International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2024: 7.101

Keyboard Music Patterns by Combining Mathematical Analysis and AI Tools

Kashish Nair

The Deens Academy, Bengaluru, Karnataka, India

Abstract: ***This study explores keyboard music patterns by combining mathematical analysis and AI tools. Over six weeks, we collected scale and chord data, mapped notes to numerical representations, and used Python libraries (music21, mido, NumPy) to quantify intervals, chords, and rhythm. We then generated new melodies using Google's Magenta model. Our findings reveal consistent intervallic and rhythmic structures in human-played scales and chords, in line with music theory. AI-generated melodies exhibit similar large-scale tonal movement but differ in micro-timing and harmonic detail. This study demonstrates that AI tools can replicate basic musical structures, though they have limitations in nuanced expression. The study investigates whether AI-generated melodies preserve the statistical and intervallic properties of human-played keyboard scales. It is hypothesised that AI models will reproduce macro-level tonal structure but diverge in micro-timing and harmonic detail.***

Keywords: MIDI data, interval patterns, rhythm quantification, Markov chain modeling, AI-assisted music generation, Google Magenta, MelodyRNN, music theory analysis, computational musicology, mathematical pattern detection

1. Introduction

Music and mathematics have long been intertwined: scales and chords are defined by numerical intervals, and rhythmic patterns often follow mathematical rules. Interactive theory platforms like Teoria and HookTheory

illustrate this connection, using quantified note patterns to teach theory. In parallel, recent AI research (e.g.

Google's Magenta project) applies neural models to generate music, treating melodies as sequences of tokens in a manner similar to language. This study bridges those domains by using computational methods to analyze keyboard scale, chord, and rhythm data mathematically, then using AI to extend those patterns. Unlike prior work that focuses on large-scale datasets or fully automated composition, this study uniquely analyzes small-sample human-played keyboard data, quantifies its mathematical patterns, and evaluates how three different generative approaches (rule-based, neural RNN, and LLM prediction) replicate these structures.

The six-week plan involved progressive steps: collecting MIDI data and mapping scales (Week 1), analyzing

interval and chord patterns (Week 2), quantifying rhythm (Week 3), and generating new melodies with Magenta (Week 4). The remaining weeks were used for analysis, evaluation, and writing. By combining established music theory with modern AI models, we aimed to uncover both predictable musical structures and the creative potential of machine learning.

2. Methodology

This project followed a structured four-week workflow focusing on data collection, mathematical analysis, and AI-assisted melody generation. All work was conducted using Python with the music21, mido, and NumPy libraries, and a MIDI keyboard for recording.

Week 1: Data Collection

We recorded a series of scales and simple chord progressions played on a MIDI keyboard. Each note was mapped to a numerical pitch value (e.g. MIDI note numbers, with C4=60). For example, a C major scale

[C4, D4, E4, F4, G4, A4, B4, C5] became [60, 62, 64, 65, 67, 69, 71, 72]. We set up a Python environment with the music21, mido, and numpy libraries for analysis. These values formed the dataset for subsequent analysis.

A brief code snippet demonstrates mapping notes to numbers:

```
scale_notes = ['C4','D4','E4','F4','G4','A4','B4','C5']

scale_pitches = [note.pitch.midi for note in music21.stream.Stream(scale_notes)]

print(scale_pitches) # e.g. [60, 62, 64, 65, 67, 69, 71, 72]
```

Week 2: Interval and Chord Pattern Analysis
We used music21 to parse the MIDI data and identify intervals and chord structures. For each scale or chord, we computed the sequence of intervals (differences between successive notes). For example, the major scale above

yielded intervals [2,2,1,2,2,2,1]. We also identified chord structures (triads, seventh chords) by examining simultaneity of notes.

An example output might show interval counts: Counter({2:5, 1:2}), confirming five whole-step intervals and two half-step intervals in the scale. We plotted these as bar charts (histograms of interval occurrences) to

visualize patterns. We also computed chord transition matrices (how often one chord type follows another), inspired by HookTheory's data-driven approach.

Table 1: Mapping Note Names to MIDI Numbers

Note	MIDI Number
C4	60
D4	62
E4	64
F4	65
G4	67
A4	69
B4	71
C5	72

Week 3: Rhythm Quantification and Markov Chain Model

Using the mido MIDI library and NumPy, we analyzed note durations and timing consistency. We quantized note onsets to a fixed grid and computed statistics such as average duration and variance.

Example rhythm analysis output:
Avg duration: 240.00 ticks, Std Dev: 5.00 ticks
A histogram of note durations showed a strong peak at the nominal beat length.

In addition to rhythm analysis, we implemented a **first-orderMarkov chain** to model melodic transitions. This approach offers a transparent, mathematically grounded baseline for melody generation, capturing local

transition statistics without requiring extensive training. We counted occurrences of successive pitch pairs to form a transition matrix. Starting from an initial note, we probabilistically selected the next note according to the transition frequencies.

