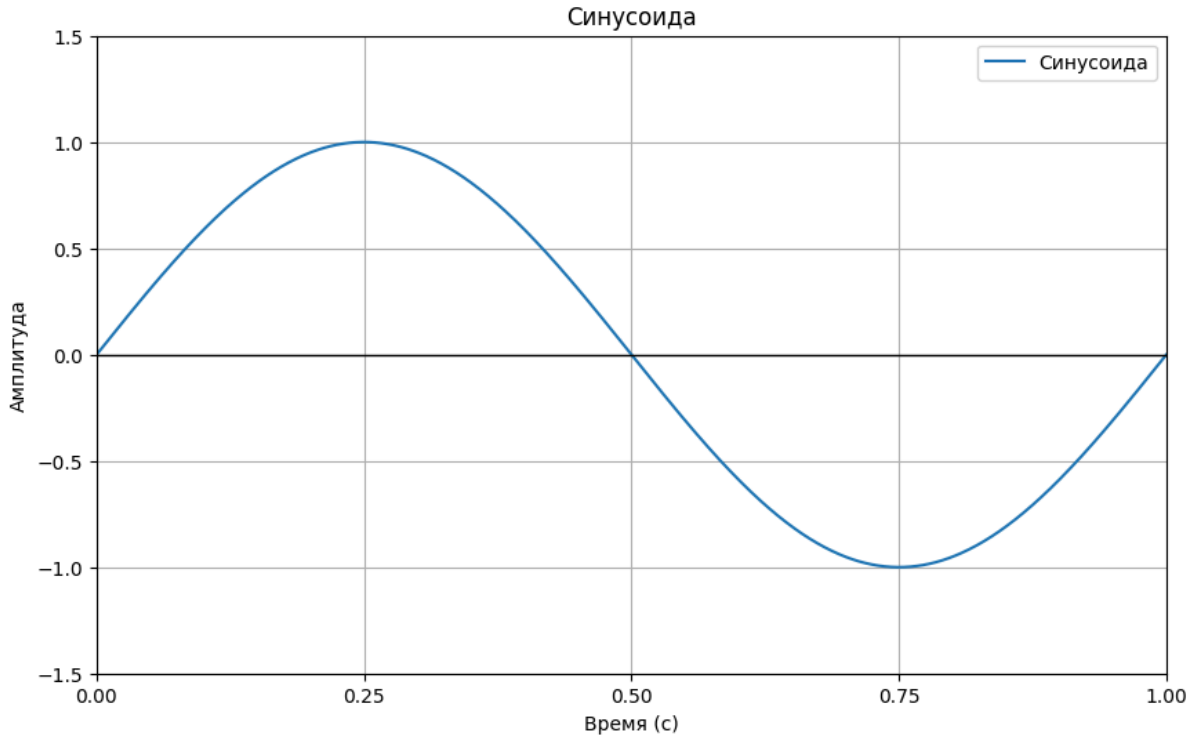
```
# Импорт библиотек
import numpy as np
import matplotlib.pyplot as plt
from scipy.io.wavfile import write
import IPython.display as ipd
```

Синусоидальная волна

Математическое выражение синусоидальной волны задано формулой:

$$y(t)=A\sin(2\pi ft+\varphi)$$

```
# Создание графика
frequency = 1
duration = 1
amplitude = 1

time = np.linspace(0, duration, 1000)

signal = amplitude * np.sin(2 * np.pi * frequency * time)

plt.figure(figsize=(10, 6))
plt.plot(time, signal, label='Синусоида')

plt.xlim(0, duration)
plt.ylim(-1.5, 1.5)

plt.xticks(np.arange(0, duration + 0.25, 0.25))

plt.grid()
plt.legend()
plt.xlabel('Время (с)')
plt.ylabel('Амплитуда')
plt.title('Синусоида')
plt.axhline(0, color='black', linewidth=1)

plt.show()
```




```
# Создание аудиофайла
frequency = 440
duration = 3
amplitude = 1

time = np.linspace(0, duration, int(duration * 44100))
sin_wave = amplitude * np.sin(2 * np.pi * frequency * time)

write("sinusoidal_wave.wav", 44100, np.int16(sin_wave * 32767))

ipd.Audio("sinusoidal_wave.wav")
```

