e-ISSN: 2686-2573
DOI: 10.21512/emacsjournal.v7i1.12562

Developing Algorithm of Music Concepts and Operations Using The Modular Arithmetic

Kelvin Minor

Mathematics Department, School of Computer Science,
Bina Nusantara University,
Jakarta, Indonesia 11480
kelvin.minor@binus.ac.id;

Correspondence: kelvin.minor@binus.ac.id

Abstract – *The rapid development of digital music technology is closely intertwined with advancements in both music theory and mathematical formalism. This study aims to bridge the gap between these fields by exploring how mathematical concepts can enhance the understanding and analysis of music theory. Specifically, the research focuses on the application of modular arithmetic to analyze the circular structure of the chromatic scale, a key concept in music. Modular arithmetic enables the identification of patterns in pitch relationships and the manipulation of musical elements like transposition and interval calculations. In addition to modular arithmetic, the study also highlights the role of regular expressions in music theory. Regular expressions provide powerful tools for pattern matching, which can be applied to recognize and categorize musical components, such as enharmonic equivalents (notes that sound the same but are named differently). These tools allow for the development of algorithms capable of generating chords from given notes or identifying chords from existing sets of notes. By integrating modular arithmetic and regular expressions, the study proposes a framework for developing mathematical models and algorithms to facilitate digital music analysis. This approach not only enhances the theoretical understanding of music but also holds practical applications in digital music production and education.*

Keywords: music theory, enharmonic equivalents, harmony analysis, mathematical modeling, modular arithmetic

I. INTRODUCTION

The development of digital music technology is progressing rapidly, supported by advancements in algorithmic and programming applications (Gorgoglione, Garavelli, Panniello, & Natalicchio, 2023). By bridging the gap between the music industry and academic research, we can enhance the potential for significant advancements and improvements in digital music technology (Pöpel & Jürgens, 2022). Providing a foundation for learning music theory may empower musicians and researchers to understand the theoretical aspects of music, such as pitch, harmony, and composition (Hess, 2020). Mathematical formalism plays a crucial role in this process providing the exploration of music in mathematical terms, aiding in the formalization of musical structures (Mannone, 2021).

This study explores the relationship between mathematics and music in depth, highlighting how various mathematical concepts can deepen our understanding of music. It examines how mathematics explains the principles of pitch and harmony, providing precise language for describing the relationship between pitches. By integrating mathematical tools, the study enhances the theoretical framework of music and offers practical applications in digital music production and education. The study aims to bridge the gap between music theory and mathematical formalism, showcasing their synergy and potential to innovate digital music technology. Specifically, it provides mathematical models and algorithms for composing music, analyzing musical structures, and demonstrating these concepts' practical applications in digital music technology.

II. METHODS

This section provides a complete and detailed description of the steps undertaken in conducting research as shown in **Figure 1**. The research started with reviewing journals to collect information of musical concepts and mathematical concepts. The next step is formulating all musical concepts using mathematical equations, and then translating all mathematical equations and all musical analysis algorithms to programming language. The last step is validating the system by doing conduct testing and validation.

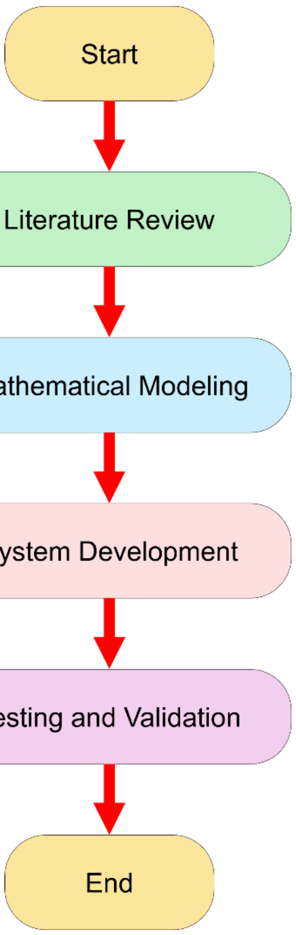


Figure 1. Research Method

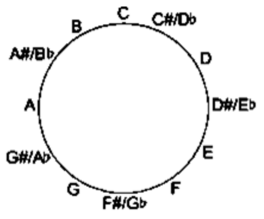
1. Literature Review

This phase collects all the information from scholarly sources, including books and journal articles, to support the system's implementation. The literature review for this study encompasses a comprehensive examination of music theory, mathematical concepts, and the use of regular expressions. To accurately model and manipulate musical elements programmatically, the study delves into music theory, such as musical notes, scales, chords, and intervals. Furthermore, the study of mathematics is applied to the development of the algorithms for semitone note shifting, interval calculations, chord notes generation and chord identification. Regular expressions are included in the literature review as well, which is essential for the identification of enharmonic equivalents. Then the problem analysis process will be performed when all the information has been collected to produce problem formulations.

