Formant Wave-Function Synthesis

An Investigation and MATLAB Implementation

Adam Garay

MUMT 501
December 2022
McGill University

Introduction

In his 1984 paper [1], Rodet introduces a new method for synthesizing speech sounds using formant wave-functions. This method, called FOF (Fonction d'Onde Formantique) synthesis, allows for the creation of highly realistic and natural-sounding speech by accurately modelling the vocal tract and its resonance frequencies, known as formants. Rodet presents the mathematical principles behind this synthesis technique and demonstrates its effectiveness through various examples and comparisons to other speech synthesis methods.

The formant wave-function is derived from a second-order filter, which describes the resonant frequencies of the vocal tract. The equation for this wave-function is given by:

$$h[n] = A[n] \cdot Ge^{-\alpha n} \cdot \sin(\omega_c n + \phi)$$

where $A[n]$ is the amplitude envelope, G is the gain, α controls the exponential damping, ω_c is the fundamental frequency, and φ is the initial phase. This equation describes how the formant wave-function varies over time, allowing for the creation of complex and dynamic speech sounds.

Rodet's paper on formant wave-function synthesis introduced a new and innovative method for synthesizing speech sounds. His work had a significant impact on the field of speech synthesis, leading to the development of new techniques and algorithms for creating natural-sounding speech. In particular, Rodet's CHANT implementation [2] demonstrated the effectiveness of formant wave-function synthesis in generating highly realistic and natural-sounding speech. This implementation was widely used and served as a foundation for further research in the field. Overall, Rodet's paper and CHANT implementation greatly advanced the state of the art in speech synthesis and paved the way for the development of more sophisticated and accurate speech synthesis systems.

Formants and the FOF Wave-Function

The human voice produces formants, or resonances, as a result of the vibrations of the vocal cords and the filtering effect of the vocal tract. The vocal cords produce a fundamental frequency, which is the base pitch of the sound, and this frequency is then modified by the shape and size of the vocal tract. The vocal tract acts as a resonator, with different parts of it resonating at different frequencies. These resonances, or formants, give the human voice its characteristic timbre and make it possible to distinguish one person's voice from another. The formants are produced by the filtering effect of the vocal tract on the sound waves produced by the vocal cords. As the sound waves pass through the vocal tract, they are selectively amplified or attenuated at different frequencies, depending on the shape and size of the tract. This creates the characteristic resonances, or formants, of the human voice.

Second-order filters are a type of filter that is commonly used to model these formants. When second-order filters are used in parallel, each filter is responsible for producing one of the formants in the synthetic sound. The filters are typically arranged in a way that each one is tuned to a different frequency, allowing them to produce the different resonances that make up the formants. The output of each filter is then combined together to produce the final synthetic sound. A common second-order filter used to produce formants has the following transfer function:

$$H(z) = \frac{b_0 + b_1 z^{-1}}{1 - a_1 z^{-1} - a_2 z^{-2}} = \frac{b_0 + b_1 z^{-1}}{(1 - pz^{-1})(1 - \overline{p}z^{-1}})$$

where b_0 and b_1 determine the center frequency and bandwidth, a_1 controls the total magnitude, and a_2 controls the slope of the envelope.

There are a number of disadvantages to using such a filter model. Firstly, it can be more computationally expensive than other methods of synthesis. This is because each filter in the model requires its own set of calculations, which can add up to a significant amount of processing power. Another disadvantage of this type of filter model is that it can be difficult to control and fine-tune the filters to produce the desired sound. Because each filter is responsible for producing a specific formant, the overall sound can be sensitive to the settings of each individual filter. This can make it challenging to achieve the desired timbre and quality of the synthetic sound.

In contrast, one advantage of FOF (formant plus fundamental frequency) synthesis is that it can be more computationally efficient than a filter model that uses second-order filters in parallel. This is because FOF synthesis combines the fundamental frequency with a set of predetermined formant frequencies, rather than using individual filters to produce each formant. This means that it requires fewer calculations and can be processed more quickly. Another advantage of FOF synthesis is that it can be easier to control and fine-tune the timbre and quality of the synthetic sound. This is because the formant frequencies are predetermined and do not need to be adjusted individually, as they would in a filter model.