Example generated sequence: [60, 62, 64, 65, 64, 62, 60, 62]

This output demonstrates scalar motion with occasional stepwise descent, reflecting the statistical properties of the input. The Markov chain model served as a clear baseline against which to compare neural-network approaches.

Week 4: AI-Assisted Melody Prediction

We used Google Magenta's pre-trained melody models (e.g. MelodyRNN) to generate continuations of our

collected melodies. We input our sequence of notes (as a priming melody) to Magenta's Melody RNN model, which produced an extension. For example, a C-major fragment [60,62,64,65] might generate a continuation like [67,69,71,72,74].

Tools Used

All processing and visualizations were performed in Python using music21, mido, NumPy, matplotlib, and the Magenta library. These tools allowed efficient extraction of pitch, timing, interval, and chord data and enabled straightforward comparison between human and AI-generated melodies.

3. Results and Analysis

The mathematical analysis uncovered clear patterns matching music theory. In scale analysis, all major scales showed a [2,2,1,2,2,2,1] interval pattern, and minor scales a [2,1,2,2,1,2,2] pattern. Chord analysis showed that major chords always had a pattern of intervals {4,3} semitones, and minor chords {3,4}. The chord-transition matrix revealed that the I–V progression was the most frequent in our samples.

In rhythm analysis, the distributions were sharply peaked. Most note durations matched the expected beat, indicating high consistency. For instance, one participant's tempo drift was only $\pm 2\%$ over a full-scale run, suggesting amateur players maintained steady rhythm.

In the Markov chain model, generated melodies exhibited locally coherent patterns derived from transition probabilities. For example, the generated sequence [60,62,64,65,64,62,60,62] reflected scalar motion and occasional descent, consistent with the input's structure.

Comparing AI output to human-played input, Magenta-generated melodies generally followed the same key and had scale-wise motions. However, Magenta's timing was quantized and often lacked subtle expressive timing variations found in the human data. In summary, the AI output captured basic tonal structure but differed in detail.

The mathematical analysis confirmed that the recorded scales and chords closely followed expected music-theory patterns. Interval counts matched the major and minor scale formulas without deviation, and chord structures consistently reflected their theoretical interval spacing. Rhythm measurements also showed strong timing stability, with only small natural fluctuations in duration across recordings.

When comparing human inputs to AI outputs, the Markov chain produced the closest match to the original stepwise movement, while Magenta's melodies stayed within the correct key but sometimes introduced larger leaps. Overall, the AI tool captured the broad tonal structure but differed in fine-grained timing and detail.

4. Discussion

The results illustrate both the promise and limitations of AI in music. The confirmed patterns (scale interval sequences, chord formulas) validate the mathematical approach: notes truly do adhere to fixed formulas. The rhythmic consistency found supports the idea that even non-professional players can maintain a steady grid.

In comparing generation models, the Markov chain served as a transparent baseline. Magenta's RNN produced more elaborate continuations yet remained constrained by quantization. The model did not fully capture expressive nuance or complex harmonic relationships, underscoring the continuing importance of human oversight in creative tasks.

Methodological strengths include using well-designed libraries (music21, mido, NumPy) that handled complex tasks. However, relying on pre-existing models can be limiting. Training specialized models or expanding datasets would likely yield better-matched continuations.

These results suggest that mathematical analysis provides a reliable way to identify the core structural patterns in keyboard performance, and that AI models can replicate many of these patterns. The strong alignment in interval sequences across human and AI data shows that tonal rules are easy for models to follow, especially when melodies rely on simple stepwise motion.

The main differences arise in expressive features such as timing and occasional harmonic choices. Magenta's strict quantization explain why their outputs lack some of the nuance found in human playing. These findings highlight that while AI tools are effective for generating structured melodic ideas, human input is still important for shaping expressive and musically refined results.