This phase provides a comprehensive account of the knowledge related to the algorithm. It covers foundational theory, such as

- **Chromatic Scale and Enharmonic Equivalent**

The chromatic scale consists of 12 notes that are a semitone (half step) apart (Palmer, Manus, & Lethco, 1994). These notes are named as follows: C, C#/Db, D, D#/Eb, E, F, F#/Gb, G, G#/Ab, A, A#/Bb, B. After B, the pattern repeats with C again. This repetition is what creates the circular nature of the notes as shown in **Figure 2**. This representation may help musicians and researchers visualize key relationships, intervals, and modulations within the twelve-tone equal temperament system (Durfee & Colton, 2015).



- **Figure 2**. Chromatic Circle

- **Modular Arithmetic**

Modular arithmetic is a system of arithmetic for integers, where numbers wrap around when they reach a certain value (Irving, 2004). Understanding the Division Theorem provides a solid foundation for learning modular arithmetic. The Division Theorem states that for two positive integers a and b, there exist unique nonnegative integers q and r, with $r < a$, as shown in **Equation (1)**. This theorem provides a fundamental understanding of how integers can be divided to produce a quotient q and a remainder r. When a is a negative integer, the quotient q may also be negative, but the remainder r must still be non-negative and less than a.

$$b = a \times q + r \qquad (1)$$

Modular arithmetic relies on the concept of remainders, which is directly derived from the Division Theorem in **Equation (1)**. In modular arithmetic, we focus on the remainder r when an integer b is divided by another integer a. In this way, two integers b and r are said to be congruent modulo a if they have the same remainder when divided by a as shown in **Equation (2)**.

$$b = r \ (mod\ a) \qquad (2)$$

Applying the concept of modular arithmetic in **Equation (2)** can lead to the creation of a new set of integers known as the set of equivalence classes modulo a, denoted as $\mathbb{Z}_a$. The set $\mathbb{Z}_a$ denotes the set of integers modulo a, where $\mathbb{Z}_a = \{0,1,2,3,4,5,\dots,a-1\}$. Therefore, $\mathbb{Z}_a$ is a finite group of order a, that satisfies four properties (Baumslag & Chandler, 1968). A finite group must satisfy closure property which ensures that combining any two elements in the group under the operation yields another element in the same group. A finite group must satisfy associativity property which ensures that the way operations are grouped does not affect the outcome. A finite group must have an identity element that acts as a neutral element under the group operation, meaning it leaves other elements unchanged when combined with any element. A finite group must have an inverse element that acts as a counterpart element under the group operation, meaning it produces the identity element when combined with any element.

- **Regular Expression**

Regular Expression, often abbreviated as regex, is a sequence of characters that forms a search pattern that can be used to recognize the pattern in the strings (Friedl, 2006). The sequence of characters in regular expression can include literal characters and metacharacters. Literal characters can include letters (e.g., A, B, C), digits (e.g., 0, 1, 2), and other non-special characters (e.g., #, !, @). In regular expression, literal characters are used to specify exact sequences to match the text. While metacharacters are special characters in regex that have a specific meaning or functionality, beyond their literal interpretation. Common metacharacters can be shown in **Table 1**.

Table 1. Metacharacters in Regular Expression

Metacharacters	Roles
^ (caret)	Matches the start of the string
$ (dollar)	Matches the end of the string
? (question mark)	Matches zero or one occurrence of the preceding element
[...] (square brackets)	Matches any single character within the brackets
(…|…) (pipe)	Used to specify alternatives
\d	Matches any digit

2. Mathematical Modeling

This phase translates all the insights from the literature review into mathematical equations, enabling the precise manipulation and analysis of musical notes. In this study, mathematical modeling played a crucial role in developing algorithms that accurately reflect musical concepts. In this phase, regular expressions are used to match the enharmonic equivalents. The study shows the role of modular arithmetic to calculate intervals and handle octave shifts. Besides that, modular arithmetic operation can be used to shift notes by a specified number of semitones. The mathematical formulations were designed to provide a solid mathematical formalism for implementing the system across a variety of musical scenarios.