A second-order filter as above has the following impulse response, which can be found by performing an inverse Discrete Fourier Transform:

$$h[n] = Ge^{-\alpha n} \cdot \sin(\omega_c n + \phi)$$

This is an exponentially damped sinusoid. However, to model more naturally occuring spectra, it is convenient to multiply this damped sinusoid by an envelope $A[n]$. Rodet uses the following envelope with an attack duration of π/β:

$$A[n] = \begin{cases} \frac{1}{2}(1-\cos(\beta n)) & 0 \le n \le \frac{\pi}{\beta} \\ 1 & n > \frac{\pi}{\beta} \end{cases}$$

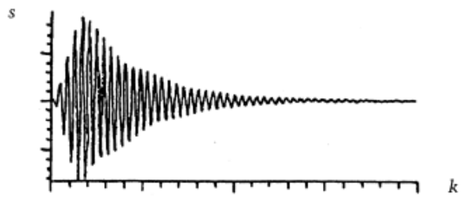
Thus the general form of a formant wave-function is:

$$h[n] = \begin{cases} 0 & n < 0 \\ \frac{1}{2}(1-\cos(\beta n))e^{-\alpha n}\sin(\omega_c n + \phi) & 0 \le n \le \frac{\pi}{\beta} \\ e^{-\alpha n}\sin(\omega_c n + \phi) & n > \frac{\pi}{\beta} \end{cases}$$

Each FOF can be uniquely defined by the following parameters:

- ω_c = the center frequency,
- $\alpha/\pi = BW$ = the bandwidth,
- A = the amplitude,
- π/β = the attack time, and
- ϕ = the initial phase

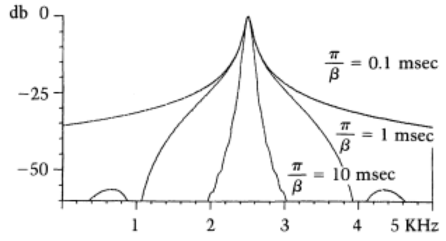
Fig. 7. A formant wave function (FOF).



When different FOFs are used in parallel to perform synthesis, each FOF is responsible for producing one of the formants in the synthetic sound. The FOFs are typically arranged in a way that each one is tuned to a different frequency, allowing them to produce the different resonances that make up the formants. The output of each FOF is then combined together to produce the final synthetic sound. This allows the FOFs to work together to create a more accurate and realistic imitation of human speech.

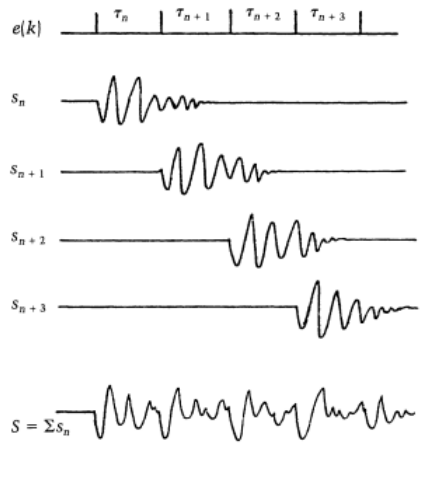
The attack parameter pi/β affects the frequency response. In particular, we can see β controls the skirt width but does not actually change the bandwidth: the bandwidth is controlled solely by α, not by β at all.

Fig. 4. Power spectrum of the FOF. A(k) sin($\omega_c \cdot$ k + ϕ) for different values of π/β, where ω_c = 2.5 KHz and α/π = 80 Hz.



