APPLIED INTERPOLATION FOR MUSICAL MELODIES COMPLEMENTATION AND FILLING

David Calle González

Department of Informatics and Systems
Universidad EAFIT
Medellín, Colombia
dcalle@eafit.edu.co

Juan Sebastián Díaz Osorio

Department of Informatics and Systems
Universidad EAFIT
Medellín, Colombia
jsdiaz@eafit.edu.co

Simón Marín Giraldo

Department of Informatics and Systems
Universidad EAFIT
Medellín Colombia
smaring1@eafit.edu.co

June 2, 2021

ABSTRACT

Mathematics and music traditionally have been seen as very opposite areas of human development, one being rational and deterministic and the other more emotional and creative. A question that frequently arises is whether these two worlds can be joined in a certain manner, and that is what this work is about. Through the use of interpolation techniques we seek to generate functions for representing songs and thus determine new ways of musical improvisation which allow us to experience music not merely from the symmetry brought by the sounds themselves, but also from the rigour and accuracy of the mathematical and numerical methods.

1 Introduction

Numerical methods are an indispensable modern technique for data analysis. They have determined new paths with which we can model processes, interpret information and develop more efficient programs [1] that support different fields of human progress. Nevertheless, one area is always disregarded when it comes to the application of numbers: art. The purpose of this research is to experiment with music and its sounds, applying the concepts of the most fundamental numerical methods in order an attempt to identify behavioral patterns.

Our goal will be to analyze whether it is possible with these approaches to build new melodies from the functions that we describe in this analysis to continue generating the emotions that any song can provoke.

2 Materials and Methods

2.1 Data Acquisition and pre-processing

Our data is derived from sheet music, a written document that describes the melodies and rhythms of a song. In them, there is a very well defined notation for the musicians regarding the times and moments in which a particular sound must be played in order to interpret a song.

Himno Nacional de Colombia

Orestes Sindice Música
Rafael Nuñez Letra
directoriopax.com

Himnos del Mundo

Por Gerson
Editada por diegosax.es
tubepartitura.com

Clarinete

Cl

¡Oh Glo - ria ....

Figure 1: Colombia's National Anthem

We will not dive into understanding the reading of sheet music, but we will learn how to create functions from them. We can thus define points in the plane with a simple mapping of values.

The fundamental notes in music are: **C, D, E, F, G, A, B**, along with other complementary ones consisting of **C#, D#, F#, G#, A#** which form the chromatic scale of music. [2]

At the physical level, musical notes are sounds at different wave frequencies [3] (with other characteristics such as harmonics that distinguish them among instruments). The **Hz (Hertz)**, as a measure of frequency, therefore allows us to label the notes by values which can describe a function:

Musical Notes	Hz
C	261.63
C#	277.18
D	293.66
D#	311.13
E	329.63
F	349.23
F#	369.99
G	392
G#	415.3
A	440
A#	466.16
B	493.88

Table 1: Musical notes to Hz conversion

2.2 Preparing the dataset

We have selected three songs for this project. One of them and the one to be used for the examples in this article is Twinkle Twinkle Little Star.

Figure 2: Twinkle Twinkle Little Star music sheet

Taking the xy plane, let the x -axis be the time and the y -axis the Hz to be played at that moment. The following graph of points of the song can be obtained:

Figure 3: Twinkle Twinkle Star points

In music, besides sound, silences exist. They are the basis of rhythm and are important because they represent gaps between melodies that shouldn't be taken in consideration.

Therefore, we decided to formulate the problem as a piecewise function which only defines moments with sound. Hence, the following function results:

Figure 4: Twinkle Twinkle Little Star piecewise function

3 Algorithm

3.1 Input definition

We require "the song function" to be in Geogebra's piecewise function format for our algorithm. We do so because it provides us with a graphical view prior to its use in our algorithm.

Listing 1: Input example

```
if(0 <= x < 1, 261.63, 1 <= x < 2, 261.63,
2 <= x < 3, 392, 3 <= x < 4, 392,
4 <= x < 5, 440, 5 <= x < 6, 440,
6 <= x < 8, 392, 8 <= x < 9, 349.23,
9 <= x < 10, 349.23, 10 <= x < 11, 329.63,
11 <= x < 12, 329.63, 12 <= x < 13, 293.66,
13 <= x < 14, 293.66, 14 <= x < 16, 261.63)
```

3.2 Conversion process

In order to perform the conversion process, we set up a function which receives as parameter a string s , which corresponds to Geogebra's piecewise function. Initially, for the implementation, we create a dynamic vector *results* for storing the results. Subsequently we remove the first 3 characters of s , to get rid of the `if (`, which is useless. We then create a vector *lines*, that will hold each line of s . Afterwards, we proceed to traverse each element of the vector *lines* using a FOR loop. To each element (line) we eliminate the last character, which would be an extra comma. Let's continue with a `split` of the line with the comma separator `,`, which will be stored as a vector in the variable *div*.

