A FUNCTION SIMULATOR AS A TOOL FOR EDUCATING DIGITAL SOUND BASICS

David Weenink

University of Amsterdam, the Netherlands
david.weenink@uva.nl

ABSTRACT

We describe a computer application that offers students and teachers, especially those with no real mathematical background, the possibility to learn the relation between, for example, elementary sounds and their mathematical formulation as sine functions with frequency, amplitude and phase parameters. The sampling of analog signals and the relation between the sampling frequency and a faithful representation of the sound can easily be demonstrated. Aliasing can be made audible and visible in several ways.

Keywords: interactive phonetic education, pure tone, spectrum, sine, sampling

1. INTRODUCTION

In teaching students of linguistics the elements of speech signal processing, a little bit of mathematics is unavoidable. The sine and cosine functions are indispensable here. However, at the same time, mathematical functions like sines and cosines are not the bread and butter of students of linguistics. Many of these students don't feel comfortable when they see formulas. Nevertheless, if we want to talk about pure tones, for example, it is difficult to avoid sine functions. The function occurs over and over again.

In order to help students to get a grip on these functions and to have them experience the relation between a formula and the corresponding sound we have developed an application that may help these students. The application is based on the demo facilities in the freely available, widely known, general purpose speech processing program Praat [1]. The application is a single separate script that can simply be run by the user and will be made freely available. The application offers, among others, a display of the parametrized function of time; direct manipulation of the parameters of the function as well as playing the function as a sound; a spectrum of the function and visualization of the function as a sampled waveform.

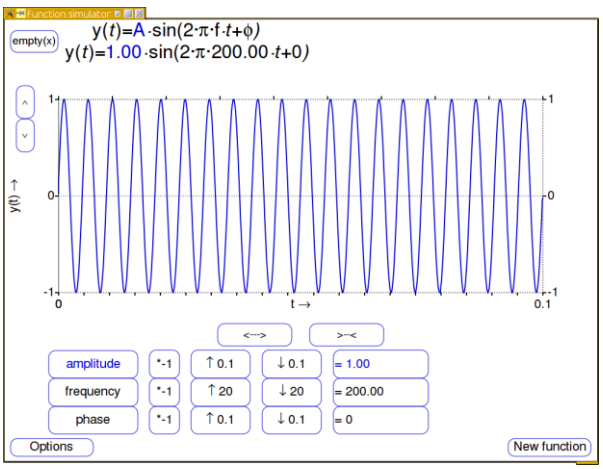
2. THE FUNCTION SIMULATOR

The function simulator is an application to help students master the relation between elementary functions like sines and cosines and the sounds that can be generated according to these functions. Because the application is explorative by nature it is difficult to describe with words only. Therefore we will describe how a particular session might develop.

2.1. Basic simulator layout and functionality

As an example let us start with a function of time that models a pure tone: y(t)=A sin(2 ft+ φ). If we start the function simulator with this function the following window appears:

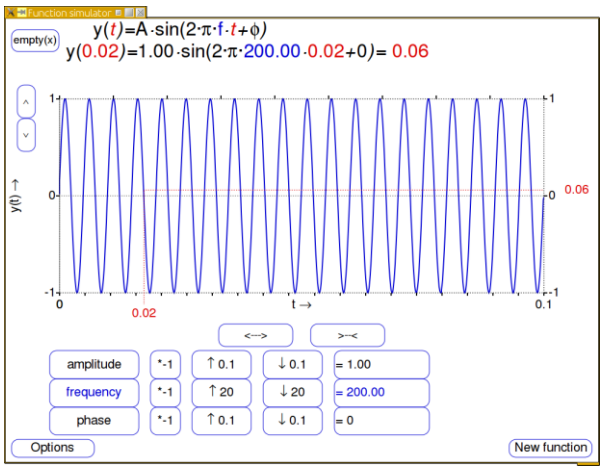
Figure 1: The function simulator for a pure tone.



