КОМП'ЮТЕРНІ НАУКИ

УДК 004.93:519.6:78.01 *http://dx.doi.org/10.30970/vam.2026.36.00000*

ГЕНЕРУВАННЯ МУЗИЧНИХ КОМПОЗИЦІЙ НА ОСНОВІ МАТЕМАТИЧНИХ ФУНКЦІЙ

І. Бернакевич, А. Дубровська

Львівський національний університет імені Івана Франка,
вул. Університетська 1, Львів, 79000, Україна,
e-mail: iryna.bernakevych@lnu.edu.ua, anastasiia.dubrovska@lnu.edu.ua

У статті розглянуто підхід до створення музичних композицій на основі математичних функцій. Актуальність дослідження зумовлена зростаючим інтересом до алгоритмічної композиції та використання математичних моделей для автоматизованого генерування музики за допомогою сучасних інформаційних технологій. Використання математичних методів у сфері комп'ютерної музики відкриває нові можливості для процесу створення музичних творів, а також для дослідження взаємозв'язку між математичними структурами та звуковими характеристиками. Запропоновано алгоритм генерування звукових сигналів, який перетворює значення математичної функції у послідовність звукових частот, формуючи мелодійну структуру відповідно до поведінки графіка функції. Такий підхід дає змогу інтерпретувати математичні залежності як музичні послідовності та створювати унікальні мелодії на основі різних типів функцій. Для підвищення виразності звукового сигналу застосовано додаткові аудіоефекти, зокрема амплітудну модуляцію, вібрато та інші параметри синтезу звуку, що дає змогу моделювати різні варіанти звучання та розширює можливості генерації музики. У межах дослідження розроблено вебзастосунок, який реалізує запропонований алгоритм та забезпечує зручне керування параметрами генерування музики і аудіоефектами. Користувач має можливість задавати математичні функції, змінювати параметри синтезу звуку, а також експериментувати з різними налаштуваннями для надання нового звучання музичним композиціям. Проведено аналіз існуючих середовищ для алгоритмічного генерування музики, зокрема Wolfram Cloud та Sonic Pi. Виконано порівняльний аналіз музичних композицій, згенерованих розробленим вебзастосунком та зазначеними середовищами, що дозволило оцінити особливості звучання, гнучкість налаштувань та можливості керування параметрами генерування. Отримані результати демонструють доцільність використання математичних функцій як формального апарату для алгоритмічного генерування музики та підтверджують ефективність запропонованого підходу у навчальних, дослідницьких та творчих застосуваннях.

Ключові слова: алгоритмічна композиція, математична функція, генерація музики, синтез звуку, аудіоефекти, Web Audio API.

1. Вступ

У сучасному світі активно розвиваються напрями, які поєднують науку, мистецтво та інформаційні технології. Одним із таких напрямів є генеративне мистецтво, де алгоритми використовуються для створення музики, графіки чи тексту [7]. Особливий інтерес викликає можливість створення музичних композицій на основі математичних функцій – як спосіб не лише візуалізувати, а й "озвучити" абстрактні математичні ідеї. Попри наявність окремих інструментів для звукової синтизації, більшість із них або складні у використанні, або не орієнтовані на інтерактивну

взаємодію з математичними функціями. Це створює потребу у простому, доступному інструменті, що поєднує графічну візуалізацію функцій із аудіовідтворенням, відкриваючи нові горизонти у сфері освіти, творчості та інклюзивності [5].

Об'єднання математичних моделей і музичних патернів не тільки дає змогу по-новому осмислити природу функцій, але й відкриває нові горизонти для креативного навчання, наукової візуалізації та цифрового мистецтва. Особливо актуальною розробка є для студентів, викладачів, митців і розробників, які прагнуть створювати взаємодійні ідеї на перетині різних дисциплін.

Новизна роботи полягає у розробці підходу до звукової інтерпретації математичних функцій, який базується на безпосередньому перетворенні їх числових значень у параметри аудіосигналу. На відміну від існуючих програмних засобів, орієнтованих переважно на створення музичних ефектів, запропонований підхід забезпечує більш тісний зв'язок між аналітичним виглядом функції та її звуковим представленням, що дає змогу використовувати аудіо як інструмент аналізу поведінки функцій.

2. Огляд сучасних підходів до генерування музики

Алгоритмічне генерування музичних композицій активно розвивається в межах комп'ютерної музики, цифрової обробки сигналів та штучного інтелекту. Перші формалізовані підходи до автоматизованого створення музики ґрунтувалися на математичних моделях і стохастичних процесах. Зокрема, використання ймовірнісних структур і марковських ланцюгів для побудови музичних послідовностей було продемонстровано в роботі Iannis Xenakis [14], який обґрунтував можливість застосування математичних закономірностей до композиційної техніки. Подальший розвиток ці ідеї отримали у межах алгоритмічної композиції, де музичний матеріал генерується на основі формальних правил, рекурсивних структур та функціональних залежностей [15].

Сучасні підходи до генерації музики можна умовно поділити на декілька напрямів: правилово-орієнтовані системи, стохастичні моделі, методи, засновані на математичних функціях і фрактальних структурах, нейромережеві моделі глибинного навчання та середовища live coding.

Правилово-орієнтовані системи передбачають використання гармонічних, ритмічних і структурних правил для побудови композицій. Такі підходи забезпечують контрольованість результату, проте часто обмежені за рівнем варіативності та потребують складного формалізованого опису музичної теорії [4, 11]. Стохастичні моделі, зокрема марковські процеси, урізноманітнюють музичний матеріал, однак не гарантують структурної цілісності композиції [2, 14].

Окремий напрям досліджень пов'язаний із використанням математичних функцій для генерування звукових сигналів. У таких підходах значення функції інтерпретуються як висота тону, амплітуда або тривалість звуку [8, 14]. Перевагою цього підходу є формальна визначеність і можливість чіткої математичної інтерпретації процесу генерації. Водночас більшість досліджень зосереджені на теоретичних аспектах [6, 10] і не пропонують універсальних алгоритмів інтеграції функціональних залежностей із засобами цифрового синтезу звуку.

