Generating Quadratic Difference Tone Spectra for Auditory Distortion Synthesis

Esteban Gutiérrez
Department of Information and Communications Technologies
Universitat Pompeu Fabra
egutierreo@mat.uc.cl

Christopher Haworth
Department of Music
University of Birmingham
c.p.haworth@bham.ac.uk

Rodrigo F. Cádiz
Music Institute and Department of Electrical Enginering
Pontificia Universidad Católica de Chile
rcadiz@uc.cl

ABSTRACT

Quadratic difference tones are one of a family of perceptual phenomena that arise from the neuromechanics of the auditory system in response to particular properties of physical sound. Long deployed as 'ghost tones' by improvisers, computer musicians and sound artists, in this paper we address the problem of creating quadratic difference tone spectra, where a QDT fundamental and harmonic overtone series is specified and the necessary acoustic components needed to evoke it are synthesised. A numerical algorithm for solving target distributions of amplitudes for the synthesis of quadratic distortion tone spectra is proposed. The algorithm aims to find a solution for a target distribution of amplitudes that matches the desired spectrum as closely as possible. The experiments were conducted using different parameter settings and target distributions. The results show that the algorithm is effective in solving the problem in the majority of cases, with at least 99% of the cases being solvable in real-time. The article also discusses the convergence of the algorithm and its potential mathematical properties. Additionally, audio examples implemented in Max are provided to demonstrate the synthesis of different quadratic distortion tone spectra using this approach.

1. INTRODUCTION

Auditory distortion products are perceptual illusions which arise from the neuromechanics of the auditory system. Often referred to as difference, sum or combination tones, these ghostly sounds have a long history as objects of musical intrigue, from their discovery by Giuseppe Tartini in 1754, to their contemporary use by improvisers, sound artists and computer musicians. In a hearing science context, evoked auditory distortion products have served as a key method used to probe the workings of the cochlea. Once believed to be caused when the otherwise linear mechanics of the auditory system were driven into a nonlinear region by high intensity sound, today they are understood to arise as byproducts of the active amplification process provided by the outer hair cells on the basilar membrane. Indeed, auditory distortion products are synonymous with the later-named 'evoked distortion product otoacoustic emissions' (DPOAEs) [1], so-called because the fluid disturbance in the cochlea backpropogates through the ear drum and middle-ear, and can be measured with a microphone in the ear canal.

Auditory distortion products (ADPs) have a rich musical history spanning experimental music, jazz improvisation, computer music, drone and noise music, and varieties of sound art. Their use varies between the fleeting, transient 'ghost' tones experienced in the improvisations of wind players like John Butcher and Evan Parker; to the disorienting, texturally dense static sound installations of La Monte Young and Catherine Christer Hennix; to the microtonal inner-ear counterpoint of Maryanne Amacher. In terms of ADPs relevance to computer music, a notable factor is their distinctive spatial imagery. When reproduced over loudspeakers, auditory distortion can be heard as a separate auditory stream localising close to the head, separate from the acoustic tones. This happens because the distortion tones are generated by the direct interaction of the acoustic components on the basilar membrane. Lateralization of the distortion tones is governed by Interaural Time (ITD) and Interaural Intensity Differences (IID) alone; head related transfers matter only insofar as they affect IIDs. Indeed, the closest comparable listening experience is headphone reproduction, where the auditory image appears inside and around the head, and is highly sensitive to head movements. But, crucially, this only happens when the acoustic tones are reproduced over loudspeakers. Over headphones, distortion tones can be difficult to hear as a separate stream.

Typical studies of distortion products deploy two sinusoidal signals (primaries) having frequencies f_1 and f_2, such that $f_2 > f_1$. In these conditions, the two most prominent tones that can be observed are the lower cubic difference tone (LCDT), which occurs at $2f_1 - f_2$, and obeys a cubic law distortion; and the quadratic difference tone (QDT), which occurs at $f_2 - f_1$, and obeys square law distortion. The LCDT is tuned, and to hear it requires that the ratio of the acoustic tones be in the range $1 < f_2/f_1 < 1.3$ [2, 514]. The highest level results from the lowest ratio, and it falls off quickly thereafter (Ibid). The level of the QDT, on the other hand, is only partially dependent on the ratio of the acoustic tones, and so retains its amplitude across large changes in interval size. Moreover, its distance from the primaries makes it much easier to perceptually segregate than the LCDT (see e.g. [3], [4] and [5].)