3. System Development

In this phase, the study implements the mathematical models in a programming language. Python was chosen for its versality and ease of use, but other programming languages (e.g. Java or C#) could also be suitable for this study. The implementation focused on creating modular, reusable code that adhered to the designed algorithms. Functions were developed for each aspect of the problem, including interval calculation and semitone shifting. By using basic functionality, this study also suggests algorithms for generating notes of a chord and identifying chords from given notes.

4. Testing and Validation

This phase aims to evaluate the system by ensuring accurate results. A variety of test cases were used, including validating musical notes and keys, calculating intervals between notes, shifting notes by semitones, and generating chords according to theoretical expectations and known standards. The results were analyzed to confirm that the implemented functions performed as intended, providing reliable and accurate outputs for all scenarios tested.

e-ISSN: 2686-2573
DOI: 10.21512/emacsjournal.v7i1.12562

III. RESULTS AND DISCUSSION

This section provides the results obtained from the application of foundational mathematical concepts to musical analysis and composition. By leveraging the chromatic scale, modular arithmetic, and regular expressions, the study suggests formalisms and algorithms that address various aspects of music theory and practice. This section provides detailed insights into the application of these mathematical concepts, that may enhance digital music technology and foster a deeper understanding of music theory.

3.1 Analyzing the Chromatic Scale through Modular Arithmetic

The circular nature of the notes in the chromatic scale can be explained using modular arithmetic. In modular arithmetic, two numbers are considered equivalent (or congruent) if they have the same remainder when divided by the modulus. This modular arithmetic concept helps to illustrate why the chromatic scale forms a circular pattern, where after B, the pattern repeats with C again. This paper suggests the creation of a list in chromatic order, including all enharmonic equivalents like C#/Db, D#/Eb, F#/Gb, G#/Ab, and A#/Bb as shown in **Table 2**. Even though the enharmonic equivalents are technically the same pitch, they have different names because of the key which they appear. Programming Languages don't naturally understand enharmonic equivalents in the same way that musicians do. When working with representing musical notes in a digital format, it can be convenient to give the same index for all enharmonic equivalents.

Table 2. Musical Key Index

Key	C	C#/Db	D	D#/Eb	E	F	F#/Gb	G	G#/Ab	A	A#/Bb	B
Index	0	1	2	3	4	5	6	7	8	9	10	11

When dealing with musical notes programmatically, accounting for enharmonic equivalents (e.g., C# and Db) is necessary. To achieve this, tools that recognize these patterns are needed, and regular expressions (regex) provide an effective solution because regex can recognize the pattern. By using regex, patterns can be defined that match both the natural notes and their enharmonic equivalents. This study also proposed a list of regular expressions that ensure correct recognition of musical notes by accounting for their enharmonic equivalents as shown in **Table 3**. By using these patterns, the program can correctly identify and index notes, ensuring that each enharmonic equivalent is treated as the same pitch.

Table 3. Musical Key Regular Expressions List

Index	Regular Expression (Regex)	Recognize Keys
0	^(C)$	C
1	^(C#|Db)$	C# and Db
2	^(D)$	D
3	^(D#|Eb)$	D# and Eb
4	^(E)$	E
5	^(F)$	F
6	^(F#|Gb)$	F# and Gb
7	^(G)$	G
8	^(G#|Ab)$	G# and Ab
9	^(A)$	A
10	^(A#|Bb)$	A# and Bb
11	^(B)$	B

After identifying and indexing the musical keys based on their various notations, the study employs modular arithmetic to analyze their circular nature. In this approach, each key is assigned a numeric index within a modular system of 12 as shown in **Equation (3)**. This representation captures the twelve semitones in an octave, where each key's position wraps around cyclically after reaching the 12th index. This modular framework facilitates the exploration of key relationships and transitions in music theory, illustrating how musical keys relate to one another in a continuous and cyclical manner.

$$index = index \ (mod \ 12) \qquad (3)$$

3.2 Musical Transposition using Modular Arithmetic Operations

Music transposition refers to the process of moving a piece of music from one key to another while maintaining the same intervals between notes (Hunt, 1970). This practice is common in music composition, arrangement, and performance to suit different instruments or vocal ranges. In computational music applications, musical transposition is formulated using modular arithmetic operations. In music theory and practice, transposition involves two distinct processes: transposing keys and transposing notes, each serving different purposes and contexts within musical composition and performance. While transposing key refers to shifting a piece of music from one key to another, transposing note refers to adjusting individual notes within a musical piece a specified number of semitones.