Значного розвитку набула генерація музики із застосуванням методів штучного інтелекту. Нейромережеві архітектури (RNN, LSTM, Transformer) дозволяють моделювати стиль конкретних композиторів або створювати нові музичні структури на основі великих навчальних вибірок [3]. Найбільш відомими є системи, розроблені компаніями OpenAI (модель MuseNet) та Google Brain (платформа Magenta). Попри

високу якість згенерованих композицій, такі підходи характеризуються значною обчислювальною складністю, залежністю від навчальних даних і обмеженою прозорістю внутрішніх механізмів формування музичної структури.

Практичну реалізацію алгоритмічного створення музики забезпечують спеціалізовані програмні середовища. Зокрема, Wolfram Cloud [13] надає інструменти для математичного моделювання та синтезу аудіосигналів із використанням мови Wolfram Language. Середовище Sonic Pi [12] орієнтоване на live coding [9] і дає змогу генерувати музичні фрагменти шляхом програмування часових та частотних параметрів звуку. Вказані системи забезпечують гнучкість налаштувань та інтерактивність, проте вимагають від користувача знань програмування та не завжди передбачають безпосередню інтерпретацію математичної функції як цілісної мелодійної структури.

Таким чином, аналіз існуючих досліджень свідчить про наявність різноманітних підходів до алгоритмічного генерування музики, кожен з яких має свої переваги та обмеження. Водночас недостатньо дослідженим залишається питання розробки формалізованого алгоритму, що забезпечує пряме та кероване перетворення значень математичних функцій у параметри звукового сигналу з можливістю інтеграції аудіоефектів (амплітудної модуляції, вібрато, тембрових модифікацій) у межах єдиної програмної системи. Це зумовлює актуальність подальших досліджень у напрямі поєднання математичного апарату та цифрового синтезу звуку для створення керованих музичних композицій.

3. Математична постановка задачі

Нехай функція $f : \mathbb{R} \to \mathbb{R}$ визначена на відрізку $[a, b]$. Задача полягає у створенні музичної композиції на основі функції. Для цього необхідно здійснити математичну обробку даних [11], яка передбачає перетворення функції $f(x)$ у послідовність нот із частотами, амплітудами та аудіоефектами.

Обчислення значень функції. Розбиваємо інтервал $[a, b]$ на n рівних частин, обчислюємо значення

$$x_i = a + i \cdot \Delta x, \quad \Delta x = \frac{b-a}{n-1}, \quad i = 0, 1, \ldots, n-1 \tag{1}$$

та відповідні їм значення

$$y_i = f(x_i). \tag{2}$$

Рівномірний розподіл точок забезпечує однакову тривалість нот у музичній композиції.

Нормалізація отриманих значень. Передбачає приведення значень y_i до діапазону $[0, 1]$ за формулою

$$y_{norm,i} = \frac{y_i - y_{min}}{y_{max} - y_{min}}, \tag{3}$$

де $y_{min} = \min\{y_i \mid y_i \text{ є скінченним}\}$, $y_{max} = \max\{y_i \mid y_i \text{ є скінченним}\}$

Нормалізація дає змогу уніфікувати відображення на частоти незалежно від амплітуди функції. Якщо $y_{max} = y_{min}$, то $y_{norm,i} = 0$, що відповідає мінімальній частоті. Нормалізація застосовується до всіх y_i, навіть якщо функція має різкі піки.

Перетворення нормалізованих значень на частоти певного діапазону. Нормалізовані значення $y_{norm,i}$ відображємо на діапазон частот у межах

$[F_{min}, F_{max}]$ за формулою [7]

$$F_i = F_{min} + y_{norm,i} \cdot (F_{max} - F_{min}) \tag{4}$$

де $F_{min} = 220\,\text{Гц.}$, $F_{max} = 880\,\text{Гц}$.

Значення F_{min} та F_{max}, як правило, вибирають у межах [220, 880] Гц., що відповідає приблизно двом октавам (від A3 до A5) та забезпечує приємне звучання. Лінійне відображення $y_{norm,i} \in [0, 1]$ на [220, 880] гарантує пропорційність між значеннями функції та частотами.

Застосування амплітудної модуляції. Для моделювання цього ефекту використаємо першу похідну від функції $f'(x_i)$, яку апроксимуємо методом скінченних різниць

$$f'(x_i) \approx \frac{f(x_i + h) - f(x_i - h)}{2h}, \quad \text{де} \quad h = 0.001.$$

Амплітуду звуку A_i визначаємо відповідно до формули [11]:

$$A_i = \min\left(1, \frac{|f'(x_i)|}{S_A}\right), \tag{5}$$

де S_A – масштабуючий коефіцієнт (наприклад, $S_A = 10$).

Обґрунтуємо вибір саме таких параметрів. Перша похідна $f'(x_i)$ відображає швидкість зміни функції, що інтерпретується як динаміка звуку (гучність). Числова апроксимація через центральну різницю з кроком $h = 0.001$ забезпечує достатню точність для гладких функцій. Масштабуючий коефіцієнт S_A нормалізує значення $|f'(x_i)|$ таким чином, щоб $A_i \in [0, 1]$. Використання функції $\min(1, \cdot)$ запобігає надмірній гучності.

Застосування ефекту вібрато. Вібрато моделює частоту ноти на основі другої похідної функції, додаючи ефект тремоло. Для моделювання цього ефекту застосуємо другу похідну від функції, апроксимовану числовим методом

$$f''(x_i) \approx \frac{f(x_i + h) - 2f(x_i) + f(x_i - h)}{h^2}, \quad \text{де} \quad h = 0.001.$$

Функція $s(t)$ описує звуковий сигнал із реалізованим ефектом вібрато [10]. Вона базується на гармонічному сигналі сталої частоти, до фази якого додається періодична модуляція

$$s(t) = A_i \sin(2\pi F_i t + V_{d,i} \sin(2\pi V_{r,i} t)). \tag{6}$$

Тут $V_{d,i}$ – глибина вібрато, $V_{r,i}$ – частота вібрато. Параметр A_i визначає амплітуду сигналу та відповідає за гучність звуку. Зміна цього параметра впливає лише на інтенсивність звучання, не змінюючи висоту тону. Параметр F_i задає базову частоту ноти та визначає її основну висоту. За відсутності модуляції сигнал був би звичайною синусоїдою зі сталою частотою F_i. Для визначення глибини вібрато $V_{d,i}$ та частоти вібрато $V_{r,i}$ скористаємось відповідними формулами

$$V_{d,i} = \frac{|f''(x_i)|}{S_V}, \qquad V_{r,i} = R_0 + \frac{|f''(x_i)|}{S_R}, \tag{7}$$

де $S_V = 20, S_R = 5, R_0 = 5\,\text{Гц.}$ – базова частота вібрато.