The top of the window shows the function in a *symbolic* notation as it is was given above and as it is written in many text books. This function has three parameters: amplitude (A), frequency (f) and phase (φ). At the line below the symbolic representation, the *actual* representation appears where the current numerical values of the parameters are shown. The current parameters in Fig. 1 are A=1, f=200 and φ=0. The middle part of the window is dominated by the graph of the function on the time interval from 0 to 0.1 s. It

shows exactly twenty periods of a sine with an amplitude of 1. The two boxes on the left of the graph increase or decrease the vertical scale by a factor of 2 if clicked. The two boxes just below the graph expand or shrink the time domain of the graph. Given this graph you might want to explain how this graph was constructed.

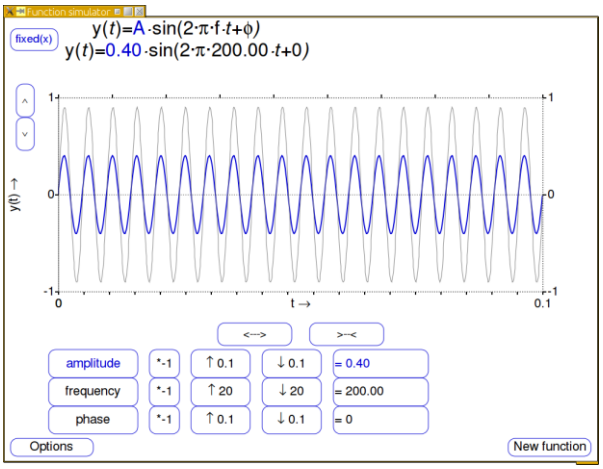
Figure 2: How the function graph is calculated.



By clicking with the mouse pointer near the time axis, at the bottom of the graph, a number of things happen. Fig. 2 gives an impression. First of all, the corresponding time at the mouse position is displayed in red colour below the time axis. A red dotted line is drawn vertically from this time point to the corresponding function value in the graph and from the function value a horizontal dotted line is drawn to the right-hand side of the graph where the function value is displayed at the right of its endpoint. Next, in the symbolic display of the function the time variable is marked with a red colour too, and, most importantly, the value of the time variable is substituted in the actual representation of the function and its outcome is displayed as well. As Fig. 2 shows, we clicked at the axis near time t=0.02. By clicking in succession on a number of different time values, and seeing the actual calculation enrolled, helps in demystifying what is going on behind the scenes.

At the bottom of the graph the three rows of boxes show buttons that enable you to change the parameters. The parameter name is in the left box of the row. Clicking with the mouse pointer on it merely makes this parameter box active, and all occurrences of this parameter in the window. In Fig. 2 the frequency box was activated and it shows a deep blue colour, as do the frequency parameter symbol and its numerical value in the symbolic and actual representations of the function at the top of the window. The next three boxes in the same row allow changing the parameter by clicking on them and the result of the change is displayed in the last box. Of course, the display of the function immediately adapts to the new situation. The first of the three change boxes inverts the parameter while the second and third increase or decrease the parameter with the number indicated on the box. For example, given the default values of the parameters in Fig. 1, clicking once on the amplitude step down by 0.1 box will decrease the amplitude to the value of 0.90. The last box in this row will now show =0.90, the actual representation will show the value 0.90 for the amplitude parameter and the graph's amplitude has shrunken too. Of course the step sizes and the multiplier can be changed at will.

Figure 3: Changing a parameter and compare.



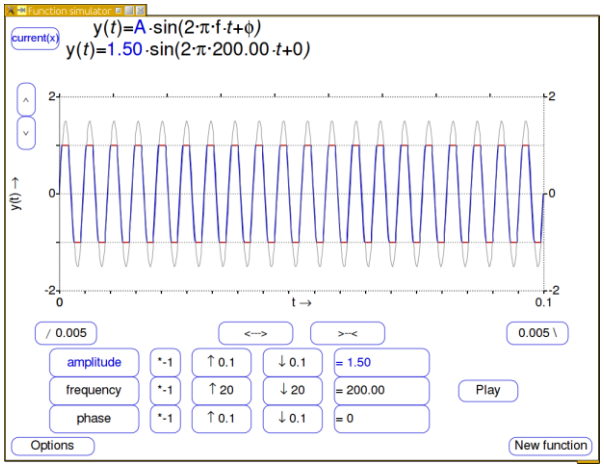
Many times you only want to see what happens if you change a parameter. However, sometimes you want to display a change with respect to a previous configuration of the parameters. This is possible and is visualized in Fig. 3 where an actual function with A=0.4 is displayed in blue and a previous version with A=0.9 is displayed in grey colour in the background. The situation of Fig. 3 was achieved by first making the amplitude 0.9, then clicking on the box at the top left which was labeled empty(x) but after the click toggles to fixed(x). The parameter values will be remembered and the corresponding curve will be displayed in grey. Changing a parameter will now display the actual parameter curve in blue.

