Sonification of Quantum Algorithms with Spline Models of Timbre, Melody, and Rhythm

Matt Klassen[1]*

[1]DigiPen Institute of Technology
Redmond, Washington, USA

mklassen@digipen.edu

Abstract. *In this paper we describe methods for mapping the information contained in a spline model of one cycle of an audio signal to the quantum information setting. We use this mapping to describe methods of sonification of quantum algorithms which use spline models to describe and generate timbre, melody, and rhythm. We show how a musical state vector given by n B-spline coefficients, which represent one cycle of a waveform, can produce these three musical elements simultaneously. We present two modes of correspondence between quantum and musical states. The first mode is derived from proposed signal processing of quantum audio, such as QPAM (Quantum Probability Amplitude Modulation), and the second mode works with superposition states achievable through quantum algorithms. Musical states are initialized using approximate timbres derived from instrument samples or synthesized timbres. Melodic contours are derived from spline models of these timbres, using continuous pitch and duration spaces. Musical examples are discussed, and larger compositional works are currently in production.*

1 Overview

Musical elements such as timbre, melody, and rhythm can describe the state of a piece of music at various points in time. A listener can experience these elements and a music theorist can describe them and their interrelationships. Further, an audio engineer or scientist can analyze how these elements occur precisely in time through sampling and recording. All of these activities are classical, occuring in the domain of human experience and classical physics. Only recently are researchers now imagining these things in the domain of quantum computation.

In this paper we investigate the question: *How might these musical elements evolve in time if they were reflecting the types of change of state which occur in the quantum domain?*

In order to explore this question, we create some embeddings of timbre, melody, and rhythm in the space of quantum computation. A key starting point is the notion of a cycle, or approximately repeating segment of a waveform. In the digital audio domain this can be described by a piecewise linear graph over some time interval $[a, b]$, with positive slope at the two endpoints, and length $L = b - a$. Further, the waveform should have approximate fundamental frequency $f_0 = 1/L$ in some region containing the interval $[a, b]$. We use models of such cycles with cubic B-splines allowing for uniform description by sets of B-spline coefficients. Timbre of one tone can be nicely controlled by a few such cycles, with methods to interpolate between them. See [1] for more details on these models.

Since the cycles are represented by vectors of real numbers (of B-spline coefficients) we can devise methods to embed these vectors into the quantum setting. Methods of doing this already exist for audio samples and so we borrow some of these techniques (such as QPAM in [2]) to arrive at embeddings of this representation of timbre. These methods form the first mode of interaction between musical states (such as timbre and melody) and quantum states (such as multi-qubit states). In this mode we embed vectors as probabilities of basis states by doing quantum measurements. This allows for the possibility of implementation, and for instance of running actual quantum circuits which can be used to produce statistical output. From this output we could extract versions of the vectors that are used for modeling the musical state.

The second mode works with quantum states in superposition and could be implemented through simulation of quantum algorithms on classical computers. In this mode we are more interested in the quantum states from an abstract perspective, but there is still the goal of understaning how such musical elements can evolve in the context of quantum algorithms.

In the following sections we summarize the spline models of cycles, then describe some methods for encoding these cycles as qubit registers. These are followed by two sections on modeling of timbre and melody using splines. Finally, we give a short example of sonification using a simple quantum circuit.

2 Spline models of cycles

Previous use of splines in audio synthesis and f_0 model envelopes, can be found in [3] and [4] respectively. We use the techniques described in [1] to model audio segments, having some discernable but approximate fundamental frequency f_0, using splines. One can choose a set of representative cycles, or segments which are approximately repeating, and use interpolation to fill in the intermediate cycles. This works well for instrument samples, where it is possible to construct accurate models of audio by using data from a small number of cycles. The local timbre of a waveform is preserved if the number of points in one cycle matched by the spline (called interpolation points) is approximately one third of the number of samples in the cycle.

*Supported by DigiPen.