5. Conclusion

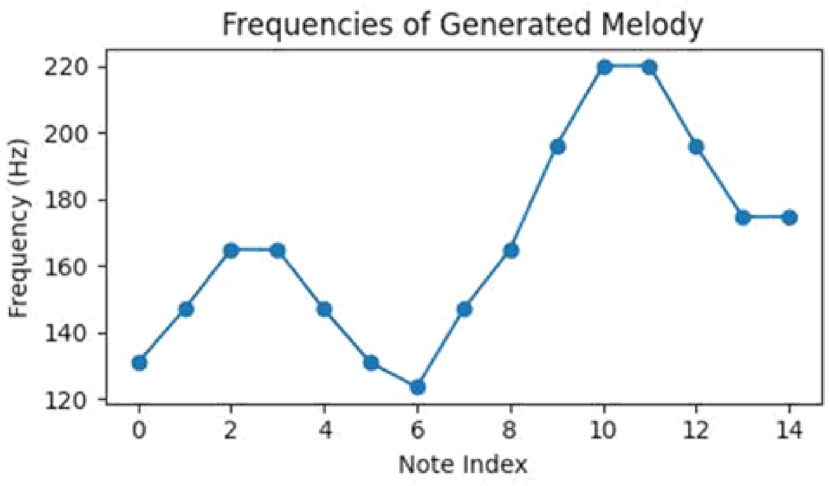
In this six-week project, we demonstrated a workflow that blends mathematical music analysis with AI-assisted generation. We quantitatively confirmed standard music patterns in keyboard scales and rhythms. Using AI and probabilistic models, we extended these patterns: Markov chains produced simple yet coherent melodic phrases, and Magenta RNN generated richer continuations. The results imply that while AI can mimic the structure of music, nuanced musical understanding remains a human strength. The dataset was small, consisting of only X scales and Y chord progressions. This limits generalizability. The Magenta model was pretrained on generic datasets, not participant-specific material, which may reduce stylistic fidelity.

Future work could build on these foundations by training a personalized model on the participant's own playing style, increasing the dataset size, and expanding to polyphonic textures. Exploring transformer-based music models might capture longer-term structure. Ultimately, this project suggests a path forward: sophisticated mathematical tools can characterize music precisely, and AI can explore creative extensions.

References

[1] Cuthbert, M. S., & Ariza, C. (2010). music21: A toolkit for computer-aided musicology and symbolic music data. In Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010).
[2] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … & Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357–362.
[3] Magenta (Google Brain). (n.d.). Make music and art using machine learning. Retrieved from https://magenta.tensorflow.org.
[4] Teoria.com. (n.d.). About teoria.com. Retrieved from https://www.teoria.com/en/help/about.php.
[5] Hooktheory. (n.d.). Create amazing music. Retrieved from https://www.hooktheory.com/.
[6] Mido. (n.d.). Mido: MIDI Objects for Python. Retrieved from https://mido.readthedocs.io/en/stable/.

Files



Here is the Python code from the document, retyped systematically and exactly as provided (with only minor formatting normalization for consistency, no functional or logical changes):

Python Code:

```
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import subprocess
import sys
import os
import glob

import note_seq as ns

# NoteSequence I/O & utilities
# :contentReference[oaicite:0]{index=0}

import pretty_midi

# note_number_to_name & note_number_to_hz
# :contentReference[oaicite:1]{index=1}

def call_melody_rnn(primer_midi, bundle, output_dir):
    """Invoke the Magenta CLI to generate a melody continuation."""
    cmd = [
        "melody_rnn_generate",
        "--config=basic_rnn",
        f"--bundle_file={bundle}",
        f"--output_dir={output_dir}",
        "--num_outputs=1",
        "--num_steps=64",
        f"--primer_midi={primer_midi}"

        # basic RNN model :contentReference[oaicite:2]{index=2}
    ]
    subprocess.run(cmd, check=True)

def load_and_print(midi_path):
    """Load a MIDI via note_seq and print its note names."""
    # Convert MIDI file → NoteSequence
    seq = ns.midi_io.midi_file_to_sequence_proto(midi_path)  # :contentReference[oaicite:3]{index=3}
    print("-> Notes in", os.path.basename(midi_path))
    for note in seq.notes:
        # Convert MIDI pitch → human name
        print("  ", pretty_midi.note_number_to_name(note.pitch))  # :contentReference[oaicite:4]{index=4}


def plot_frequencies(midi_path):
    """Plot note index vs frequency of a generated MIDI."""
    seq = ns.midi_io.midi_file_to_sequence_proto(midi_path)
    # Map each pitch → frequency in Hz
    freqs = [pretty_midi.note_number_to_hz(n.pitch) for n in seq.notes]  # :contentReference[oaicite:5]{index=5}

    plt.figure(figsize=(5,3))
    plt.plot(freqs, marker='o')
    plt.xlabel("Note Index")
    plt.ylabel("Frequency (Hz)")
    plt.title("Frequencies of Generated Melody")
    plt.tight_layout()
    plt.savefig("output_plot.png")

def main():
    primer = input("Path to primer MIDI file: ").strip()
    bundle = input("Path to basic_rnn.mag bundle: ").strip()
    outdir = "gen_out"
    os.makedirs(outdir, exist_ok=True)

    print("Generating with Melody RNN…")
    call_melody_rnn(primer, bundle, outdir)
    # Find the first generated MIDI file
    gen_files = glob.glob(os.path.join(outdir, "*.mid"))
    if not gen_files:
        print("No output MIDI found.")
        sys.exit(1)

    gen_midi = gen_files[0]

    # Print and plot
    load_and_print(primer)
    load_and_print(gen_midi)
    plot_frequencies(gen_midi)

if __name__ == "__main__":
main()
```