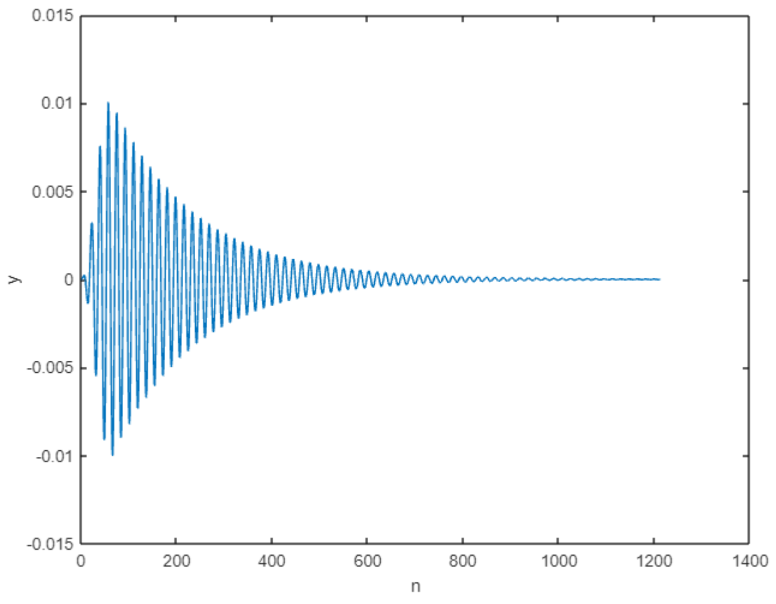
The fundamental frequency is the base pitch of the synthetic sound. This frequency is produced by applying the impulse response (the summed output of the FOFs) once every fundamental period. When an impulse is applied to the summed output of the FOFs, it causes a sudden spike in the amplitude of the sound. This spike occurs at the same frequency as the fundamental frequency, and it produces a periodic vibration in the output that is perceived as the fundamental frequency of the synthetic sound. This ensures that the fundamental frequency is produced at the correct pitch and with the desired level of accuracy, which allows for the creation of synthetic sounds that are more realistic and accurate.

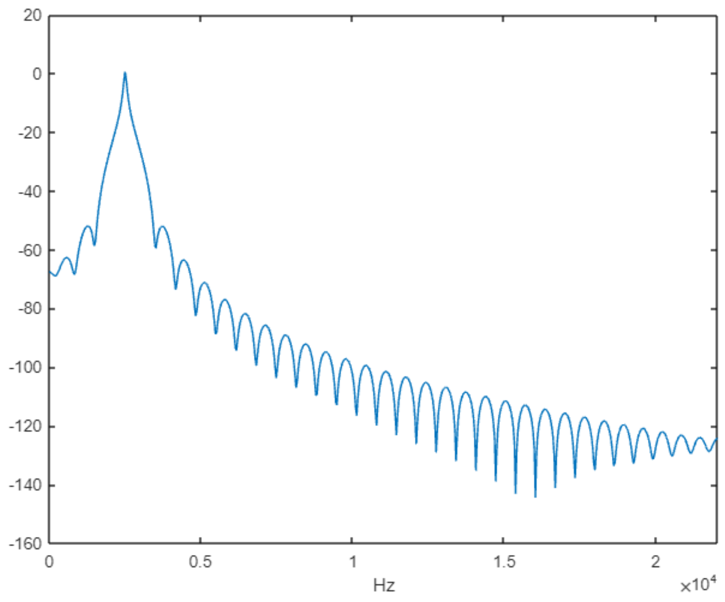
Fig. 6. Construction of the response S as a sum of the responses S_n.