Such spline models of audio are:
- *low resolution* models, can be used to reconstruct an audio signal from a smaller set of data
- *time domain* models, differing from typical compression models which work with frequency bands.
- *locally computable* models, using de Boor algorithm (see de Boor's book [5])

An important point is that cycles are modeled over time intervals $[a, b]$ with a and b real numbers, not restricted to samples. So exact values of fundamental frequency f_0 correspond to exact cycle lengths $1/f_0$. Since we do compute with sampled waveforms, it will be assumed that sampling can be treated as independent of f_0 and cycle length. The spline function used to model a cycle can be specified to match certain interpolation points coming from a real waveform. In this case we treat the sampled waveform as a continuous function by using linear interpolation between sample values, so that selecting say n evenly spaced points along a cycle can also be independent of the placement of samples. Once a spline model of a cycle is determined, as a set of B-spline coefficients, it can then be used on a cycle of any length, appropriately resampled.

Another important point is that even though cycles are initially thought of as representing waveforms with some constant f_0, this changes when we do cycle interpolation, or when we vary pitch as in glissandos. Briefly, the method of cycle interpolation is to take a sequence of *key cycles* which capture the waveform at some important points in time (much like key frames for animation). Then one can interpolate between vectors of B-spline coefficients, allowing the computation of intermediate cycle representations (much like computing animation frames). Whether we are modeling recorded sounds or synthesizing new sounds, or a mixture of these two, the cycle modeling and cycle interpolation methods thrive in the context of waveforms with varying pitch and timbre.

We define the basic model to be the sequence of zero crossings z_j, $j = 0, \ldots, p$ and the sequence of spline functions $f_j(t)$ on the interval $[z_j, z_{j+1}]$. We also refer to each of these splines and their associated data as one "cycle" C_j, so that the basic model is the sequence of cycles C_j for $j = 0, \ldots, p-1$. Further, let the B-spline coefficients for cycle C_j be c_i^j, $i = 0, \ldots, n-1$.

The B-spline functions can be defined with divided differences, and knot sequence (non-decreasing sequence of real numbers) $\{t_0, \ldots, t_N\}$ as:

$$\mathcal{B}_i^d(t) = (-1)^{d+1}(t_{i+d+1} - t_i)[t_i, t_{i+1}, \ldots, t_{i+d+1}](t-x)_+^d$$

or using the (de Boor-Cox) recursion formula as

$$\mathcal{B}_i^d(t) = \frac{t - t_i}{t_{i+d} - t_i}\mathcal{B}_i^{d-1}(t) + \frac{t_{i+d+1} - t}{t_{i+d+1} - t_{i+1}}\mathcal{B}_{i+1}^{d-1}(t)$$

We compute values of the B-spline functions, or of the interpolating spline f, with nested linear interpolation, known as the de Boor algorithm:

- (init:) Set $c_i^0 = c_i$ for $i = 0, \ldots, N-d-1$.
- For $t \in [t_d, t_{N-d})$ set J to be the index so that $t \in [t_J, t_{J+1})$.
- (Stage p:) For $p = 1, \ldots, d$, for $i = J-d+p, \ldots, J$: set

$$c_i^p = \frac{t - t_i}{t_{i+d-(p-1)} - t_i}c_i^{p-1} + \frac{t_{i+d-(p-1)} - t}{t_{i+d-(p-1)} - t_i}c_{i-1}^{p-1}$$

- $f(t) = c_J^d$.

Note: to solve for the coefficients c_i, $i = 0, \ldots, n-1$, we set up the interpolation problem as a linear system by requiring that f agrees with the audio output data at each of the n input values. Since the input values are evenly distributed, we are guaranteed a unique solution, according to the Schoenberg-Whitney Theorem on B-spline interpolation (see [5]).