At the same time, audibility of the QDT is highly depen-

dent on intensity. Greater than 50db spl is required for the component to be heard at all, and its amplitude grows proportionally to the level of the primaries, such that for every 1db increase the QDT grows by 2db [2, 514]. In [6], the authors described methods for ameliorating the listening fatigue caused by the high intensity needed to hear the QDT. Drawing on a method presented in [7], they utilised a sinusoidal complex with constant difference frequencies. They found that each adjacent pair of sinusoids produces the identical QDT frequency, adding linearly to its total gain, and thereby increasing the level of the distortion tone. On top of this, other QDTs are produced among the components, producing multiple harmonics of the fundamental (e.g. $f_2 - f_1, f_3 - f_1, f_4 - f_1$ and so on for all combinations). In [7], the authors reported that 'an harmonic complex tone . . . can produce a sizeable DS [distortion spectrum], even at moderate to low sound levels'. Thus, increasing the number of components, and spreading them out over a wider frequency range, permits the subjective level of the acoustic tones to be reduced, without compromising the audibility of the QDT.

This paper describes a method for direct synthesis of quadratic difference tone spectra (QDTS). We look at how to specify the fundamental frequency and relative amplitudes for an arbitrary number of harmonics, so as to generate complex QDT spectra. Through this, we are able to match the QDT spectra to a target timbre with considerable accuracy in particular cases. In section 2 the model used to synthesize QDTSs is explained. Section 3 contains a discussion regarding the well definition of the mathematical problem involved and a numerical procedure to solve it. Section 4 contains a brief discussion about the implementation of our algorithm in MaxMSP. Finally, Sections 5 and 6 discuss how the algorithm was adjusted and explain the examples included with this article respectively.

2. MODELLING THE QDT SPECTRUM

The problem of how to exert greater control of the timbre of the QDT spectra has been theorized in [6]. There it is stated that in order to generate a QDT from a complex of pure tones with frequencies $F, 2F, \ldots, (n-1)F$ and *target* respective amplitudes $t_1, t_2, \ldots, t_{n-1}$, it is necessary to have a *carrier* complex of n pure tones with the frequencies $C, C+F, C+2F, \ldots, C+(n-1)F$, where $C > 0$, and respective amplitudes $x_0, x_1, \ldots, x_{n-1}$, such that the following system of equations is satisfied

$$\begin{array}{rcl} t_1 & = & x_0 x_{n-1} \\ t_2 & = & x_0 x_{n-2} + x_1 x_{n-1} \\ \vdots & & \vdots \\ t_{n-1} & = & x_0 x_1 + x_1 x_2 + \cdots + x_{n-2} x_{n-1} \end{array} \tag{1}$$

(for a graphic representation of this setting see figure 1).

Synthesising auditory distortion products using this approach is in theory capable of using QDTs to produce harmonic spectrums of arbitrary order, a result that goes beyond the 3-4 harmonics achieved in [6]. Because of the new control of timbre this affords, we have decided to call this method *quadratic distortion tone spectra* (QDTS) synthesis.

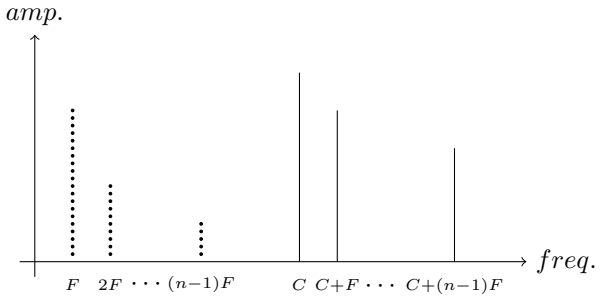

Figure 1. On the right, the amplitude and pitches of the *carrier* complex of pure tones are shown as continuous segments. On the left, the amplitude and pitches of the *target* complex of pure tones produced by QDT are shown as dotted segments. For comparison with the typical notation of combination tones used in the introduction, here $C = f_1$ and $C + F = f_2$, so that $F = f_2 - f_1$ symbolizes the difference tone.

Observe that the system (1) has $n-1$ equations and n variables, so with the purpose of simplifying computations and using some particular numerical methods we decided to add an equation that fixes the amplitude of the first pure tone of the *carrier* complex to, let us say t_0.
Finally, and for the sake of order let us define the following n polynomials

$$\begin{array}{rcl} p_0 & = & x_0 - t_0 \\ p_1 & = & x_0 x_{n-1} - t_1 \\ p_2 & = & x_0 x_{n-2} + x_1 x_{n-1} - t_2 \\ \vdots & & \vdots \\ p_{n-1} & = & x_0 x_1 + x_1 x_2 + \cdots + x_{n-2} x_{n-1} - t_{n-1}, \end{array} \tag{2}$$

and the polynomials $d_j = p_j + t_j$ for $j = 1, \ldots, n-1$, and observe that:

1. The *distortion* function $D = (d_1, \ldots, d_{n-1})$ maps the amplitudes of the *carrier* complex into the amplitudes of the auditory distortion products as a result.