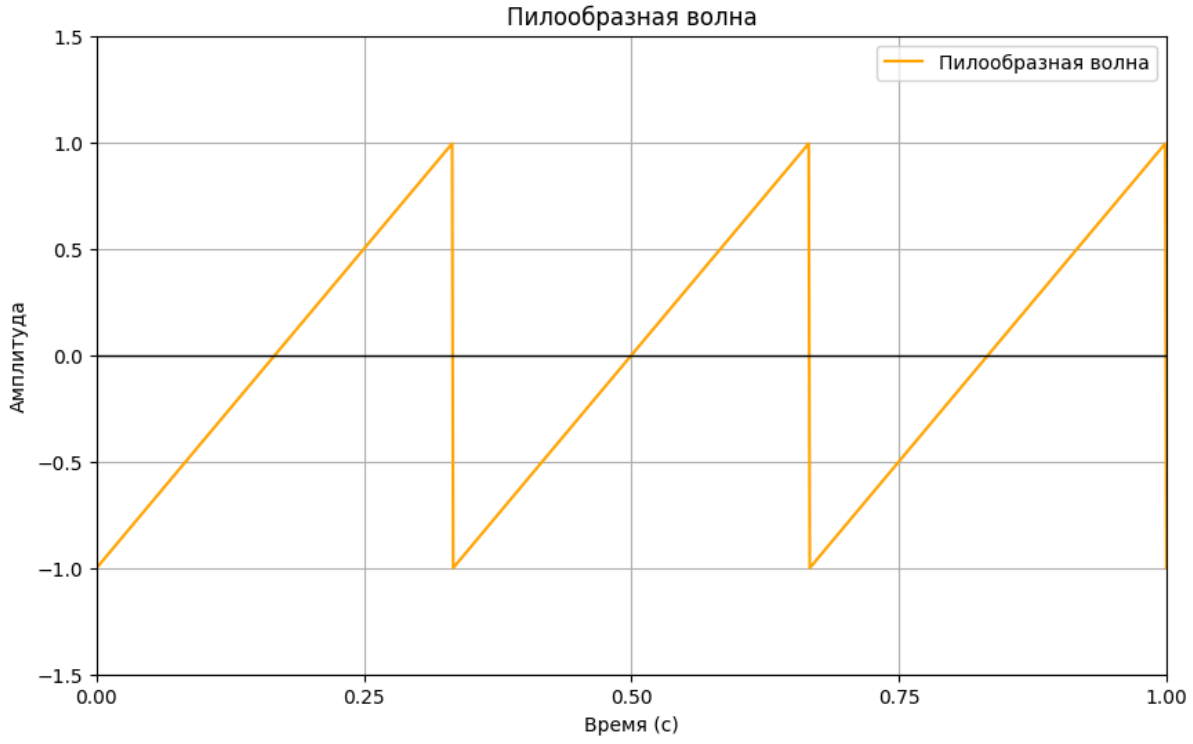
Пилообразная волна

Математическое выражение пилообразной волны задано следующей формулой:

$$y(t)=2A\pi\sum n=1\infty(-1)n+1n\sin(2\pi nft)$$

```
#Создание графика
frequency = 3
duration = 1
amplitude = 1

time = np.linspace(0, duration, 1000)

def sawtooth_wave(time, frequency, amplitude):
    period = 1 / frequency
```

```
    phase = (time / period) % 1
    return (amplitude * 2 * phase) - amplitude

signal = sawtooth_wave(time, frequency, amplitude)

plt.figure(figsize=(10, 6))
plt.plot(time, signal, label='Пилообразная волна', color='orange')  # Изменение цвета для отличия от синусоиды

plt.xlim(0, duration)
plt.ylim(-1.5, 1.5)

plt.xticks(np.arange(0, duration + 0.25, 0.25))

plt.grid()
plt.legend()
plt.xlabel('Время (с)')
plt.ylabel('Амплитуда')
plt.title('Пилообразная волна')
plt.axhline(0, color='black', linewidth=1)

plt.show()
```




```
# Создание аудиофайла
frequency = 440
duration = 3
amplitude = 1
```