Also note: we may normalize a set of B-spline coefficients for various purposes, in order to focus on the shape of a cycle rather than its peak amplitude, for example. When constructing tones which can represent instrument samples of short duration (say several seconds) it is often desireable to have an envelope which is independent of timbre. This can be easily achieved by choosing a set of key cycles and a corresponding set of amplitude multipliers. Once the key cycles are scaled in this way, an envelope is naturally computed by the cycle interpolation method.

In Quantum Computing it is necessary to use the Euclidean Normalization, which specifies that the sum of the squares of the complex norms of coefficients for the computational basis states is 1. In the case of B-spline coefficients, we adopt this normalization for simulation in the example of the last section.

3 From Cycles to Qubits

Our goal is to represent cycles as vectors of B-spline coefficients with qubit registers. Once these are defined, one can then proceed to compute samples as arrays of fixed point integer representations or as floats. The method of cycle interpolation allows for the generation of arrays on the order of several seconds representing musical notes. Additionally, the same cycles can be used to generate melodic contours and pitch sequences.

As a first method of mapping models of cycles to qubits, we use Quantum Probability Amplitude Modulation (QPAM) which is described in [2]. This method is described for audio signal values, assumed to be in the interval $[-1, 1]$. A multi-qubit register is used to encode time values as its possible states, resembling an array but in quantum superposition. Since we can also normalize B-spline coefficients to satisfy this restriction, we can follow the same procedure. However, in this case we do not explicitly specify time values for samples a_i. Instead, we can think of i as specifying an index into the array of B-spline coefficients. These c_i are in sequence and indeed this sequence does follow the flow of time. The values c_i can in this way be thought of as a "down-sampled" version of

the audio which forms the signal graph over the interval defining one cycle. A direct connection to sample values can then be made from the representation through the de Boor algorithm which can be used to compute samples via nested linear interpolation of the B-spline coefficients.

The encoding of B-spline coefficients c_i then follows these basic transoformations: first map c_i to $c_i' = (c_i + 1)/2$ (so that these values are each in the interval $[0, 1]$), then normalize the sum to 1, so $c_i'' = c_i'/c$ where c is the sum of all the c_i. Then finally we put $\alpha_i = \sqrt{c_i''}$. In a multiple qubit representation we have the assignment of the probability $|\alpha_i|^2$ to the computational basis element $|i\rangle$. This procedure then enables the encoding of cycle models given as vectors of n B-spline coefficients $\mathbf{c}$ into some quantum register of size n.

QPAM retrieval can also be applied to extract such a set of B-spline coefficients from the quantum domain using measurement statistics. As pointed out in [2] this process may require the preparation of many states in order to form a histogram of measurements. This mapping allows for the translation of data which specifies cycle models back and forth between the classical and quantum contexts. Once probability amplitudes are determined experimentally through measurements as $\tilde{p_i}$ we can reverse the above transforms to get

$$c_i = 2g\tilde{p_i} - 1,$$

where g is a normalization coefficient. As indicated by the authors in [2], g can be chosen somewhat arbitrarily to determine the amplitudes of the c_i, which are used to generate an audio signal. In their paper, $c_i = a_i$ which are the raw audio sample values. In the present work, the c_i are B-spline coefficients which may be used to generate an audio signal through a process of cycle interpolation. The size of these coefficents then determines the size of the audio samples generated. The choice of g is thus one step removed, but is still important in determining the amplitude range of the audio signal. An important fact here is that the B-spline basis functions are locally a partition of unity, so $c_i \in [-1, 1]$ implies that the signal values generated will also be in $[-1, 1]$. Further, one can use envelopes to control the shape of the audio signal output in the case that cycles are used to mimic instrument samples.