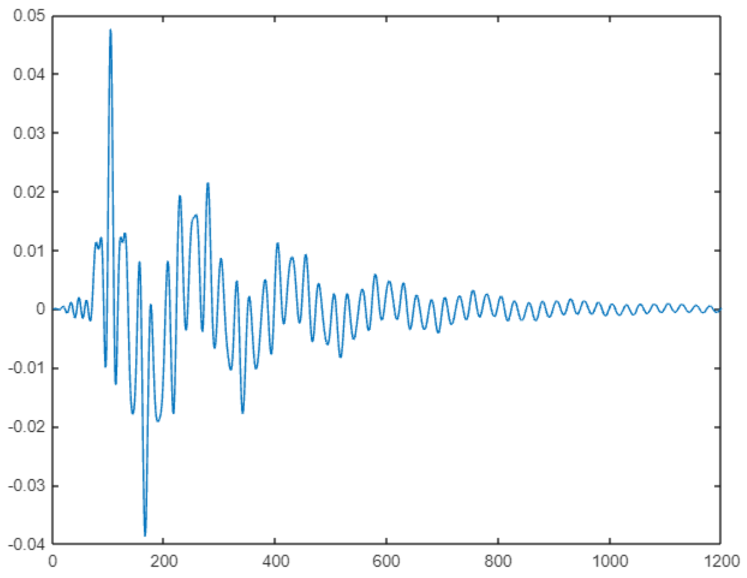
My Implementation

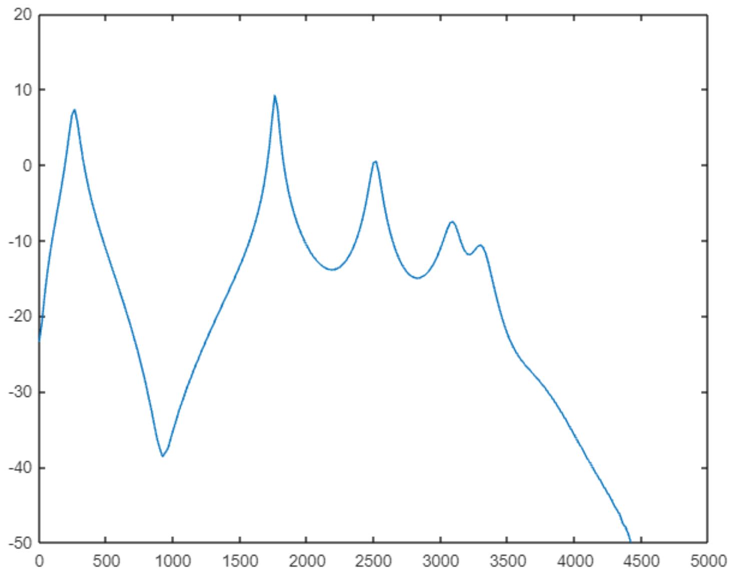
First, I wrote a function for generating a formant waveform that takes as input a specified frequency, bandwidth, amplitude, attack, and phase, as described above, as well as a sample rate. The function first calculates the omega, beta, and alpha values based on the input parameters. It then calculates the length of the formant using the decay time (T60) and the alpha value, and creates an array of samples for the formant using this length and the specified sample rate. The function then separates the samples into two sections: the attack portion, which is the initial rise of the waveform, and the rest of the waveform. The function then sets the values of the samples in each section according to the input parameters, and returns the resulting waveform as an array of samples.



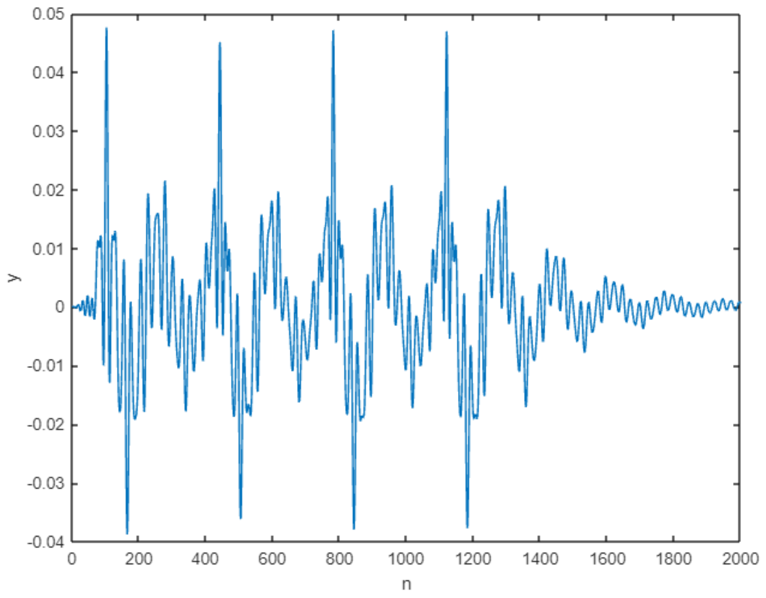


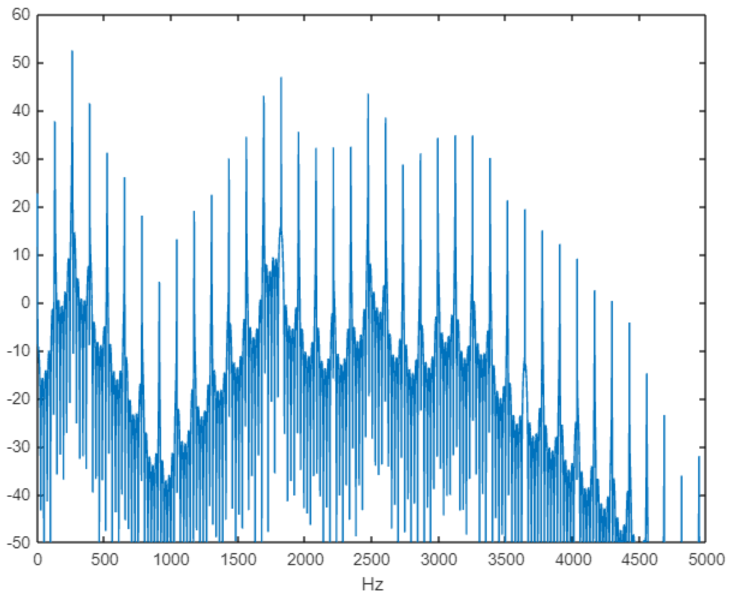
Then I wrote a function that takes as input a matrix of formant wave-functions, the fundamental frequency of the sound, the number of periods to synthesize, and the sampling frequency of the synthesized sound. The function first sums the formant wave-functions to create the overall waveform for the sound.