2. Solving the system (1) together with the condition $x_0 = t_0$ is, of course, equivalent to finding a root for the system $P = (p_0, p_1, \ldots, p_{n-1}) = (0, 0, \ldots, 0)$.

3. EXISTENCE OF SOLUTIONS AND HOW TO FIND THEM

In this section the existence of the *carrier* complex of pure tones together with some ways of finding such solutions are discussed.

3.1 About the existence

As seen in [6], finding a root for P can be done by hand for $n = 2, 3$ and 4, however after this point the problem gets increasingly complicated.
The problem of finding whether a system of polynomials in many variables has a solution is not trivial, and the state-of-the-art theory in this matter is known to be complicated and computationally expensive. The most widely used algorithm to solve this issue consists in finding a mathematical object called Gröbner basis, which strongly depends on the problem, and then use it to determine the number

of solutions of the problem (see for example [8]). Since we are interested in solving the system for any choice of *target* amplitudes $(t_1, \ldots, t_{n-1})$, it would be necessary to compute the Gröbner basis considering the constants t_j as variables, which is rather problematic, since this roughly duplicates the number of variables and the computational complexity of computing a Gröbner basis has been estimated to be double exponential in the number of variables (see [9] and [10]).
With the last being said, we have been able to compute a Gröbner basis for the system when $n = 2, ..., 6$ using an implementation of Faugère algorithm (for a brief summary of the results see 5.1), which has lead us to determine that in this cases there is always at least one complex solution and that the amount of solutions is always finite.
Proving only the existence of complex solutions is by no means our goal, however, one can prove by hand that if $(a_0 + ib_0, a_1 + ib_1, \ldots, a_{n-1} + ib_{n-1})$ is a complex solution for the problem, then

$$\begin{aligned} D(a_0, a_1, \ldots, a_{n-1}) - D(b_0, b_1, \ldots, b_{n-1}) \\ = \quad (t_1, t_2, \ldots, t_{n-1}), \end{aligned} \tag{3}$$

that is, assuming that one can control the phase of the auditory distorted products, one could use phase cancellation to produce the *target* auditory distortion products by the superposition of two complex pure tones using the real and imaginary parts of the complex solution.
This approach has two problems:

1. little can be found in the current literature about determining the phase of ADPs, and
2. even though this could potentially generate the target auditory distortion products accurately, any implementation of this would still require to first compute the complex solution, which is in any case double as much work as finding a real solution provided that such solution actually exists.

To end this section, let us remark that discerning whether there are real solutions is an even more challenging problem than simply finding if there is a solution at all. Actually, even though such a problem can always be theoretically solved according to Tarski theorem (see for example [8]), the state-of-the-art algorithms used to solve this issue efficiently start by finding a Gröbner basis, which is still a problem. Moreover, we are already aware that not much more can be said in the general case since even when n is small, we have been able to find examples that do not have a real solution at all (*e.g.* consider $n = 5$ and $(t_0, t_1, t_2, t_3, t_4) = (1, 1, 1, 1, 1)$), so a reasonable goal would be to find a criterion that solves the question if there is a real solution or not before we even start looking for it.

3.2 Algorithm

The absence of real solutions for the system $P = 0$ appears to be a dead-end problem. However, we propose a workaround algorithm designed to avoid null returns. The algorithm first tries to find a solution for a given initial *target* distribution of amplitudes. Then, if it does not find it fast enough, it restarts the process with a new *target* that is chosen randomly in a small neighbourhood of the initial *target*, and repeats such process until a solution to the case with the new *target* is found.
An algorithm such as the one described before does not have any reason to converge in general. However, we have implemented it using the Newton-Raphson algorithm in the part where the solution is sought, and not even a single case where this does not work could be found in our experiments (see 5.2 and 5.3 for the details of the experiments), which gave us enough reasons to implement a solution for the problem using this approach.
Since we are interested in using this algorithm in real-time, we decided to try some other techniques in order to optimize the algorithm as much as possible. These techniques are described here:

1. The initial condition of the Newton-Raphson algorithm can be chosen to be:
   (a) the vector $(0.5, 0.5, \ldots, 0.5)$, since this is the average of the vectors in the ideal space of solution $[0, 1]^n$, or
   (b) a randomly chosen vector in the ideal space of solutions $[0, 1]^n$.
2. If a solution is not found using the Newton-Raphson algorithm, the process will restarts with a new initial condition (that has to be chosen randomly to avoids infinite loops) and a new *target* distribution of amplitudes, which can be chosen to be:
   (a) changed to a new *target* distribution of amplitudes that is close enough to the initial *target*.
   (b) changed to a new *target* distribution of amplitudes that is close enough to the **last** *target* used.