Another method, also described in [2], is Single Qubit Probability Amplitude Modulation (SQPAM) which encodes the values c_i into $\text{ceiling}(\log \text{n}) + 1$ qubits instead of using raw probability amplitudes. There is also a dedicated qubit $|\gamma_i\rangle$ used for the encoding of values as coefficients of basis kets, by mapping

$$c_i \longrightarrow |\gamma_i\rangle = \cos\theta_i\,|0\rangle + \sin\theta_i\,|1\rangle\,,$$

where

$$\theta_i = \sin^{-1}\sqrt{\frac{c_i + 1}{2}}.$$

The preparation of this encoding uses only one qubit per coefficient c_i as described above, but n qubits for the time. Just as with QPAM, we consider those as indices but not as times. Also as with QPAM, we can do the reconstruction of the set of B-spline coefficients using

$$c_i = \frac{2p_{\gamma_i}(|1\rangle)}{p_{\gamma_i}(|0\rangle) + p_{\gamma_i}(|1\rangle)} - 1,$$

with

$$p_{\gamma_i}(|0\rangle) = \cos^2\theta_i,\;\; p_{\gamma_i}(|1\rangle) = \sin^2\theta_i.$$

These methods are called "coefficient-based representations". It is also possible to use "state-based representations" which store the values of B-spline coefficients more directly as multi-qubit states in superposition.

In all of these cases we extract information from the quantum setting which is used to define vectors of B-spline coefficients, which can be used to model cycles of some waveform. Typically such cycles can be designated as key cycles for the purpose of generation of full audio waveforms. For example, one might generate a waveform of length 1 second with 5 cycles each on the order of several milliseconds. This process of cycle interpolation yields a waveform with smooth evolution of cycle shape, mimicking real waveforms of instrument samples. Even though the cycle lengths can be taken to be uniform, and we speak of $f_0 = 1/L$, the cycle shapes are not periodic.

The above describes the reconstruction of a full set of cycles as vectors of B-spline coefficients. In order to convert these into actual samples we use the de Boor algorithm to evaulate B-spline functions with nested linear interpolation.

4 Timbre modeling

With these methods of encoding B-spline coefficients, we can set up a layered approach to musical states which determine timbre, pitch, and duration. These musical elements can be determined at the sample level of audio, or at a higher level of musical time. For example, timbre can be determined by the evolution of a waveform from one cycle to another. To be more precise, we can generate a one second audio segment with fundamental frequency $f_0 = 220$ Hz, by setting initial and final cycles of length $1/220$ sec each with two sets of B-spline coefficients. These may be extracted from recorded audio samples, for instance, in order to mimic an instrumental timbre. If the sample rate is 44100 samples per second, then each cycle will have about 200 samples. A good B-spline model for each cycle can be given with 32 B-spline coefficients. The remaining samples can be computed by cycle interpolation, for example linear interpolation between corresponding coefficients of the two cycles, followed by waveform computation using the de Boor algorithm.

5 Melodic contour modeling

The method of melodic contour modeling using B-spline curves is to start with a spline function plot over the interval $[0, 1]$ with function values in the interval $[-1, 1]$. (This also can be thought of as a standard normalized model for a cycle.) Such a spline can be plotted with any set of

basis coefficients for some B-spline basis on the interval $[0, 1]$. The number of B-spline coefficients n is somewhat independent of this construction and generation of melodic contours. We interpret the x-axis of such a plot as time and the y-axis as pitch on a logarithmic scale, so that one octave is spanned by the interval $0 \leq y \leq 1$. We equate pitch with fundamental frequency, and also assume that some chosen pitch represents $y = 0$. There are many ways to proceed from this point to generating a sequence of pitches, and we mention just one of these here. A simple method is to compute the stationary points on the spline (where the first derivative is zero) and to use these points to break up the x-axis into subintervals. Suppose that the stationary points occur at points (x_i, y_i), for $i = 1, \ldots, m$. One can then generate the melody with initial pitch $y = 0$ lasting for time interval $[0, x_1]$ followed by pitch with f_0 given by 2^{y_1} for time interval $[x_1, x_2]$, and so on, finishing with pitch given by 2^{y_m} on the last interval $[x_m, 1]$. All time interval lengths can be scaled according to compositional preference. In Figure 1 there is an example of such a spline, with line segments indicating constant pitch, with length indicating note duration. For more details on melodic contour modeling we refer to [6].