Друга похідна $f''(x_i)$ відображає кривизну функції, що інтерпретується як інтенсивність вібрато. Апроксимація $f''(x_i)$ через другу центральну різницю з $h = 0.001$ є стандартною для числового диференціювання. Глибина вібрато $V_{d,i}$ масштабується через $S_V = 20$, щоб обмежити амплітуду модуляції частоти. Частота вібрато $V_{r,i}$ починається з базового значення $R_0 = 5\,\text{Гц}$. і зростає залежно від $|f''(x_i)|$. Для функцій із великими значеннями $|f''(x_i)|$ вібрато може бути надмірним, тому застосовується масштабування. Синусоїдальна модуляція $V_{d,i}\sin(2\pi V_{r,i}t)$ додає періодичне відхилення частоти, створюючи ефект вібрато.

Отже, ефект вібрато реалізується шляхом додавання до фази сигналу періодичної складової. Такий підхід відповідає фазовій модуляції, оскільки миттєва частота сигналу визначається похідною фази за часом. У результаті частота коливається навколо базового значення F_i.

Параметр $V_{d,i}$ визначає глибину вібрато, тобто амплітуду відхилення частоти від її основного значення. Зі збільшенням цього параметра зростає інтенсивність коливань висоти тону. Параметр $V_{r,i}$ визначає частоту вібрато, тобто швидкість періодичних змін висоти звуку. Він задає кількість коливань частоти за одиницю часу.

Застосування ефекту відлуння (Echo). Відлуння моделює природне відбиття звуку, додаючи затримані копії сигналу з поступовим загасанням амплітуди. Для моделювання цього ефекту додаємо до звуку затриманий сигнал із часом затримки T_{echo} (за замовчуванням $T_{echo} = 0.3$ с) та коефіцієнтом зворотного зв'язку β_{echo} (за замовчуванням$\beta_{echo} = 0.4$) [11], тобто

$$s_{echo}(t) = s(t) + \sum_{k=1}^{\infty} \beta_{echo}^{k} \cdot s(t - k \cdot T_{echo}). \tag{8}$$

Вибір параметрів зумовлений тим, що час затримки $T_{echo} = 0.3\,\text{с}.$ відповідає короткому відлунню, придатному для музичних ефектів, а коефіцієнт зворотного зв'язку $\beta_{echo} = 0.4$ забезпечує помірне загасання, що дає змогу уникнути надмірного накопичення сигналу. Сума обривається після кількох ітерацій(зазвичай $k \leq 5$), коли β_{echo}^{k} стає незначним

Застосування гармонізації. Гармонізація додає додаткові ноти, зміщені на задану кількість півтонів, створюючи акорди [11]. Для кожної ноти з частотою F_i додаємо гармонійні ноти з частотами, зміщеними на k-півтонів, тобто

$$F_{harmonic,i,k} = F_i \cdot 2^{k/12}, \tag{9}$$

де $k \in \{4, 7\}$ для мажорної гармонії (терція та квінта), $k \in \{3, 7\}$ для мінорної, $k \in \{4, 7, 10\}$ для септакорду. За зміщення на k півтонів відповідає множник $2^{k/12}$, оскільки 12 півтонів складають октаву.

Застосування арпеджіатора. Арпеджіатор розбиває акорд на послідовні ноти, створюючи ефект швидкого перебору та відтворює ноти акорду послідовно з заданим інтервалом. Це перетворення передбачає групування нот в акорди розміром m, для кожного акорду $\{F_i, F_{i+1}, \ldots, F_{i+m-1}\}$ відтворюємо ноти послідовно з інтервалом порядку, визначеному патерном (вгору, вниз, вгору-вниз, випадковий)

$$t_{i,j} = t_i + j \cdot T_{arp}, \quad j = 0, 1, \ldots, m-1, \tag{10}$$

де $T_{arp} = 0.1$ с, $m = 3$ (розмір акорду), t_i - час початку акорду.

Патерни (вгору, вниз, вгору-вниз, випадковий) визначають порядок нот: наприклад $F_i, F_{i+1}, \ldots, F_{i+m-1}$ – вгору, $F_{i+m-1}, \ldots, F_{i+1}, F_i$ – вниз.

Зміна тону. Зміна тону масштабує частоту кожної ноти відповідно до формули

$$F_{shifted,i} = F_i \cdot 2^{p/12}, \quad p \in [-12, 12]. \tag{11}$$

За зміщення на p півтонів відповідає множник $2^{p/12}$, де $p = 12$ підвищує ноту на октаву. Діапазон $p \in [-12, 12]$ дає змогу зміщувати тон на ± 1 октаву.

Низькочастотний осцилятор (LFO). LFO модулює частоту відсікання низькочастотного фільтра для створення ефекту "пульсації" [11].

$$F_{filter}(t) = F_{base} + D_{LFO} \cdot \sin(2\pi R_{LFO} t). \tag{12}$$

Тут $F_{base} = 5000\,\text{Гц}$. – базова частота підходить для низькочастотного фільтра, який пропускає більшість частот нот. Глибина $D_{LFO} = 300$ і швидкість $R_{LFO} = 0,5\,\text{Гц}$. створюють помітний, але неагресивний ефект.

Відтворення звуку. Створюємо послідовність нот із частотами F_i, тривалістю T (за замовчуванням $T = 0.2\,\text{с}.$), амплітудою A_i, із можливим вібрато $V_{d,i}, V_{r,i}$.

Візуалізація функції. Будуємо графік функції $f(x)$ на відрізку $[a, b]$ за координатами обчислених точок (x_i, y_i).

4. Проєктування системи та програмна реалізація

4.1. Інтерфейс користувача

Для практичного застосування описаного алгоритму генерування музики було створено вебзастосунок [1], інтерфейс якого наведено на рис. 1.

Меню зверху надає такі можливості: *Генератор* – для створення музичної композиції, *Приклади* математичних функцій, де наведено їхні графіки, *Інформація* про вебзастосунок та *Інструкції* з його використання. Розглянемо детальніше кожен пункт меню.