Every combination of techniques described was largely tested, and even though there was no objectively best option, we have decided to implement the combination of techniques given by 1.(b) and 2.(a), because of its speed and stability (see subsection 5.4 for the details of the experiments and results), giving algorithm 1 as a result.

4. IMPLEMENTATION IN MAX

In order to implement our algorithm in an efficient way and in a friendly environment, we decided to use the Min dev-kit, a software development kit containing an example package with the current best practices for package creation using modern C++ code. More precisely, using the Min dev-kit we were able to build an external object for Max that implements our algorithm in an efficient way and that can be freely downloaded from our GitHub repository [1] .
The resulting external object was named `qdt.solver`, and its main utility is solving the system of equations (2) considering $x_0 = t_0 = 1$. `qdt.solver` has one input and two outputs: the input is a Max list of integer or floating numbers that represent the *target* distribution of amplitudes; the left output is a Max list containing the normalized estimation for the amplitudes of the *carrier* complex

[1] The GitHub repository can be accessed here `https://github.com/cordutie/QDTS`

Algorithm 1 QDT solver

1: Define the initial target $T_{\text{init}} \in \mathbb{R}^n$, the error tolerance $\varepsilon > 0$ and the number of iterations tolerance $N \in \mathbb{N}$
2: Define the initial conditions:
- $X =$ Random vector in $[0,1]^n$
- $T = T_{\text{init}}$

3: **while** $||D(X) - T|| > \varepsilon$ **do**
4: Run the Newton-Raphson algorithm on the function $D(\cdot) - T$ with intial condition X updating it in each iteration until either $||D(X) - T|| \leq \varepsilon$ or the number of iterations gets to N.
5: **if** $||D(X) - T|| \geq \varepsilon$ **then**
6: Redefine the initial condition
- $X =$ Random vector in $[0,1]^n$
- $T = T_{\text{init}} + \varepsilon \cdot R$, where $R =$ Random vector in $[0,1]^n$

7: **end if**
8: **end while**

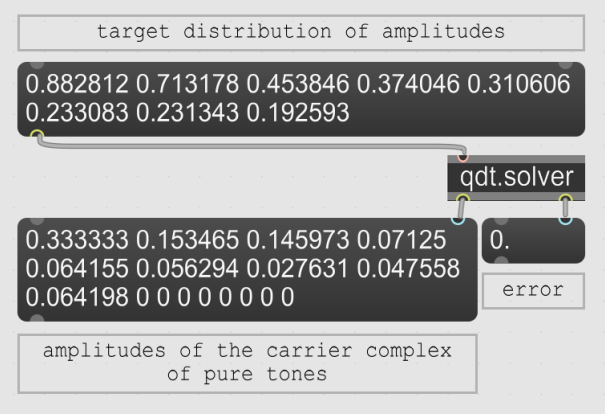

Figure 2. The `qdt.solver` object is used to find a solution for the *target* distribution of amplitudes at the top of the figure. At the bottom of the figure, the normalized solution together with the estimation error can be seen.

of pure tones necessary to generate the *target* distribution of amplitudes inputted; and the right output is a Max message containing a float number that represents the ℓ^2-error of the estimation, which can be thought as the energy of the difference between the targeted signal and the theoretical auditory distortion product described by QDT produced by the estimated *carrier* complex of pure tones (see figure 2). Additionally, whenever a solution could not be found in a reasonable amount of time, a message is sent to the Max console warning that experimental methods were used to approximate a solution in order to avoid Max from freezing.

Since most experiments were conducted using $n = 8, 12$ and 16, we also made three patches with a fixed amount of harmonics in the *target* distribution of amplitudes corresponding to these numbers. Such patches were made in a way that when loaded using the `bpatcher` object in Max, a simple but functional GUI is shown (see figure 3) whose main purpose is to make the usage of the external as simple as possible.