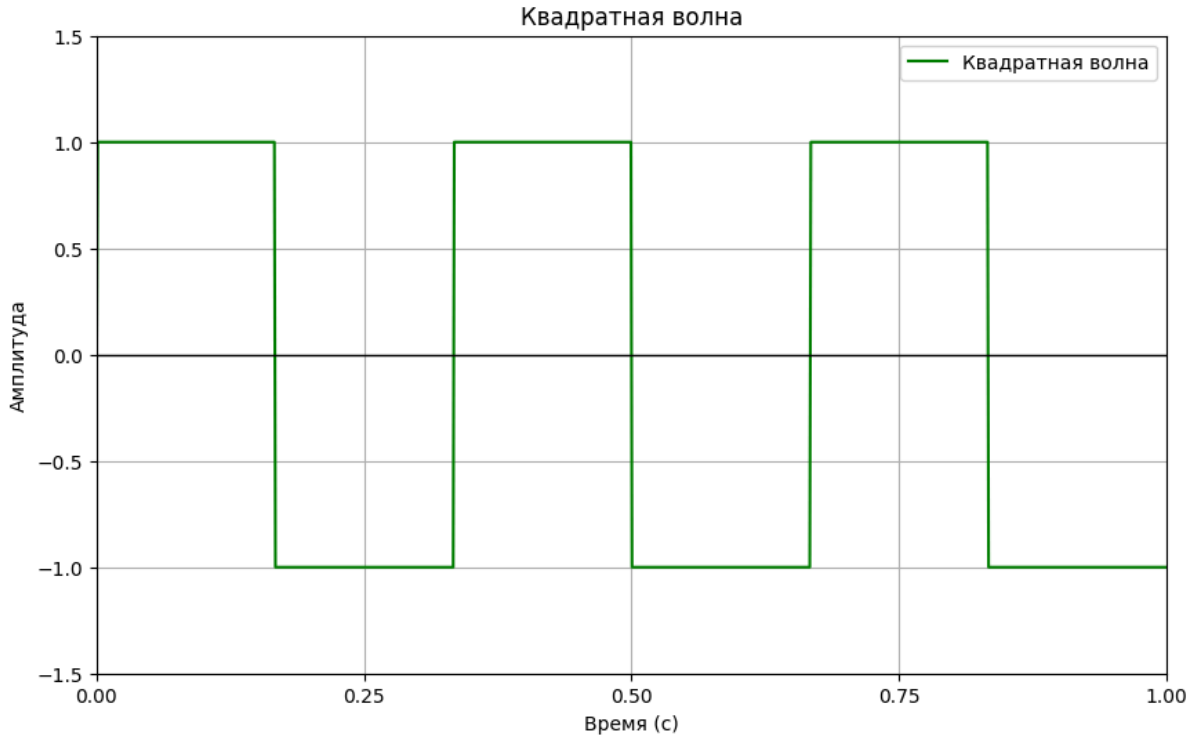
```
time = np.linspace(0, duration, int(duration * 44100))

signal = sawtooth_wave(time, frequency, amplitude)

write("sawtooth_wave.wav", 44100, np.int16(signal * 32767))

ipd.Audio("sawtooth_wave.wav")
```

Квадратная волна

Математическое выражение квадратной волны может быть представлено следующим образом:

$$y(t) = \{A, \text{если } 0 \leq t < T2 - A, \text{если } T2 \leq t < T$$

```
# Создание графика
frequency = 3
duration = 1
amplitude = 1

time = np.linspace(0, duration, 1000)

square_signal = amplitude * np.sign(np.sin(2 * np.pi * frequency * time))

plt.figure(figsize=(10, 6))
plt.plot(time, square_signal, label='Квадратная волна', color='green')

plt.xlim(0, duration)
plt.ylim(-1.5, 1.5)

plt.xticks(np.arange(0, duration + 0.25, 0.25))

plt.grid()
plt.legend()
plt.xlabel('Время (с)')
plt.ylabel('Амплитуда')
plt.title('Квадратная волна')
plt.axhline(0, color='black', linewidth=1)

plt.show()
```




```
# Создание аудиофайла
frequency = 440
duration = 3
amplitude = 1

time = np.linspace(0, duration, int(duration * 44100))

square_wave = amplitude * np.sign(np.sin(2 * np.pi * frequency * time))

write("square_wave.wav", 44100, np.int16(square_wave * 32767))

ipd.Audio("square_wave.wav")
```