Генератор. Користувач може ввести математичну функцію у спеціальне текстове поле в розділі "Генератор". Після натискання кнопки "Генерувати" система обробляє введену функцію, обчислює її значення на заданому інтервалі (за замовчуванням $[-10, 10]$), перетворює ці значення на частоти звукових нот (у діапазоні $220 - 880\,\text{Гц}.$) і генерує аудіодані. Процес передбачає нормалізацію значень функції і можливість додавання модуляції гучності (на основі першої похідної від функції) та вібрато (на основі другої похідної від функції) за допомогою відповідних прапорців. Користувач може відтворити згенеровану композицію, натиснувши кнопку "Відтворити". Аудіо відтворюється через Web Audio API з плавними огинаючими для гучності та підтримкою вібрато. Доступні кнопки "Пауза" (призупиняє відтворення), "Стоп" (повністю зупиняє та скидає прогрес) і "Зберегти аудіо" (експортує композицію у форматі WAV для локального завантаження). Під час відтворення відображається прогрес-бар із поточним і загальним часом, а також анімація курсору на графіку, що вказує на поточну позицію. Користувач може налаштувати діапазон значень x (наприклад, від -10 до 10) через поля введення, кількість точок (за замовчуванням 100 точок), та тривалість кожної ноти (за замовчуванням 0.2 секунди) за допомогою відповідних бігунків. Кнопка "Застосувати" оновлює аудіодані з новими параметрами.

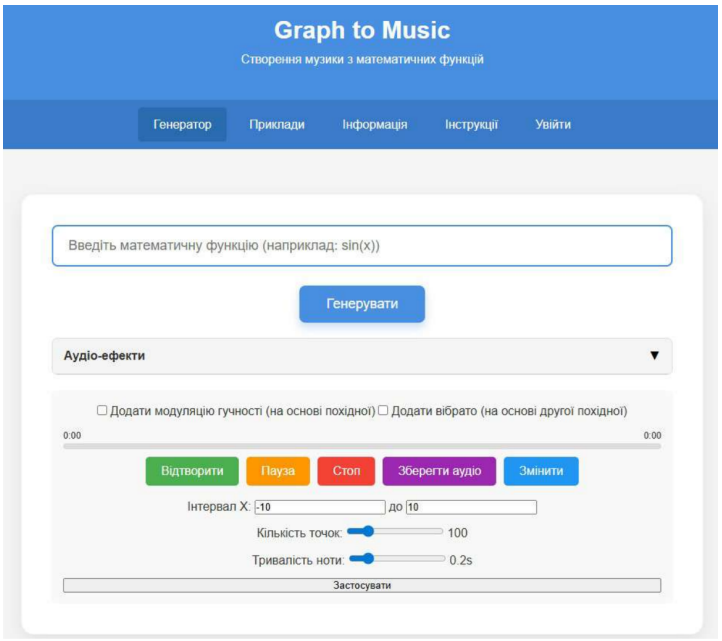

Рис. 1. Інтерфейс вебзастосунку

Користувач може активувати ефекти через інтерфейс, який містить розкривну панель із параметрами (рис. 2):

- Відлуння (Echo): додає затримку звуку з налаштуванням часу (0.1-1 с.) і зворотного зв'язку (0.1-0.9).
- Гармонізація: додає гармонійні ноти (мажорна, мінорна або септакорд) на основі інтервалів (терція, квінта).
- Арпеджіатор: відтворює ноти акорду послідовно з патернами (вгору, вниз, вгору-вниз, випадково) і регулюванням швидкості (0.05-0.5 с).
- Зміна тону: зсуває висоту тону на ± 12 півтонів.
- LFO-фільтр: модулює частоту фільтра з налаштуванням швидкості (0.1-5 Гц.) і глибини (50-1000).
- Форма хвилі: дає змогу вибрати тип осцилятора (синусоїда, квадратна, пилкоподібна, трикутна).

Приклади. Система надає користувачам готові приклади функцій для швидкого старту (див. рис. 3). Кожен приклад містить такі компоненти: назва функції, її формула, графік, опис та кнопку "Спробувати" для завантаження функції у генератор для подальшої обробки. Після завантаження прикладу користувач може скористатися функціоналом генератора, описаним вище.

4.2. Використані технології

Проєкт реалізовано з використанням клієнт-серверної архітектури. Для реалізації клієнтської частини вибрано стек технологій HTML5, CSS3 та JavaScript (ES6+), яка є основною мовою програмування для реалізації клієнтської логіки. Для відображення графіків функцій використано бібліотеку Plotly.js, яка спеціалізується