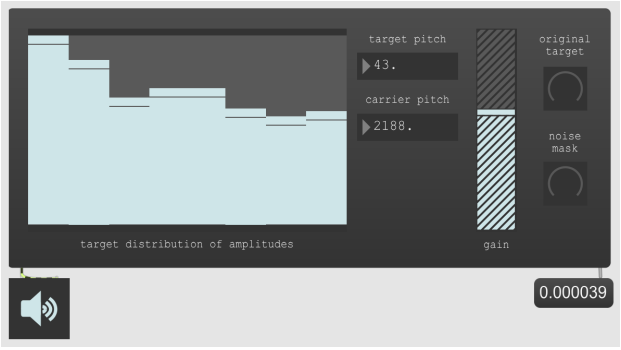

Figure 3. Example patch considering $n = 8$ loaded with the `bpatcher` object. In this example the patch is used to synthesize an Auditory Distortion Product with a frequency of 43Hz and a particular distribution of harmonics through a carrier complex of pure tones with frequencies 2188Hz, 2231Hz, 2274Hz, ..., 2489Hz and having an ℓ^2-error of 0.000039.

5. ADJUSTING THE ALGORITHM

In this section, the main experiments conducted to adjust the algorithm to its optimal form are described.

5.1 Experiment 1. Finding a Gröbner Basis

In order to find a Gröbner basis for the system of equations we tried several approaches. The following are some of the instances that we implemented.

1. Taking advantage of the fact that the computation complexity of a Gröbner Basis has been computed with accuracy in the case of homogeneous polynomials (see [11]), we homogenized the system by changing every t_j with t_j^2 and removing the polynomial p_0. This does not change the solutions of the system, since the targets are expected to be positive numbers, and one can simply add p_0 to the basis once it is computed to add conditions on x_0. This in fact works for the cases where we could find a Gröbner basis, and adding the polynomial p_0 is fundamental to reduce the dimension of the problem, that is, to make it a system with a finite amount of solutions.

2. Since in practice one can fix at least one variable, we tried $x_0 = t_0 = 1$. Also, since this implies that $p_1 = x_{n-1} - t_1$ and $p_2 = x_{n-2} + x_1 x_{n-1} - t_2$, one can easily replace $x_{n-1} = t_1$ and $x_{n-2} = t_2 - t_1 x_1$. This reduces the number of variables in 4 and does not change the total degree of the system, which is extremely useful since the complexity of computing a Gröbner basis has been estimated to be double exponential in the number of variables with base being the total degree of the system (see [9], [10]).

In both cases, we used the implementations of Buchberger and F5 algorithms in the Python package `sympy`, however after running every combination in parallel in the High Performance Computing Cluster at the School of Engineering of UC for over a couple of months, just the cases with $n = 2, 3, 4, 5$ and 6 have successfully finished. Moreover, the case $n = 7$ seems to be using a huge amount of ram, implying that the Gröbner basis for such a case is extremely large and making its use in real-time non-viable in a compact environment.

5.2 Experiment 2. Adjusting Newton-Raphson algorithm to the problem

The Newton-Raphson algorithm is known for its efficiency provided that it is known to converge, however since in our case we do not have certainty of its convergence, it is important to decide how many iterations will be tolerated before the process restarts with different initial condition and target.
In order to find a reasonable tolerance, we conducted an experiment that estimate the number of iterations needed for the algorithm to converge provided that it converges. For this task, we made a sample of 10^6 *target* distributions of amplitudes chosen at random in the space $[0,1]^n$ for each of the cases $n = 8, 12$ and 16. For each *target*, we tried the algorithm with random initial conditions chosen in the space of ideal solutions $[0,1]^n$ up to 10 times each, and for the cases that could be solved with these constraints, the number of iterations needed to find a solution was saved. A summary of these results can be found in table 1.
As a conclusion for the results found in this experiment, we decided that the tolerance will be fixed to satisfy at least 99% of the cases, that is, the algorithm will restart once it reaches the percentile 99 found in table 1 for each case. Finally, it is remarkable that with the data obtained and using a simple test of proportions, we can conclude that with a confidence of 99%, the proportion of the cases that can be solved in less than 10 tries (which is extremely fast and can be computed in real-time) correspond to at least the 99.45% of the cases.

n	mean $\pm$ sd	Percentiles $50, 90$ and 99
8	$12.55 \pm 12,9$	$(8, 26, 72)$
12	$17,41 \pm 15,92$	$(11, 38, 81.1)$
16	$21,26 \pm 18.21$	$(14, 47, 88)$

Table 1. In this table the results concerning the number of iterations needed to solve the problem provided that it could be solved in less than 10 tries are shown. In each case, a sample of 10^6 *target* distributions of amplitudes was used, and the number of samples that could be solved in less than 10 tries correspond to $99.46\%, 99.89\%$ and 99.98% of the total corresponding to $n = 8, 12$ and 16.