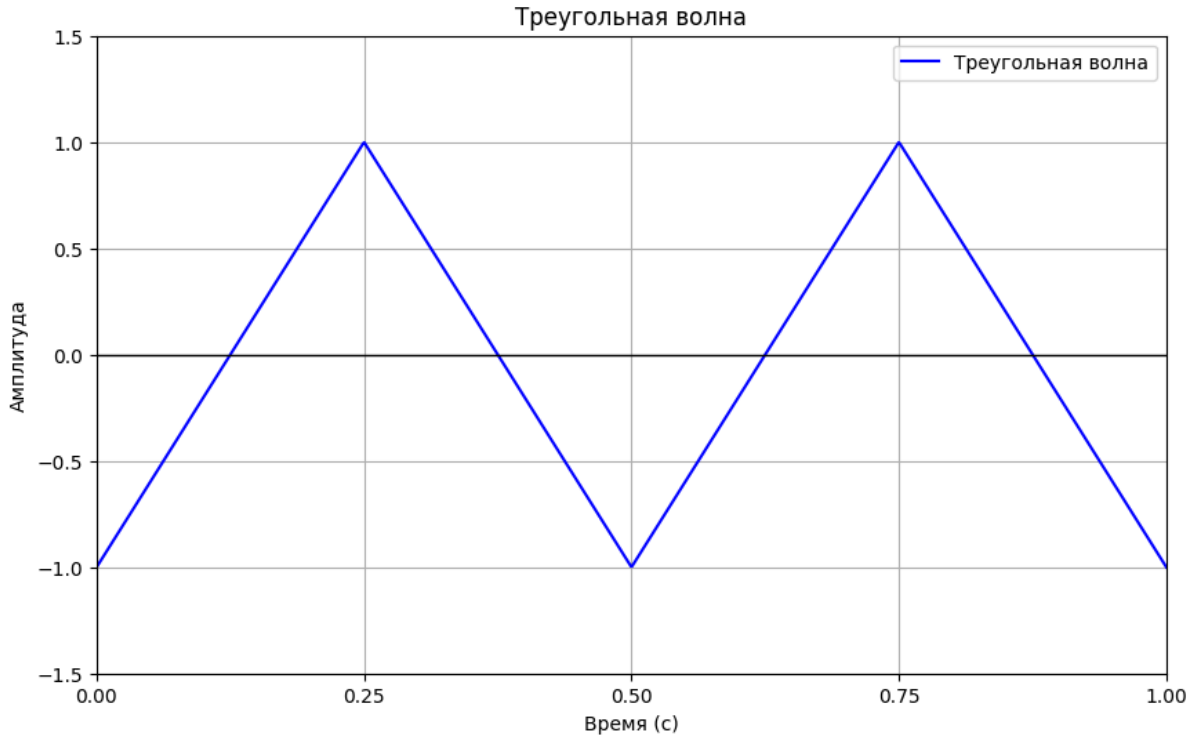
Треугольная волна

Математически треугольная волна может быть описана следующим образом:

$$y(t)=A(1-2|tT-\text{round}(tT)|)$$

```
# Создание графика
frequency = 2
duration = 1
amplitude = 1

time = np.linspace(0, duration, 1000)

triangle_signal = amplitude * (2 * np.abs(2 * (time * frequency - np.floor(time
* frequency + 0.5))) - 1)
```

```
plt.figure(figsize=(10, 6))
plt.plot(time, triangle_signal, label='Треугольная волна', color='blue')

plt.xlim(0, duration)
plt.ylim(-1.5, 1.5)

plt.xticks(np.arange(0, duration + 0.25, 0.25))

plt.grid()
plt.legend()
plt.xlabel('Время (с)')
plt.ylabel('Амплитуда')
plt.title('Треугольная волна')
plt.axhline(0, color='black', linewidth=1)

plt.show()
```




```
# Создание аудиофайла
frequency = 440
duration = 3
amplitude = 1

time = np.linspace(0, duration, int(duration * 44100))

triangle_wave = amplitude * (2 * np.abs(2 * (time * frequency - np.floor(time *
frequency + 0.5))) - 1)

write("triangle_wave.wav", 44100, np.int16(triangle_wave * 32767))
```

```
ipd.Audio("triangle_wave.wav")


# Создание функции для добавления гармоник
import numpy as np
from scipy.io.wavfile import write
import IPython.display as ipd

def generate_wave_with_harmonics(frequency, num_harmonics, duration,
sampling_rate=44100, amplitude=1):
    t = np.linspace(0, duration, int(sampling_rate * duration), endpoint=False)
    signal = amplitude * np.sin(2 * np.pi * frequency * t)

    for i in range(1, num_harmonics + 1):
        harmonic_amplitude = amplitude / (i + 1)
        harmonic_phase = 0
        signal += harmonic_amplitude * np.sin(2 * np.pi * (i + 1) * frequency *
t + harmonic_phase)

    return t, signal

def save_and_play_wave(signal, sampling_rate, filename="harmonic_wave.wav"):
    signal = signal / np.max(np.abs(signal))
    write(filename, sampling_rate, np.int16(signal * 32767))
    return ipd.Audio(filename)


t, signal = generate_wave_with_harmonics(220, 2, 3)
save_and_play_wave(signal, sampling_rate)
```