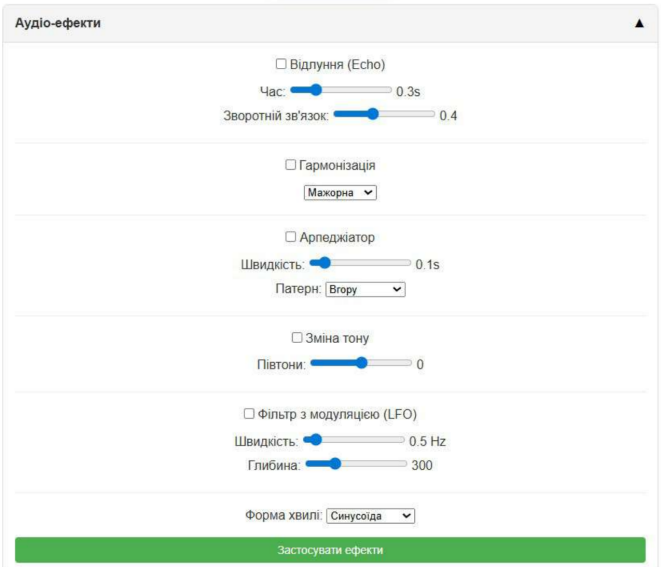

Рис. 2. Налаштування аудіоефектів

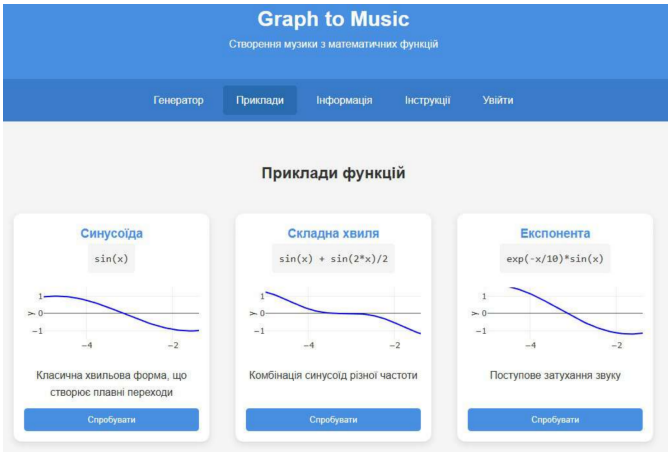

Рис. 3. Приклади функцій

на математичних та наукових візуалізаціях, що ідеально підходить для відображення графіків математичних функцій з можливістю масштабування та інтерактивності. Web Audio API є ключовою технологією для генерування звуку в реальному часі, надає низькорівневий доступ до аудіообробки в браузері, дає змогу створювати складні звукові ефекти, осцилятори та фільтри. Вибір цієї технології забезпечує високу якість звуку та можливість реалізації складних аудіоалгоритмів не залежно від зовнішніх плагінів. Серверну частину побудовано на базі Node.js із використанням фреймворку Express.

4.3. Архітектура взаємодії математичного модуля з аудіоінтерфейсом

Розглянемо архітектуру взаємодії математичного модуля з аудіоінтерфейсом (див. рис. 4). Після введення користувачем математичної функції $f(x)$ здійснюємо її обробку у математичному модулі, тобто виконуємо розбір формули, перевірку синтаксису та подальшу дискретизацію значень функції на заданому інтервалі $[a, b]$ (1-2). Далі нормалізуємо значення функції відповідно до формули (3), які опісля перетворюємо у частоти звуку за формулою (4).

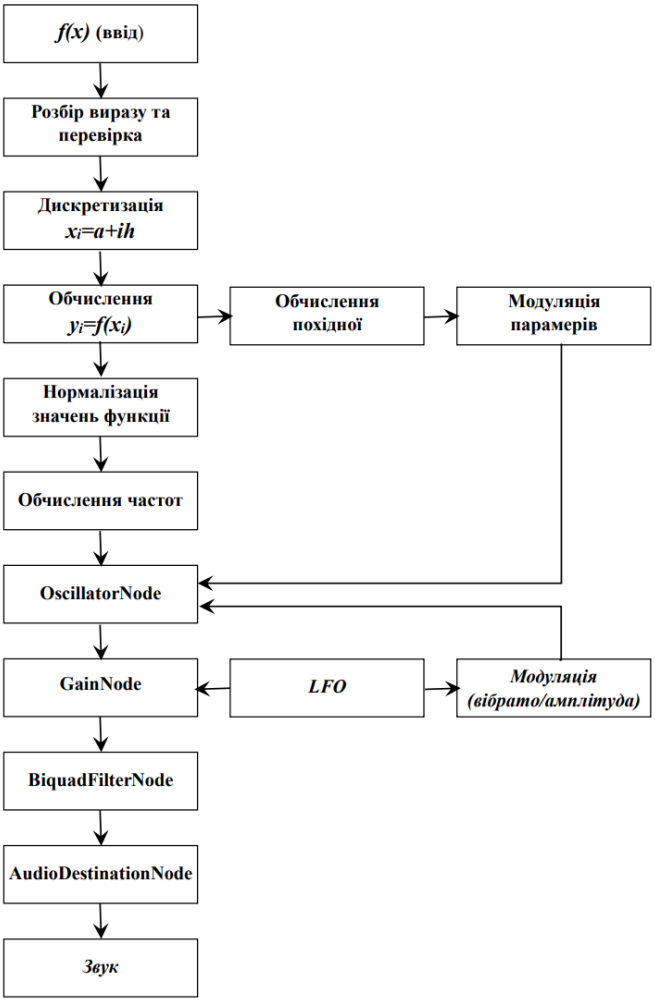

Рис. 4. Архітектура взаємодії математичного модуля з аудіоінтерфейсом

Отримані значення передаємо до аудіоінтерфейсу, реалізованого з використанням Web Audio API. Генерація звуку здійснюється за допомогою вузла **OscillatorNode**, на вхід якому динамічно у реальному часі подаються частоти F_i, відповідні

значенням функції $y_{norm,i}$. Сформований сигнал має вигляд:

$$(s(t) = \sin(2\pi f_i t).$$

Надалі обробка сигналу виконується вузлом **GainNode**, який використовується для керування амплітудою сигналу:

$$s_{out}(t) = A(t) \cdot s(t)$$

Для додаткової обробки застосовується вузол **BiquadFilterNode**, який дає змогу реалізувати фільтрацію сигналу та створення ефектів, зокрема за рахунок модуляції частоти зрізу:

$$F(t) = F_{base} D \cdot \sin(2\pi R t)$$

Кінцевий сигнал передається до вузла **AudioDestinationNode**, що забезпечує його відтворення через аудіопристрої користувача. Результати обчислення похідних, описані у розділі 3, інтегруються у програмну реалізацію як додатковий параметр керування звуком. Похідна обчислюється чисельно та використовується для модулювання параметрів аудіосигналу. Зокрема, значення похідної можуть впливати на:

- амплітуду сигналу через GainNode, що дає змогу відобразити швидкість зміни функції у вигляді змін гучності;
- частоту сигналу через OscillatorNode, що створює додаткову динаміку звучання.

Таким чином, похідна функції не лише обчислюється, але й безпосередньо інтегрується у процес генерації звуку, що забезпечує більш глибоке відображення поведінки функції у звуковій формі. Для кращого розуміння описаної архітектури у роботі наведено схему, яка відображає послідовність перетворення математичної функції у звуковий сигнал, а також взаємодію основної гілки обчислень із додатковими модулями, такими як обчислення похідних та модуляція сигналу через LFO (див. рис. 4).

5. Порівняльнимй аналіз програмних засобів генерування музики

У процесі розробки вебзастосунку було проаналізовано сучасні програмні засоби, зокрема Wolfram Mathematica (Cloud) та Sonic Pi, які частково реалізують функціональність, пов'язану з математичною візуалізацією або генерацією звуку на основі числових даних. Кожен із зазначених інструментів реалізує окремі аспекти задачі, але жоден із них не забезпечує повноти підходу, який поєднує інтерактивну побудову графіка математичної функції та генерацію музики у вебсередовищі.