Key Transposition can be achieved by operating the index using modular arithmetic as shown in **Equation (4)**. The identification of note and the corresponding key index in **Table 3** are fundamental to applying this formula. The (old) key index is then manipulated using the equation to facilitate key and note transpositions. Using this equation, the (new) key index wraps around within the range of 0 to 11 after applying the operation with the specified number of semitones (both positive and negative), and it effectively handles the cyclic nature of musical elements.

$$key\_index_{new} = (key\_index_{old} + semitones) \ (mod \ 12) \qquad (4)$$

$$octave_{new} = octave_{old} + \left\lfloor \frac{key_{index_{old}} + semitones}{12} \right\rfloor \qquad (5)$$

On the other hand, Note Transposition not only transposes the key but also shifts the note, including its key and octave. An octave in music represents a systematic division of pitch ranges based on frequency relationships, with each octave denoted by a non-negative integer (natural number). To achieve this, the operation determines the octave by combining the original octave of the note with the integer result of dividing the sum of the key index and the number of semitones by 12 as shown in **Equation (5)**. Adding the key index and the number of semitones gives a temporary value that may exceed the range of a single octave (12 notes). The equation can calculate how many complete octaves the transposition spans by dividing the sum by 12. To obtain the integer part of the calculation, the equation uses the floor function to remove the decimal part.

3.3 Interval Calculation using Modular Arithmetic

In music theory, an interval refers to the pitch distance between two notes, quantifying their relationship and playing a fundamental role in defining melodies, harmonies, and chords. Recognizing intervals is an essential skill for developing holistic and sophisticated musicianship (Wong, Chen, & Lim, 2021). This study provides tools to calculate intervals by determining the number of semitones between two notes (note1 and note2). When both notes are in the same octave, the interval can be calculated by simply subtracting the key index

e-ISSN: 2686-2573
DOI: 10.21512/emacsjournal.v7i1.12562

of the first note from the key index of the second note as shown in **Equation (6)**.

In music theory, notes can span across various octaves. For accurate musical analysis, especially when notes move between octaves, really depends on understanding the complete interval between them. In this study, achieving accurate interval calculation across octaves in music theory requires combinations of the pitch and octave differences as shown in **Equation (7)**. Since there are 12 semitones in an octave, multiplying the octave difference by 12 converts the octave difference into semitones. But when the notes are in the same octave, this term becomes zero and does not contribute to the interval calculation.

3.4 Harmony Analysis using Modular Arithmetic

In music theory, harmony is the combination of different musical notes played or sung simultaneously to create a pleasing sound. It involves the vertical aspect of music, focusing on how chords are constructed and how they follow each other in a piece of music (Jimenez & Kuusi, 2018). Chords are the building blocks of harmony, creating the harmonic structure that supports the melody and overall musical expression. Understanding harmony allows musicians to create tension in music by using different chord combinations. This study provides the dictionary that defines various types of musical chords and their corresponding intervals in semitones from the root note as shown in **Table 4**. This dictionary can be used to generate chords, identify chords based on their notes, or analyze chord structures.

$$interval = key\_index_{note2} - key\_index_{note1} \tag{6}$$

$$\begin{aligned} interval = (&key\_index_{note2} - key\_index_{note1}) \\ &+ 12 \times (octave_{note2} - octave_{note1}) \end{aligned} \tag{7}$$