It then adds each formant sum when excited by impulses at the fundamental frequency.

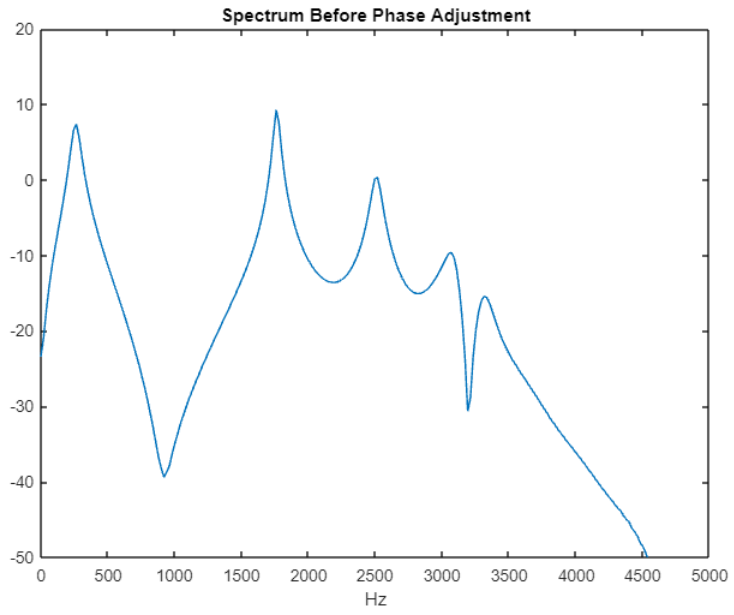


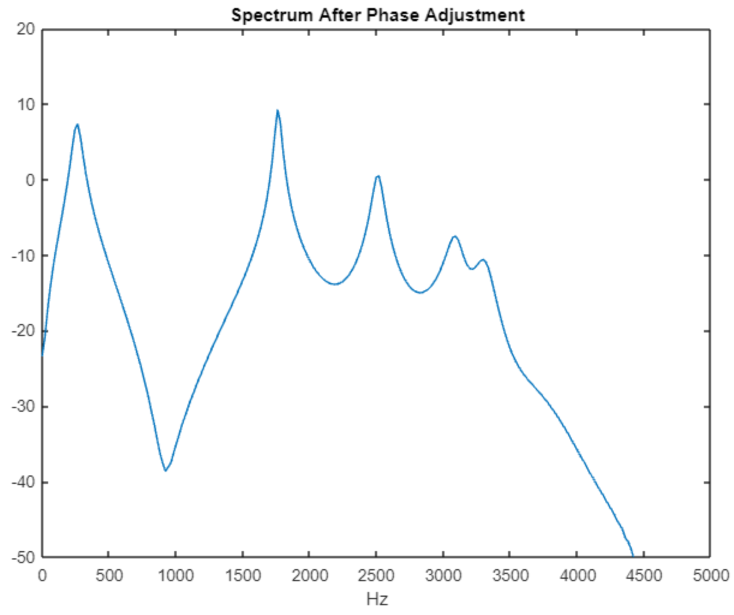


Synthesis is complete; the waveform can now be written to an audio file. I have synthesized ”a” and ”o” vowel sounds, provided in a.wav and o.wav, as well as a sound provided by parameters in [1], provided in Rodet.wav.

Implementation Issues

The biggest problem I encountered was with the phase. The problem is that the phase of each sinusoidal waveform can be different, which can cause the resulting composite waveform to have a distorted or unnatural sound. This is because the phase of a sinusoidal waveform determines its starting point in time, and if the phases of the individual waveforms are not aligned, they will not be in sync with each other when summed together. This can cause the resulting composite waveform to have a distorted sound, causing zeros as seen between formant center frequencies. To avoid this problem, it is important to ensure that the phases of the individual waveforms are properly aligned before summing them together to create a composite waveform. This can typically be done by setting the phase of each waveform to the same value before summing them together, and by aligning the maximums of each FOF envelope.