Wolfram Mathematica Wolfram Mathematica, а саме Wolfram Cloud [13] є сильною обчислювальною системою, яка вміє розв'язувати рівняння, будувати графіки, аналізувати числа. Платформа може перетворювати псевдокод, у якому містяться реалізовані системою методи для звучання мелодії та сама математична функція, у звуковий сигнал або ж мелодію. Для порівняння візьмемо функцію $y = \sin(x)$. Для того, щоб почути цю функцію у Wolfram Cloud, необхідно написати такий код:

```
EmitSound[
 Play[
  Sin[2 Pi (440 + 100 Sin[t]) t],
  {t, 0, 4}
 ]
]
```

У коді використано базові функції програми: EmitSound – програє звук, створений Play, через динаміки або навушники; Play – створює звук на основі математичної функції у діапазоні часу t. Звучання цієї функції буде з основною частотою 440 Гц., на який накладено вібрато ± 100 Гц.

Щоб згенерувати мелодію за цією ж функцію у створеному вебзастосунку, достатньо ввести у необхідне поле функцію та натиснути кнопку "Генерувати" (рис. 5). Отже, можна зробити висновок, що використання створеної вебаплікації є простішим і не вимагає від користувача знань у сфері програмування.

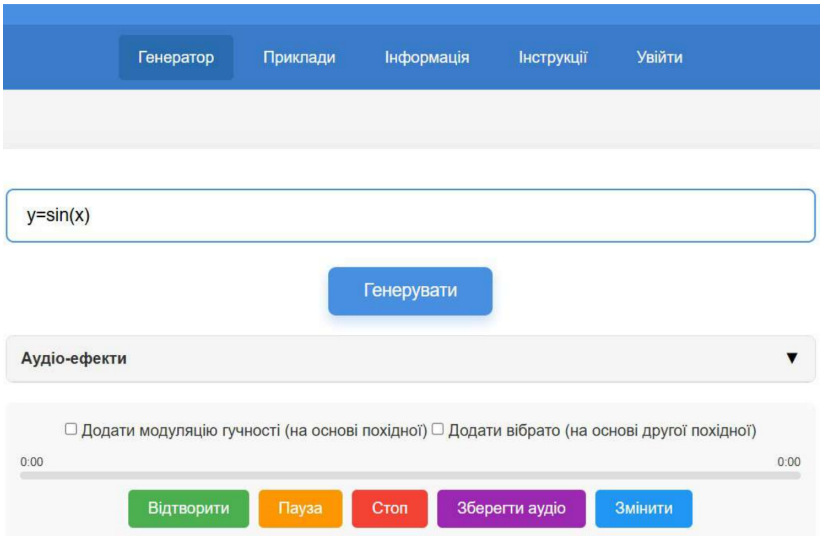

Рис. 5. Приклад використання

Розглянемо складнішу функцію $f(x) = e^{-0.2x}\sin(5x + 2\sin(3x))$ з додатковими ефектами. Опис у Wolfram Cloud буде достатньо складним (див. код нижче), а згенерована мелодія нагадує звук згинання-розгинання металевої пластини.

```
EmitSound[
  Play[
   Module[
    {
    f, carrier, tremolo, arp, echo
    },
   (* Складена математична функція *)
    f[t_] := Exp[-0.2 t] Sin[5 t + 2 Sin[3 t]];
   (* Перетворюємо значення функції у частоту (арпеджіатор вгору) *)
```

```
    arp = 440 + 120 f[t];
   (* Основний сигнал *)
    carrier = Sin[2 Pi arp t];
   (* Модуляція гучності (tremolo 6\,Гц.) *)
    tremolo = 0.6 (1 + Sin[2 Pi 6 t]);
   (* Ехо із затримкою 0.4 секунди *)
    echo = 0.5 Sin[2 Pi (440 + 120 f[t - 0.4]) (t - 0.4)];
    tremolo carrier + echo
   ],
 {t, 0, 6},
 SampleRate -> 44100
 ]
]
```

Використання цієї ж функції з тими ж параметрами у генераторі (див. рис. 6), створює мелодію, подібну до будильника на телефоні.

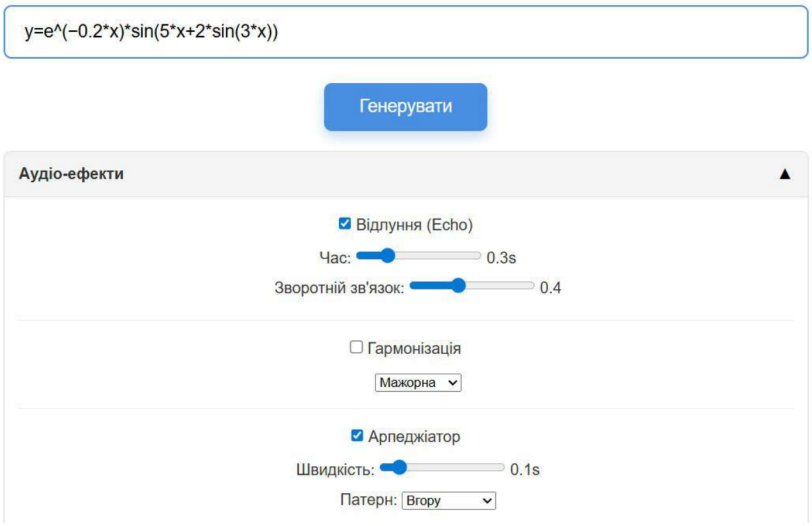

Рис. 6. Приклад використання

Sonic Pi Застосунок Sonic Pi [12] передбачає програмне керування нотами, частотами, ритмом і ефектами. Sonic Pi обчислює значення функції, перетворює їх у частоти або ноти та відтворює послідовно. Тобто це музичний live-coding інструмент. Перевагою цього застосунку є підтримка алгоритмічного синтезу музики на основі програмного коду, що дає змогу використовувати математичні вирази для керування звуковими параметрами.

Як і в попередньому випадку, згенеруємо в Sonic Pi мелодію для функції $y = \sin(x)$. Для цього необхідно написати код

```
use_synth :sine
 live_loop :sin_x do
 t = tick * 0.02
```

```
 value = Math.sin(t)
 frequency = 440 + 200 * value
 synth :sine, freq: frequency, sustain: 0.3, release: 0.1, amp: 1
 sleep 0.3
end
```

Тут використано функції Sonic Pi: synth – відтворює звук заданого типу синтезатора на вказаній частоті; tick – лічильник кроків циклу, використовується для дискретного збільшення аргументу функції; sleep – задає тривалість звучання кожного дискретного кроку та інтервал між відтворенням наступного звуку; live_loop – організовує повторюваний цикл обчислення та програвання звуку, що дає змогу відтворювати функцію послідовно в часі; amp, sustain, release – параметри, що регулюють гучність та форму звукового сигналу на кожному кроці.