2.2. Listening to the tone functions

In the previous section we were only concerned with the appearance of the sine function. Now we want to relate the appearance to actual sound. In

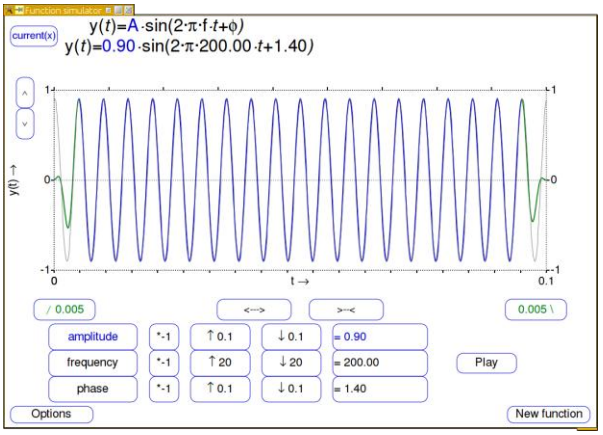
other words we want to make sounds whose amplitudes change according to the formula described above. These sounds are called pure tones. This can be arranged by clicking on the options box at the bottom left and from the form that pops up you choose the **Playable as Sound** option. Three new buttons will appear in the window as Fig. 4 shows.

Figure 4: Playing a clipped function. The representation of the sound is shown in the background with grey colour while the realized audible sound is shown by the blue curve.



First of all, the button labeled Play at the right of the parameter buttons, next a fade-in button at the bottom left of the graph labeled "/ 0.005" and a fade-out button labeled "0.005 \" at the bottom-right of the graph. The number on the fade-in button indicates a 5 ms fade-in time. We are now able to demonstrate:

- how the sound that is associated with the function sounds. You will hear that the tone is higher when you increase its frequency. You will hear that the sound becomes softer when you decrease the amplitude. You will hear that the sound becomes louder when you increase the amplitude. You will hear and see that increasing the amplitude above a certain level will distort the sound. This phenomenon is called *clipping*. In Fig. 4 we show an example of clipping. The sound drawn in grey is the sound as specified by the parameter values while the sound drawn in blue is the sound they will actually hear. The maximum sound amplitude equals one and this also happens to be the maximum amplitude for sounds to play without distortion in the Praat program. If the sound amplitude is larger than one, the digital sound representation cannot be faithfully reproduced as audio sound and distortion happens. This is indicated by the red lines at the top and bottom parts of the sound curve and they show the parts that have been clipped.
- You will hear that if you change the phase and the sounds don't start at zero amplitudes anymore, a click will be audible too and that in order to avoid this click you need to apply *fade-in*. The same applies at the end of the sound where you will need a fade-out to correct for abrupt amplitude changes. You will hear that if fade-in and fade-out are applied, the phase of the sound is not important anymore. In Fig. 5 we show the sound function with fade-in and fade-out applied. It will play without audible distortion while the sound function without the fade-in and fade-out as shown with the grey lines at the start and end parts will play with two very disturbing click sounds. To present visual feedback, the sound parts where the fade was applied will have a green colour in the graph and the buttons will also show the same green colour if active.
- **Figure 5:** Fade-in and fade-out.