With my implementation of FOF synthesis, the synthesized sounds are perceived as robotic or unnatural. This happens for a number of reasons, primarily because the human voice does not produce a perfect fundamental frequency such as one that is input when summing FOFs; there is some variation which changes its timbre. As well, the envelope function does not perfectly mimic the natural variations in the amplitude and duration of formants in human speech. In addition, the use of noise signals as the source of the formants can also contribute to a robotic or unnatural sound if the noise is not carefully shaped to match the desired formant frequencies.

Another issue with formant wave-function synthesis is the difficulty of creating synthesized sounds with perfectly desired fundamental frequencies. This is because to simulate any frequency F0, we must divide the sampling rate by it and then round to approximate the desired frequency. The rounding operation can cause some values to be rounded down or up, potentially resulting in a loss of accuracy. This can cause some values to be excluded or unable to be represented, leading to an incomplete or inaccurate representation of the data. It is possible to fix this manually using interpolation, but this is not something that I implemented myself.

Further: A Full Synthesizer

I created my own synthesizer, implemented as a function in MATLAB. The function takes as input the sampling rate, and a matrix where each row represents a note and consists of three values: the vowel, the frequency, and the duration of the note. The function generates audio samples for each of the notes in the input notes matrix and concatenates them into a single waveform. The resulting waveform is then written to a specified file at the specified sampling rate fs. To estimate formant vowel parameters, I used values provided by CSOUND [3], which is a programming language for music and audio synthesis that was developed in the 80s and written in C. It includes a number of built-in functions for generating formant vowel sounds, and in the appendix of its manual it provides the frequencies, bandwidths, and amplitudes of the first four formants for each vowel sound, which I used as a starting point when creating formant vowel sounds. I have synthesized the vowel sounds of the first line of "Ave Maria", provided in avemaria.wav.

Using FOF synthesis, it is possible to create a full synthesizer with diverse sounds: all that is needed to synthesize any vocal or instrumental sound is a large set of formant parameters, which can be determined experimentally or with some other method. It is even possible to let the user control the parameters; this only causes minor changes in the synthesis and so could be implemented rather efficiently. Overall, FOF synthesis is capable of synthesizing complex sounds by using the formant parameters to create a wide range of timbres and shapes, and could be implemented as a convenient, simple and efficient synthesizer if a simple UI was built alongside it.

Conclusion

In conclusion, Rodet's FOF synthesis is a powerful and versatile technique for synthesizing speech and other sounds. It allows for the generation of complex and realistic waveforms by combining multiple formant frequencies and other parameters. The technique has been widely used in a variety of applications, including speech synthesis, music synthesis, and sound design. Overall, Rodet's FOF synthesis has proven to be a valuable tool for creating a wide range of sounds, and it has been an important area of research and development in the field of audio synthesis. While my implementation has many flaws and much room for improvement, it demonstrates the capability of FOF synthesis to create vowel sounds and synthesize simple musical phrases.

Bibliography

[1] X. Rodet, "Time-Domain Formant-Wave-Function Synthesis," *Computer Music Journal*, vol. 8, no. 3, 1984.

[2] X. Rodet, Y. Potard, and J.-B. Barriere, "The CHANT Project: From the Synthesis of the Singing Voice to Synthesis in General," *Computer Music Journal*, vol. 8, no. 3, 1984.

[3] "Appendix D. Formant Values," *www.csounds.com*. [Online] Available: http://www.csounds.com/manual/html/MiscFormants.html [Accessed Dec. 14, 2022].

[4] J. Neri, P. Depalle, "REDS: A new asymmetric atom for sparse audio decomposition and sound synthesis", *International Conference on Digital Audio Effects (DAFx17)*, Edinburgh, Scotland, pp. 268-275, September 2017.

[5] C. d'Alessandro and X. Rodet, "Synthèse et analyse-synthèse par fonctions d'ondes formantiques," *Journal d'Acoustique*, vol. 2, 1989.

[6] "Formant Synthesis Models," *ccrma.stanford.edu*. [Online] Available: https://ccrma.stanford.edu/ jos/pasp/Formant_Synthesis_Models.html [Accessed Dec. 14, 2022].