Отриманий аудіо сигнал сприймається швидше як безперервний ритмічний стукіт, подібний до частого постукування пальцями, ніж як плавне хвилеподібне звучання синуса. Навіть після спроб різних модифікацій коду характер звучання залишився незмінним. У даному випадку створений генератор звуку є більш ефективним для сприйняття синусоїдальної функції, оскільки забезпечує відчутний перехід від вищих до нижчих частот, що відповідає формі графіка функції. Таким чином слухове сприйняття прямо відображає візуальну структуру синусоїди, дозволяючи краще усвідомити її хвилеподібну природу.

Тепер візьмемо складнішу функцію, яка використовувалась у попередніх дослідженнях, $f(x) = e^{-0.2x}\sin(5x + 2\sin(3x))$ з додатковими звуковими ефектами відлуння, арпеджіатор(вгору) та модуляція гучності

```
use_synth :sine
 live_loop :complex_wave do
 t = tick * 0.05
 f = Math.exp(-0.2 * t) * Math.sin(5*t + 2*Math.sin(3*t))
 freq = 440 + 200 * f
 with_fx :echo, mix: 0.3, phase: 0.25 do
  with_fx :tremolo, mix: 0.2, wave: 3 do
   [0, 2, 4].each do |i|
    synth :sine, freq: freq * (1.0 + i*0.03), sustain: 0.1,
                        release: 0.05, amp: 0.6
   end
  end
 end
 sleep 0.1
end
```

Згенерована мелодія нагадує звук завершення виклику на старих аналогових телефонах, однак додаткові ефекти створюють відчуття приглушеності сигналу. Ефекти відлуння та модуляції гучності у Sonic Pi відтворюються за тією ж самою логікою, що й у створеному генераторі – суть звучання залишається однаковою, проте в Sonic Pi вона реалізована через готові аудіофільтри, а в генераторі – через пряме математичне керування амплітудою та частотою сигналу.

Sonic Pi дає змогу відтворювати ефекти відлуння, арпеджіо та модуляції гучності, проте самі математичні функції звучать не так, як очікувалося: хвилеподібна

структура сигналу не відчувається чітко, і звук сприймається швидше як ритмічний або текстурний ефект, ніж як відтворення самої функції.

Отже, на основі аналізу генерування мелодій у середовищах Wolfram та Sonic Pi можемо зробити висновок, що використання сучасних програм для генерування мелодій часто вимагають значних зусиль та знань, а згенеровані мелодії не відповідають очікуванням. Створений вебзастосунок позбавлений цих недоліків. Скористатись генератором може будь-хто без попередніх знань в галузі програмування, а згенеровані мелодії є більш релевантними для відповідних математичних функцій.

6. Тестування та оцінка ефективності

З метою перевірки працездатності та надійності розробленого вебзастосунку було проведено технічне тестування, спрямоване на перевірку коректності роботи системи, її стійкості до некоректних вхідних даних, стабільності при тривалій генерації звуку, а також на перевірку окремих програмних модулів. Тестування проводилось у браузерному середовищі з використанням Chrome DevTools.

Перевірка стійкості до некоректних вхідних даних. Було перевірено роботу системи при введенні некоректних математичних виразів, зокрема синтаксичних помилок, незакритих дужок, використання недопустимих символів, ділення на нуль, чи введення порожнього виразу. У таких випадках застосунок не запускає процес генерування звуку та інформує користувача про некоректність введених даних. Це підтверджує стійкість системи до некоректних вхідних даних і запобігає аварійному завершенню роботи програми.

Тестування продуктивності. Метою цього етапу було встановити, як зміна кількості точок дискретизації функції впливає на якість звукового відтворення та швидкодію системи. Встановлено, що зі збільшенням кількості точок зростає точність відтворення звуку, однак це призводить до збільшення обчислювального навантаження на браузер. За оптимально підібраних параметрів (500 – 1500 точок дискретизації) забезпечується баланс між якістю звуку та продуктивністю системи.

Тестування стабільності та тривалої роботи. Перевірено роботу застосунку при тривалому генеруванні звуку. У процесі тестування оцінювалась відсутність зависань, збоїв генерування, переривання аудіосигналу та накопичення помилок у браузері. Результати показали, що за умови використання оптимальних параметрів генерування система працює стабільно протягом тривалого часу, без аварійного завершення роботи.

Оцінювання плавності звуку. Плавність відтворення звуку забезпечується за рахунок безперервної передачі значень параметрів до вузлів Web Audio API. Для перевірки плавності звучання було протестовано як прості функції, так і складні вирази, що містять вкладені тригонометричні залежності, експоненціальні складові та додаткові параметри модуляції. У результаті встановлено, що при помірній складності функції та достатній кількості точок дискретизації звук зберігає плавний характер. При значному ускладненні обчислень або надмірній кількості точок можуть з'являтися незначні перебої, пов'язані з обчислювальними обмеженнями браузерного середовища. Отже, плавність звучання забезпечується за умови оптимального вибору параметрів дискретизації.

Тестування окремих модулів. Окрему увагу приділено тестуванню математичного модуля. Проведено перевірку коректності обчислення значень функцій та їх похідних шляхом порівняння з аналітичними результатами. Отримані результати підтверджують правильність реалізації алгоритмів.

Проведене тестування показало, що вебзастосунок є стійким до некоректних вхідних даних, зберігає працездатність при тривалій генерації звуку та забезпечує задовільну плавність відтворення за умови оптимального вибору кількості точок дискретизації.

7. Висновки

У межах даного дослідження запропоновано алгоритм генерування музичних композицій на основі математичних функцій. Розроблений алгоритм покладено в основу вебзастосунку [1], який дає змогу створювати унікальні музичні композиції на основі математичних функцій, візуалізувати їх у вигляді графіків, застосовувати різноманітні аудіоефекти та зберігати отримані результати. Проведене тестування підтвердило стійкість роботи системи за некоректних вхідних даних та стабільність при тривалій генерації звуку.