6 Sonification

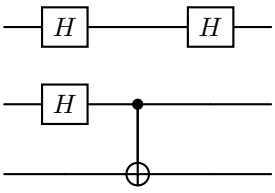
We have set up some correspondences between musical elements, which we have collectively referred to as "musical state", and numerically derived vectors which can be embedded in quantum states. Although the digital signal processing of audio in quantum states is not fully developed, we produce here some examples inspired by these correspondences. The first step is to choose a simple B-spline representation with 8 coefficients, which we call $\mathbf{c}$:

$$\mathbf{c} = (0.4, -0.2, 0.2, 0.6, -0.4, 0.4, 0.2, -0.2)$$

As mentioned earlier, we have chosen this vector to have the Euclidean normalization:

$$c_0^2 + c_1^2 + c_2^2 + c_3^2 + c_4^2 + c_5^2 + c_6^2 + c_7^2 = 1.$$

We also refer to the cubic B-spline graph generated with these coefficients as $\mathbf{c}$ (for contour). As an exercise in sonification, we begin with a basic quantum circuit (taken from [7] page 118).



The quantum circuit above takes as input any linear combination of the computational basis states:

$$\{|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle\}$$

The unitary 8×8 matrix that implements this circuit is show here as A, which can also be seen (by taking successive powers) to have multiplicative order 8:

$$A = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \end{pmatrix}$$

In figures 2-8 we show the evolution of this contour by iteration of the unitary matrix A.

This sequence of contours can be used to describe timbres and melodies which evolve on time scales of milliseconds and seconds respectively. Abstractly, we can think of these musical states as influenced, at least conceptually, by the quantum circuit. (Accompanying music composition is forth-coming.)

Our implementation and generation of musical fragments, such as tones and melody, is currently using python and PyTorch. These scripts generate audio samples directly from spline models and accompanying data descriptions to guide numerical choices. In most cases of tone generation, instrument samples are used to extract cycles. We look forward making the connection between this set of tools and quantum computing libraries such as Qiskit.