Table 4. Chords and their Corresponding Intervals

Chord Names	Corresponding Intervals (number of semitones)
major	0, 4, 7
minor	0, 3, 7
diminished	0, 3, 6
augmented	0, 4, 8
suspended2	0, 2, 7
suspended4	0, 5, 7
major7	0, 4, 7, 11
minor7	0, 3, 7, 10
dominant7	0, 4, 7, 10
dominant7flat5	0, 4, 6, 10
diminished7	0, 3, 6, 9
minor7flat5	0, 3, 6, 10
minorMajor7	0, 3, 7, 11
augmentedMajor7	0, 4, 8, 11
augmentedMinor7	0, 4, 8, 10
added9	0, 2, 4, 7
major9	0, 2, 4, 7, 11
minor9	0, 2, 3, 7, 10
dominant9	0, 2, 4, 7, 10
dominant7sharp9	0, 3, 4, 7, 10
major13	0, 2, 4, 7, 9, 11
minor11	0, 2, 3, 5, 7, 10
dominant11	0, 2, 4, 5, 7, 10
dominant13	0, 2, 4, 7, 9, 10

This study provides two main algorithms for working with musical chords. One of the algorithms generates all combination notes that form a specified chord name as shown in **Figure 3**. It begins by extracting the root note and chord type from the input. It then retrieves the corresponding intervals for the chord type from **Table 4**. Using these intervals, the algorithm calculates the notes that form the chord by applying the necessary transformations as specified in **Equation (4)** and **Equation (5)**. This method ensures that the generated chord accurately represents the intended harmonic structure, facilitating accurate musical analysis and composition.

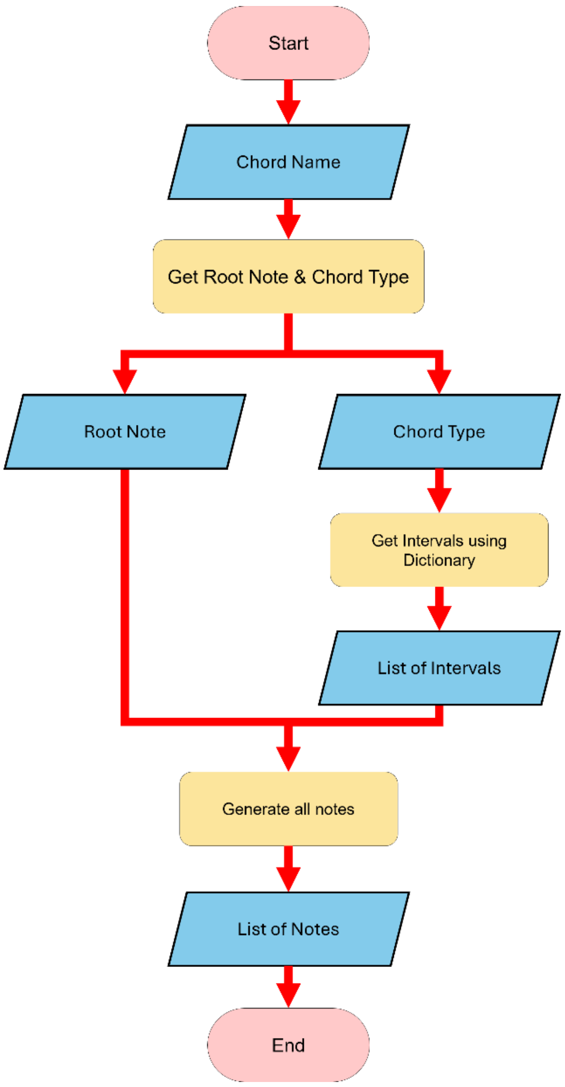
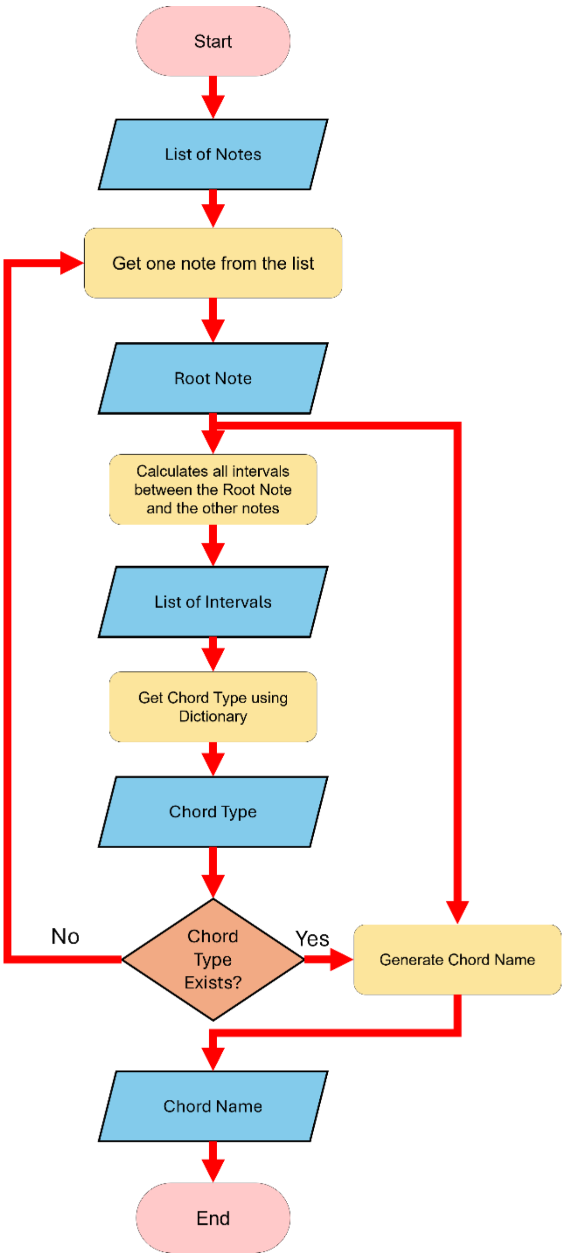