5.3 Experiment 3. Empirical proof of conjecture

As discussed in subsection 3.2, the algorithm in this paper iterates until it finds a solution for a *target* distribution of amplitudes that is close enough to the actual *target* given. The fact that this algorithm actually converges is non-trivial, so before implementing the algorithm in Max we made some tests trying to find a case that is unsolvable. For this task, we fixed a maximum acceptable error corresponding to 2% of the energy of the signal to synthesize and ran the algorithm for 10^6 randomly chosen *target* distributions of amplitudes, saving the ones that could not be solved in less than 10 tries. Finally, we tried the algorithm again for the unsolved cases but this time adding a, small enough in magnitude, random vector and saved the ones that could not be solved again in less than 100 tries. These experiments was made for the cases $n = 8, 12$ and 16, and in each case, the result was the same: the set of unsolvable cases after considering the addition of a small random vector is empty.
Before going to the next experiment it is important to mention that the fact this algorithm converges could be explained by certain mathematical properties of the system, e.g. the set of unsolvable (in $\mathbb{R}^n$) cases could have an empty interior. However, we have not been able to prove this fact yet.

5.4 Experiment 4. Optimizing the algorithm

One can adjust the algorithm in at least two way: how to chose the initial condition and how to change the *target* at restart. The techniques that we considered for each one of these adjustments were described at the end of subsection 3.2, and the results for some testings made considering 10^6 *target* distributions of amplitudes chosen at random in the space $[0,1]^n$ for each of the cases $n = 8, 12$ and 16 can be found in tables 2 and 3.

n	Technique	mean $\pm$ sd	Percentiles $50, 90$ and 99
8	(a,a)	51.65 ± 721.47	$(7, 78, 438)$
	(a,b)	38.06 ± 176.04	$(7, 78, 459)$
	(b,a)	50.65 ± 944.41	$(8, 54, 427)$
	(b,b)	34.69 ± 175.26	$(8, 53, 437)$
12	(a,a)	34.19 ± 392.61	$(10, 87, 256)$
	(a,b)	31.36 ± 100.24	$(10, 87, 252)$
	(b,a)	$34,42 \pm 406.76$	$(13, 86, 246)$
	(b,b)	31.96 ± 93.71	$(13, 86, 248)$
16	(a,a)	35.49 ± 426.71	$(13, 97, 242)$
	(a,b)	33.79 ± 60.04	$(13, 97, 251)$
	(b,a)	36.84 ± 78.97	$(17, 98, 237)$
	(b,b)	36.83 ± 65.66	$(17, 98, 237)$

Table 2. Results concerning the **amount of iterations** needed to solve the problem. In each case, a sample of 10^6 *target* distributions of amplitudes was used. Technique (x,y) means that the techniques 1.(x) and 2.(y) were used in that case.

n	Technique	mean $\pm$ sd $\left(\times 10^{-6}\right)$	Percentiles $50, 90$ and 99 $\left(\times 10^{-6}\right)$
8	(a,a)	16.33 ± 23.89	$(2.96, 51.7, 95)$
	(a,b)	162.56 ± 2646.66	$(2.79, 69.1, 995)$
	(b,a)	14.65 ± 23.24	$(1.82, 49.2, 94)$
	(b,b)	160.69 ± 2610.3	$(1.88, 62.2, 813)$
12	(a,a)	15.79 ± 23.8	$(2.37, 50.85, 95.31)$
	(a,b)	51.55 ± 1032.92	$(2.39, 61.9, 242.39)$
	(b,a)	15.19 ± 23.48	$(1.94, 49.64, 94.31)$
	(b,b)	46.22 ± 1008.52	$(1.97, 58.56, 225)$
16	(a,a)	16 ± 23.87	$(2.26, 50.88, 95.04)$
	(a,b)	24.1 ± 410	$(2.31, 60.2, 171.83)$
	(b,a)	15.82 ± 23.82	$(2.11, 49.93, 95.31)$
	(b,b)	26.17 ± 443.42	$(2.16, 58.6, 160.05)$

Table 3. Results concerning the **squared error of the estimation** given by the solutions of the algorithm. In each case, a sample of 10^6 *target* distributions of amplitudes was used. Technique (x,y) means that the techniques 1.(x), and 2.(y) were used in that case. For comparison, an error of 400×10^{-6} corresponds to 2% of the signal energy.

6. SOUND EXAMPLES

Together with this article, we have created some videos [2] showing a series of sound examples that were built using the `qdt.solver` object. A brief explanation for each of the sound examples can be found in this section. In order to hear examples 1 to 5, any reasonable loudspeaker at a moderate level should work just fine, however example 6 will require to increase the volume a little further so that the QDTS is audible.