Figure 1: basic contour $\mathbf{c}$

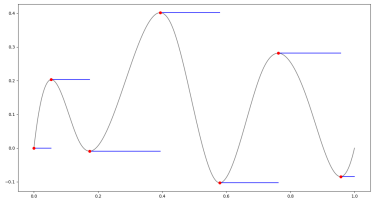


Figure 2: $A\mathbf{c}$

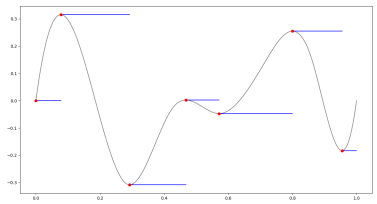


Figure 3: $A^2\mathbf{c}$

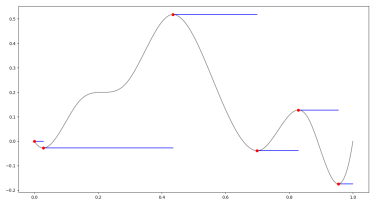


Figure 4: $A^3\mathbf{c}$

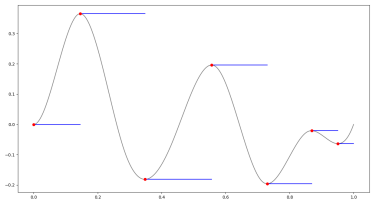


Figure 5: $A^4\mathbf{c}$

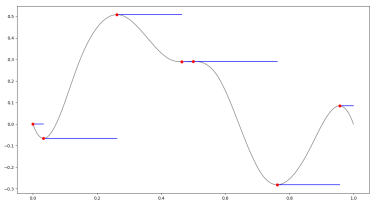


Figure 6: $A^5\mathbf{c}$

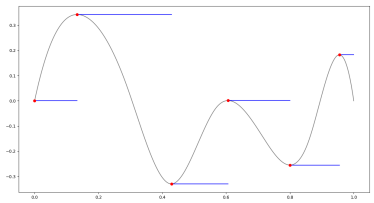


Figure 7: $A^6\mathbf{c}$

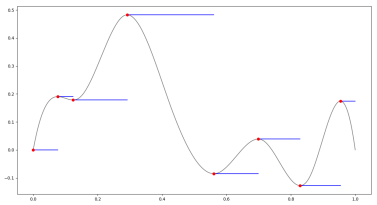
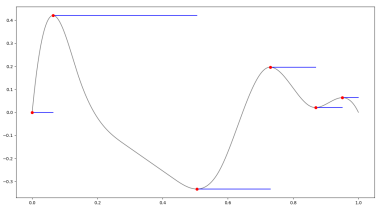


Figure 8: $A^7\mathbf{c}$



7 Conclusions and Future Work

We have shown that there are some natural ways to set up correspondences between numerical representations of musical elements such as timbre and melody, based on the notions of cycles and melodic contours, and qubit registers in the quantum domain. We propose to study this correspondence further, and in particular to begin to map out the digital signal processing that is necessary to make these techniques part of a broader sonification of quantum circuits and algorithms.

Audio and Music are well-situated to be important testing grounds for quantum processing. Audio signal processing has a rich history of hybrid techniques, with analog and digital signal processing intertwined (see [8]). We hope to see this continue with the addition of quantum signal processing. Quantum algorithms have shown exponential advantage in several areas, such as search and factoring, and we look forward to seeing such advances also in signal processing. In order to fully develop our approach to signal modeling with splines, it is necessary to solve linear systems to obtain B-spline models of cycles. This suggests that the approach of HHL (see [9]) could play a key role in data acquisition in quantum digital audio representations.

References

[1] Klassen, M.: *Spline Modeling of Audio Signals and Cycle Interpolation*, Mathematics and Computation in Music, MCM 2022, Springer, LNAI 13267 (Lecture Notes in Artificial Intelligence), Montiel, M. et al, eds., pages 155-167.

[2] Itaborai, P., Miranda, E.: *Quantum Representation of Sound: from mechanical waves to quantum circuits*, arXiv:2301.01595v1, January 1, 2023.

[3] Collins, N.: *SplineSynth: An Interface to Low-Level Digital Audio*. Proceedings of the Diderot Forum on Mathematics and Music, Vienna, 1999, ISBN 3-85403-133-5, pp 49-61.

[4] Ardaillon, L., Degottex, G., Roebel, A.: *A multi-layer F0 model for singing voice synthesis using a B-spline representation with intuitive controls*. Interspeech 2015, Sep 2015, Dresden, Germany. hal-01251898v2.

[5] de Boor, C.: *A Practical Guide to Splines*, revised edition, Springer-Verlag, New York (1980)

[6] Klassen, M., Lanthier, P.: *Melodic Contour Generation with Spline Models of Cycles*, Mathematics and Computation in Music, MCM 2024, Springer, LNCS 14639 (Lecture Notes in Computer Science), Noll, T. et al, eds., pages 210-222.

[7] Buchmann, J.: *Introduction to Quantum Algorithms*, Pure and Applied Undergraduate Texts 64, American Mathematical Society, 2024

[8] Chuang, I., Liu, Y., Martyn, J., Sinanan-Singh, J., Smith, K., Girvin, S.: *Toward Mixed Analog-Digital Quantum Signal Processing: Quantum AD/DA Conversion and the Fourier Transform*, arXiv:2408.14729v1, August 27, 2024.

[9] Harrow, A., Hassidim, A., and Loyd, S.: *Quantum algorithm for linear systems of equations*, arXiv:0911.3171v3, September 30, 2009.