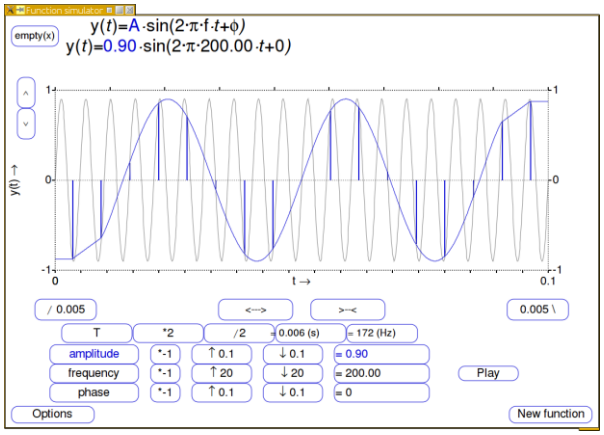
2.3. The digital representation of a sound

In the previous sections the representation of the sound functions were smooth curves. To show that the underlying representation of a sound in the computer is an array of numbers we need to explain where these numbers come from, how many of these numbers we need and how precise these numbers need to be. In other words we need to explain the sampling of an analog signal.

In Fig. 6 we show sampling. The option to show the sampling of a sound makes several things happen. First of all, instead of showing the sound function as a curve we now show the *sample points* as the endpoint of the vertical lines going from

zero to the sample value. The lines are positioned in the middle of a sampling period as each sample value is the average value of the analog signal during that small period of time. The sampling period T is the inverse of the sampling frequency. The first sample therefore is at time T/2, the second at time T/2+T, etc. The sample values therefore "represent" the numbers an Analog to Digital Converter would deliver. The blue curve in Fig. 6 shows the signal as reconstructed from the sample values. As can be seen, the reconstruction at the start and the end of the sound is not as well as in the middle because in these regions we have less samples available for interpolation. The reconstruction can be interpreted as the sound that we would hear if the samples were send to a Digital to Analog Converter, low-pass filtered and made audible by a loudspeaker. In Fig. 6 the reconstructed signal shows approximately two and a half periods in the 0.1 s duration interval. The original sound, shown in the background with grey colour is according to the specifications a 200 Hz signal. The figure thus clearly shows an example of *aliasing*. A phenomenon that occurs if the sampling frequency and the sound's bandwidth are not in concordance. As you might have noticed in Fig. 6, a new row of buttons has appeared, above the three rows we already had. These buttons change the sampling period by factors of two. The default sampling period is 1/44100 and to produce Fig. 6 we have increased this number by several factors of two, resulting in a sampling period of approximately 0.006 s, corresponding to a sampling frequency of 172 Hz. By changing the sampling period students can investigate the relation between sampling frequency and signal frequency and experience where the faithful representation of a sound breaks down.

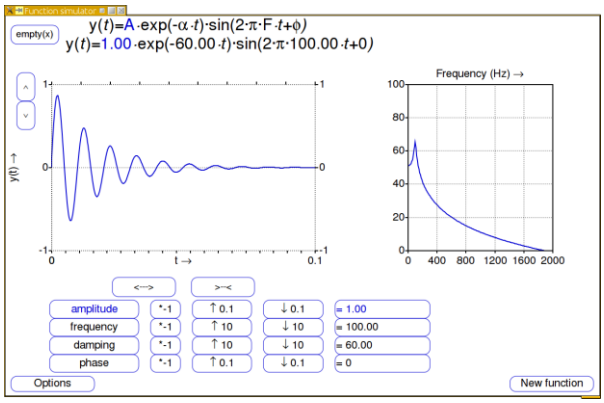
Figure 6: Sampling and aliasing.



2.4. And more...

Besides these simple sines, all numerical functions in Praat are in principle available for simulation. As an example we present Fig. 7. It shows the formant function as a damped sine function on the left and its spectrum on the right. We have not shown spectra in previous figures because they would clutter the figures too much. Needless to say that changing a parameter setting also may change the spectrum. By manipulating the damping parameter in the formant function the increase of the bandwidth in the spectrum visualizes the relation between damping and formant bandwidth.

Figure 7: A formant function and its spectrum.



3. CONCLUSION

We have shown an application to let students interactively get familiar with a number of aspects in the digital representation of sounds.

4. REFERENCES

[1] Boersma, P., Weenink, D. 2011. Praat: Doing phonetics by computer [Computer program]. Version 5.2.24, retrieved 11 May 2011 from *http://www.praat.org/*.