6.1 Example 1. Producing a sequence of pitches with the `qdt.solver` object

Example 1 consists of the synthesis of a little melody known popularly as *the lick* (see [12] and [13]) in the key of A. The audio example contains the melody synthesized using additive synthesis, the combination of the additive and QDTS synthesis versions, and the QDTS synthesis alone for comparison.
For the QDTS synthesis, the computations were made using the `qdt.solver` object together with a *carrier* complex of pure tones with $C = 2470$Hz.

6.2 Example 2. Masking the auditory distortion products with filtered noise

Example 2 consists of the synthesis of a single pitch overlapped with filtered noise. The audio example contains the single pitch synthesized using additive synthesis and QDTS synthesis first and then overlaps it with a high-passed filtered noise. The effect of the filtered noise with the synthesized pitch using additive synthesis is simply hearing both sounds, however, when the filtered noise is played together with the QDTS synthesis, the synthesized pitch gets muted because of the disorder around the *carrier* complex of pure tones.
For the QDTS synthesis, the computations were made using the `qdt.solver` object using a *carrier* complex of pure tones with $C = 2338$Hz in order to synthesize a single pitch with frequency $F = 55$Hz and a particular harmonic distribution. The noise was filtered muting all frequencies below approximately 1100Hz.

6.3 Example 3. Amplitude modulation and Auditory Distortion Products

Example 3 consists of the synthesis of an amplitude-modulated single pitch. The audio example contains the amplitude-modulated pitch synthesized using additive synthesis, the combination of the additive and QDTS synthesis versions, and the QDTS synthesis alone for comparison. It is important to note that since the Quadratic Tones arise from the Distortion function $D = (d_1, \ldots, d_{n-1})$ and such function is homogeneous of degree 2 one has that

$$D(\lambda x) = \lambda^2 D(x),$$

so in order to apply amplitude modulation, one has to use the square root of the usual AM carrier.
For the QDTS synthesis, the computations were made using the `qdt.solver` object using a carrier complex of pure tones with $C = 2338$Hz in order to synthesize a single pitch with frequency $F = 55$Hz and a particular harmonic distribution. The AM carrier used was a simple LFO of the form

$$L_1(t) = A\sin(\omega t + \varphi) + B,$$

for some $A, B, \omega, \varphi \in \mathbb{R}$ in the additive synthesis case, and the squared root version of the LFO

$$L_2(t) = \sqrt{A\sin(\omega t + \varphi) + B},$$

for its QDTS counterpart.

6.4 Example 4. Frequency Modulation and Auditory Distortion Products

Example 4 consists of the synthesis of a frequency-modulated single pitch. The audio example contains the frequency-modulated pitch synthesized using additive synthesis, the combination of the additive and QDTS synthesis versions, the QDTS synthesis alone, and a QDTS synthesis using frequency modulation on the *carrier* instead of the *target* for comparison. The result shows that frequency modulation is compatible with QDTS synthesis in certain ranges. Moreover, when the *carrier* complex of pure tones is frequency modulated, the auditory distortion product holds its pitch, since the difference of the frequencies of the *carrier* complex of pure tones holds the same, however, some amplitude changes can be heard in this case, showing the instability of the auditory distortion products with respect to the *carrier* frequencies.
For the QDTS synthesis, the computations were made using the `qdt.solver` object using a carrier complex of pure tones with $C = 2500$Hz in order to synthesize a single pitch with frequency $F = 55$Hz and a particular harmonic distribution. The FM carrier used was a simple LFO of the form

$$L(t) = A\sin(\omega t + \varphi) + B,$$

for some $A, B, \omega, \varphi \in \mathbb{R}$ in both the additive and QDTS synthesis case.

6.5 Example 5. Resynthesizing existing sounds with Auditory Distortion Products

Example 5 consists of the resynthesis of a real recording of a tuba through QDTS synthesis. The audio example contains the real tuba recording, a resynthesis of the real recording using a combination of 8 sinusoids with a pitch detector, a combination of the resynthesis made with the 8 sinusoids and with the QDTS synthesis, the combination of the real recording with the resynthesis using QDTS synthesis versions, and finally the resynthesis using QDTS synthesis alone. The results show that certain signals can be at a certain level be reconstructed using QDTS synthesis.
For the QDTS synthesis, the computations were made using the `qdt.solver` object using a carrier complex of pure tones with $C = 1979$Hz in order to resynthesize the recording of a tuba. The harmonic distribution was precomputed using the Discrete Fourier Transform of a small bin of the real recording. The pitch and envelope detection of the tuba recording were made in real-time using the

[2] All videos are available in the GitHub repository `https://github.com/cordutie/QDTS`

`sigmund~` object. Finally, the square root of the envelope of the real signal was used in the QDTS synthesis as in Example 3 (see 6.3).