Next, we grab the first position of *div* and make a `split` with the separator `<` and store it in the variable *temp*. We then continue with the conversion to float of each item in the tuple consisting of `(temp[0], temp[-1], div[-1])`. Lastly, we add this tuple to the results vector *results*. This for every run of the FOR. Finally, we return the list of tuples. The code used here is as follows:

Listing 2: Conversion algorithm

```
def ranges_to_tuples(s: str):
    results = []
    s = s[3:]
    lines = s.split("\n")
    for line in lines:
        line = line[:-1]
```

```

div = line.split(',')
temp = div[0].split('<')
elements = tuple(map(float, (temp[0], temp[-1], div[-1])))
results.append(elements)
return results

```

Listing 3: Tuples for algorithm example

```

notes = [(0.0, 1.0, 261.63), (1.0, 2.0, 261.63)
(2.0, 3.0, 392.0), (3.0, 4.0, 392.0)
(4.0, 5.0, 440.0), (5.0, 6.0, 440.0)
(6.0, 8.0, 392.0), (8.0, 9.0, 349.23)
(9.0, 10.0, 349.23), (10.0, 11.0, 329.63)
(11.0, 12.0, 329.63), (12.0, 13.0, 293.66)
(13.0, 14.0, 293.66), (14.0, 16.0, 261.63)]

```

3.3 Interpolation method

The interpolation methods employed in our investigation were: **Newton's method with divided differences** and **Lagrange's method**. We used two different methods in order to compare results as they could produce different outputs. For the coordinates we use the initial moment of the song at which each note sounds and the corresponding note value. Below, we have the plots showing the predictions with each of the implemented methods.

3.4 Initial model

To perform the initial analysis we employed both interpolation techniques, considering all the notes. The blue dots represent the notes of the song in the respective initial moment of each one, and the orange line is representing the prediction of the resulting polynomial.

These were the results of the prediction using both methods on the song **Twinkle Twinkle Little Star**. As can be observed, there is no major change between the two methods and the model seems to fit relatively well for points that are not at the corners:

Figure 5: Twinkle Twinkle Little Star plots

In the case of the song **Old McDonald**, the prediction using both methods gave the same result, in which there are abnormally high values at both ends of the song that are so huge that the remaining results cannot even be clearly appreciated, due to the fact that in comparison with **Twinkle Twinkle Little Star**, this song has more abrupt changes:

Figure 6: Prediction over Old McDonald

Lastly, the major error is perceived in the song **Final Countdown**, in which the prediction with both methods showed the same result too. Anomalously high values are evidenced at both ends of the song as well as in other zones.

Figure 7: Prediction over The Final Countdown

Based on this initial analysis we then confirmed something. It verifies the fact that for such abruptly changing sets of points, the interpolation may have problems attempting to fit all points, and the more abruptly the changes are, the more severe the problem will be. Therefore, an alternative way to analyze the melodies of the songs is required. Thus, we proposed to go ahead and apply piecewise interpolation.

4 Final model

In order to prevent the errors we obtained with the initial approach, it was decided to this time interpolate by using a maximum of 3 points on each side of the note we wanted to predict, rather than making use of the entire song, which led to results that appear to be closer to what could be a new melody.

The results for the song **Twinkle Twinkle Little Star** were as follows, the errors for both methods were not as pronounced as those seen in the initial approach, and something was noticed, a softer curve is observed for the divided differences method:

Figure 8: Twinkle Twinkle Little Star plots

Regarding the **Old McDonald** song, further improvements are noticed. Errors of excessive values at the ends of the melody are no longer noticeable and smoother curves are observed, furthermore, it is also evident that the curves tend to be less pronounced with the divided differences method.

Figure 9: Old McDonald plots

Finally, the most problematic song, **Final Countdown**, yields much more satisfactory results than those obtained previously, with softer curves and a better fit, and, as in the previous examples, the performance is noticeably more polished and improved with the divided differences method.

Figure 10: Final Countdown plots

In this final stage, we were able to conclude that the best option is to interpolate using around 3 points maximum on each side of the value to be evaluated in order to avoid errors with excessive quantities, and in addition, it was discovered that the method of divided differences enables us to achieve softer and shallower curves. Thus, only one thing left to do is to test the results in real life to check how they sound.

5 Testing

For testing our algorithm the **musicalbeeps** library and the code from **this web page** were used as a tool for converting frequencies into notes we could hear. Trying to use random beats and adding extra beats to the songs we already had, we ended up with not very successful results. By adding extra beats to the preset songs, mixed results are obtained, some times the addition sounds fine and at other times it comes out terribly wrong, besides, in the case when random beats are chosen, the song becomes meaningless and it is rare for any part to sound actually good.

As well, there were three main things noticed. Firstly, the Lagrangian method is more likely to predict lower notes than the divided difference method, secondly, in simple melodies like Twinkle Twinkle Twinkle Little Star, the probability to sound better is higher, whereas in melodies like The Final Countdown which have more abrupt changes, the predicted melodies seem to make less sense and sound worse. Furthermore, the closer the predicted point is to an original point, the more likely it is to sound fine with the melody, i.e., if a note is predicted half a beat ahead it is more likely to sound better than a note predicted 1 beat ahead.

6 Conclusions

As a result of this process we were able to listen to functionally defined music. The overall experience can hardly be described as amazing, since a musician's ability to improvise reaches the point of, at certain times, discomforting people's auditory sense to be fully focused on what he or she is doing [4].

Despite the occasional unfavorable and unusual results, we were able to observe how the generated peaks provided a certain rhythmicity, consistency and experimentation, belonging even to the melody we were working on. Thus, this is what made this experiment particularly interesting and sparked some curiosity in applying other numerical methods such as extrapolation, the Runge-Kutta method or finding a relationship between all this and linear algebra and the solution of linear equations.

References

- [1] F. J. Correa, *Métodos numéricos*. Medellín, Colombia: Fondo Editorial Universidad EAFIT, 2010.
- [2] "How to read sheet music: Step-by-step instructions," Feb 2021. [Online]. Available: <https://www.musicnotes.com/now/tips/how-to-read-sheet-music/>

- [3] C. Goss. [Online]. Available: http://www.flutopedia.com/octave_notation.htm
- [4] D. Beckstead, "Improvisation: Thinking and playing music," *Music Educators Journal*, vol. 99, no. 3, pp. 69–74, 3 2013. [Online]. Available: <https://bit.ly/2SSEVMW>