Проведено порівняльний аналіз згенерованих композицій з іншими програмними середовищами [12, 13]. Встановлено, що в більшості таких середовищ звукові ефекти, зокрема відлуння, арпеджіатор або модуляція гучності, реалізуються за допомогою стандартних аудіооб'єктів і процедур цифрової обробки сигналу. При цьому вони не мають прямого математичного відповідника функції, яка генерує базовий звук, що ускладнює можливість чіткого відстеження математичної структури функції через аудіосигнал. Таким чином, у традиційних музичних середовищах ефекти накладаються на сигнал переважно на рівні музичної обробки, а не на рівні математичної моделі. У результаті відтворення функції у вигляді звуку стає менш точним з точки зору її числових та аналітичних характеристик.

Запропонований вебзастосунок позбавлений зазначених недоліків, оскільки процес формування звуку безпосередньо пов'язаний із математичною функцією, що дає змогу більш точно відобразити її структуру в аудіальному представленні та забезпечує можливість дослідження взаємозв'язку між математичною моделлю та отриманим музичним результатом.

Практичне значення отриманих результатів полягає у можливості використання розробленого застосунку в освітньому процесі для демонстрації взаємозв'язку між математичними функціями та звуковими сигналами, а також у галузях алгоритмічної композиції, генеративної музики та інтерактивних мультимедійних систем.

Список використаної літератури

1. GraphToMusic [Electronic resource]. URL: https://github.com/anastasiia-dubrovska/GraphToMusic.
2. Ames C. The Markov process as a compositional model: a survey and tutorial / C. Ames // Computer Music Journal. – 1989. – Vol. 13, № 4. – P. 42–62.
3. Briot J.-P. Deep learning techniques for music generation – a survey / J.-P. Briot, G. Hadjeres, F. Pachet // arXiv preprint arXiv:1709.01620. – 2017. – 35 p.
4. Cope D. Experiments in Musical Intelligence. / D. Cope. – Middleton, Wisconsin: A-R Editions, 1996. – 263 p.
5. Dobosz K. Audible Charts of Mathematical Functions / K. Dobosz, D. Hanak // Computers Helping People with Special Needs. ICCHP 2024. Lecture Notes in Computer Science. – Vol. 14750. – Cham: Springer. – DOI: https://doi.org/10.1007/978-3-031-62846-7_25.
6. Herremans D. A Functional Taxonomy of Music Generation Systems / D. Herremans, C.-H. Chuan, E. Chew // ACM Computing Surveys (CSUR). – Vol. 50, Iss. 5. – Article № 69. – P. 1–30. – DOI: https://doi.org/10.1145/3108242.

7. Hermann T. Taxonomy and Definitions for Sonification and Auditory Display / T. Hermann // In: Hermann T., Hunt A., Neuhoff J.G. (eds) The Sonification Handbook. – DOI: https://doi.org/10.1007/978-3-540-49191-9_1.
8. Grond F. Singing function / F. Grond, T. Hermann // J Multimodal User Interfaces. – 2012. – Vol. 5. – P. 87–95. – DOI: https://doi.org/10.1007/s12193-011-0068-2.
9. McLean A. Making Programming Languages to Dance to / A. McLean // FARM '14: Proceedings of the 2nd ACM SIGPLAN international workshop on Functional art, music, modeling & design. – September 6, 2014. – Gothenburg, Sweden. – 2014. – P. 63–70.
10. Nierhaus G. Algorithmic Composition: Paradigms of Automated Music Generation / G. Nierhaus. – Vienna: Springer, 2009.
11. Roads C. The Computer Music Tutorial / C. Roads. – Cambridge, MA: MIT Press, 1996. – 1234 p.
12. Sonic Pi [Electronic resource]. – URL: https://sonic-pi.net.
13. Wolfram Cloud [Electronic resource]. – URL: https://www.wolframcloud.com/.
14. Xenakis I. Formalized music: thought and mathematics in composition / I. Xenakis. – Revised ed. Stuyvesant. – NY: Pendragon Press, 1992. – 387 p.
15. Zhang K. Data Melodification FM: Where Musical Rhetoric Meets Sonification. – 2025. – / K. Zhang, D. Grellscheid, L. Garrison. – DOI: https://doi.org/10.48550/arXiv.2510.00222 .

Стаття: надійшла до редколегії 22.01.2026
доопрацьована 26.02.2026
прийнята до друку 03.03.2026

GENERATION OF MUSICAL COMPOSITIONS BASED ON MATHEMATICAL FUNCTIONS

I. Bernakevych, A. Dubrovska

Ivan Franko National University of Lviv,
1, Universytetska str., 79000, Lviv, Ukraine,
e-mail: *iryna.bernakevych@lnu.edu.ua*, *anastasiia.dubrovska@lnu.edu.ua*

The paper considers an approach to generating musical compositions based on mathematical functions. The relevance of the research is determined by the growing interest in algorithmic composition and the use of mathematical models for automated music generation with the help of modern information technologies. The application of mathematical methods in the field of computer music opens new possibilities for formalizing the process of music creation and for studying the relationship between mathematical structures and sound characteristics. An algorithm for sound signal generation is proposed, which converts the values of a mathematical function into a sequence of sound frequencies, forming a melodic structure according to the behavior of the function graph. This approach makes it possible to interpret mathematical dependencies as musical sequences and to generate unique melodies based on different types of functions. To enhance the expressiveness of the generated sound, additional audio effects are applied, including amplitude modulation, vibrato, and other sound synthesis parameters. These effects make it possible to model different sound variations and expand the capabilities of music generation. Within the research, a web application implementing the proposed algorithm has been developed. The system provides convenient control over music generation parameters and audio effects. Users can define mathematical functions, modify sound synthesis parameters, and experiment with different settings to produce new sound variations in musical compositions. An analysis of existing environments for algorithmic music generation, particularly Wolfram Cloud and Sonic Pi, was also conducted. A comparative analysis of musical compositions generated by the developed web application and the mentioned environments was performed, which made it possible to evaluate sound characteristics, flexibility of settings, and

the possibilities of controlling generation parameters. The obtained results demonstrate the feasibility of using mathematical functions as a formal framework for algorithmic music generation and confirm the effectiveness of the proposed approach in educational, research, and creative applications.

Key words: algorithmic composition, mathematical functions, music generation, sound synthesis, audio effects, Web Audio API.