6.6 Example 6. Resynthesizing the same existing sound with different Auditory Distortion Products

Before start describing this example, it is important to mention that here we are resynthesizing a much higher in pitch signal than in the previous example, and hence to hear this Auditory Distortion Products it is required to increase the volume of the loudspeakers used.

Example 6 consists of the resynthesis of a real clarinet recording through various Auditory Distortion Products. The audio example contains the real clarinet recording followed by three resynthesis of it using three different QDTS synthesis settings. The results show that the same signal can be recreated using different QDTS synthesis settings allowing to have at least some control on the timbre of the *carrier* complex of pure tones.

For the QDTS synthesis, the computations were made using the `qdt.solver` object using a carrier complex of pure tones with $C = 2345$Hz. The computations were ran several times until three reasonably different solutions appeared. The harmonic distribution was precomputed using the Discrete Fourier Transform of a small bin of the real recording. The pitch and envelope detection of the tuba recording were made in real-time using the `sigmund` object. Finally, the square root of the envelope of the real signal was used in the QDTS synthesis as in Example 3 (see 6.3).

7. CONCLUSIONS

The results presented in subsection 5.2 show that, for each case (n = 8, 12, 16), we can conclude that with a confidence of 99%, the proportion of the cases that can be solved in less than 10 tries (which is extremely fast and can be computed in real-time) correspond to at least the 99.45% of the cases. In addition, experimental tests were conducted to find unsolvable cases, but in all cases (n = 8, 12, 16), the addition of a small random vector allowed the algorithm to solve all previously unsolvable cases. However, the mathematical properties of the system that explain why the algorithm converges have not been proven yet.

The algorithm presented can be optimized by adjusting the initial condition and the change in target at restart. Test results for these adjustments were presented in Tables 2 and 3, using 10^6 randomly chosen target distributions of amplitudes.

The audio examples provided in the paper demonstrate the synthesis of different target QDTS's using our algorithm. Overall, the experimental results suggest that the algorithm is highly effective in matching a target timbre to a QDTS. In particular, instrumental sounds having very low fundamental frequencies fare well with the method. In future, the algorithm could be expanded to incorporate live audio analysis and resynthesis of a sound signal.

Acknowledgments

This research was partially funded by ANID Fondecyt Grant #1230926, Government of Chile.

8. REFERENCES

[1] D. T. Kemp, "Stimulated acoustic emissions from within the human auditory system," *The Journal of the Acoustical Society of America*, vol. 64, no. 5, pp. 1386–1391, 1978.

[2] W. M. Hartmann, *Signals, sound, and sensation*. Springer Science & Business Media, 2004.

[3] J. B. Dewey, "Cubic and quadratic distortion products in vibrations of the mouse cochlear apex," *JASA Express Letters 2*, vol. 11, no. 114402, 2022.

[4] E. Zwicker, "Different behaviour of quadratic and cubic difference tones," *Hearing Research*, vol. 1, no. 4, pp. 283–292, 1979.

[5] R. Plomp, "Detectability threshold for combination tones," *The Journal of the Acoustical Society of America*, vol. 37, no. 1110, 1965.

[6] G. Kendall, C. Haworth, and R. F. Cádiz, "Sound synthesis with auditory distortion products," *Computer Music Journal*, vol. 38, no. 4, 2014.

[7] D. Pressnitzer and R. Patterson, "Distortion products and the perceived pitch of harmonic complex tones," *Physiological and psychophysical bases of auditory function*, pp. 97–104, 2001.

[8] S. Basu, R. Pollack, and M. F. Roy, *Algorithms in Real Algebraic Geometry.* Springer-Verlag, 2005.

[9] T. W. Dubé, "The structure of polynomial ideals and gröbner bases," *SIAM Journal on Computing*, vol. 19, no. 4, 1990.

[10] A. R. Meyer and E. W. Mayr, "The complexity of the word problems for commutative semigroups and polynomial ideals," *Advances in Mathematics*, vol. 46, no. 3, pp. 305–329, 1982.

[11] J.-C. F. Magali Bardet and B. Salvy, "On the complexity of the $f5$ gröbner basis algorithm," *Journal of Symbolic Computation*, vol. 70, pp. 49–70, 2015.

[12] H. Judd, "Virals, memes, and the lick's circulation through online jazz communities," *Twentieth-Century Music by Cambridge University Press*, vol. 19, no. 3, pp. 393–410, 2022.

[13] C. Stover, "Dig that lick (dtl): Analyzing large-scale data for melodic patterns in jazz performances," *Journal of the American Musicological Society*, vol. 74, no. 1, p. 195–214, 2021.