Figure 3. Generate Chord's Notes Algorithm



Start
List of Notes
Get one note from the list
Root Note
Calculates all intervals between the Root Note and the other notes
List of Intervals
Get Chord Type using Dictionary
Chord Type
No
Chord Type Exists?
Yes
Generate Chord Name
Chord Name
End

Figure 4. Chord Identification Algorithm

The second algorithm determines the name of a chord based on a given list of notes as shown in **Figure 4**. It starts by selecting one of the notes in the list as the root note one by one. The algorithm then calculates the intervals between the root note and the other notes using **Equation (7)**. These intervals are compared against a predefined set of intervals in **Table 4** for known chord types. If a match is found, the algorithm constructs the chord name by combining the root note with the identified chord type. This process allows for the accurate identification of chords from their constituent notes, aiding musicians in understanding and analyzing musical pieces.

IV. CONCLUSION

This study highlights the relationship between mathematics and music, demonstrating how the use of mathematical concepts can explain the composition and analysis of music. This research explains the circular nature of the chromatic scale using modular arithmetic and offers useful techniques for digitally expressing

e-ISSN: 2686-2573
DOI: 10.21512/emacsjournal.v7i1.12562

musical notes, including their enharmonic equivalents. The proposed algorithms for musical transposition and interval calculation provide reliable tools for both musicians and researchers. Harmony analysis through modular arithmetic demonstrates the applications of mathematical concepts in understanding chord structures and generating harmonies.

REFERENCES

Baumslag, B., & Chandler, B. (1968). *Theory And Problems Of Group Theory*. New York: McGraw-Hill Book Co.

Durfee, D., & Colton, J. (2015). The physics of musical scales: Theory and experiment. *American Journal of Physics*.

Friedl, J. E. (2006). *Mastering Regular Expressions.* O'Reilly.

Gorgoglione, M., Garavelli, A. C., Panniello, U., & Natalicchio, A. (2023). Information Retrieval Technologies and Big Data Analytics to Analyze Product Innovation in the Music Industry. *MDPI*.

Hess, J. (2020). Finding the “both/and”: Balancing informal and formal music learning. *International Journal of Music Education*.

Hunt, R. (1970). *Transposition for Music Students.* Oxford University Press.

Irving, R. S. (2004). *Integers, Polynomials. and Rings.* Springer.

Jimenez, I., & Kuusi, T. (2018). Connecting Chord Progressions with Specific Pieces of Music. *Psychology of Music*.

Mannone, M. (2021). A musical reading of a contemporary installation and back: mathematical investigations of patterns in Qwalala. *Journal of Mathematics and Music*.

Palmer, W. A., Manus, M., & Lethco, A. V. (1994). *The Complete Book of Scales, Chords, Arpeggios & Cadences.*

Pöpel, C., & Jürgens, E. (2022). On Overcoming the Gap between Industry and Academic Research in the Field of Music Technology. *Business Meets Technology*.

Wong, S. S., Chen, S., & Lim, S. W. (2021). Learning melodic musical intervals: To block or to interleave? *Psychology of Music*.