Addsynt, an additive voice synthesiser for PC
Granqvist, S.

journal: TMH-QPSR
volume: 37
number: 4
year: 1996
pages: 057-060

http://www.speech.kth.se/qpsr
Addsynt, an additive voice synthesiser for PC

Svante Granqvist

Abstract

A voice synthesiser program based on adding sinusoids was written. Additive synthesis gives good flexibility in changing the voice source spectral characteristics at the cost of slower execution. The program allows some 20 voice source parameters to be programmed over time, such as spectral tilt, individual harmonic’s levels, subharmonic content in different frequency ranges, the phase of the harmonics, F0, flutter, vibrato and noise. The voice source is filtered by up to 32 formants.

The different parameters’ values are given either graphically by mouse clicks or by typing the numbers. In each parameter window one can put any background picture, such as a spectrogram, from the clipboard or a bitmap file, in order to simplify tracing of real voices.

Finally, some features have been added to aid the construction of batch series of stimuli.

Description

Addsynt is a program aiming at frequency-domain synthesis of voices and voice qualities. It focuses on the perceptual appearance of the voice rather than modelling the vocal folds. Since the ear is mainly sensitive to the spectral characteristics of the voice, frequency-domain synthesis can be considered a straightforward method to do this. This approach gives good flexibility when testing different acoustical parameters’ perceptual value. One field of interest is to synthesise roughness by means of subharmonics. In contrast to time-domain synthesis (Gauffin et al., 1995) (i.e. jitter and shimmer), additive synthesis makes it possible to specify spectral characteristics exactly as desired.

The method is not, however, intended to replace the more common methods of synthesis in the time-domain, such as the LF model by Fant et al. (1985). These methods might still be more relevant in order to understand the physics of the vocal folds or when real-time synthesis is needed.

One benefit with frequency-domain synthesis is that it is easier to make the signal band limited to half the sampling frequency, f_s/2. This is not the case for time-domain synthesis. Typically, a time-domain synthesised glottal pulse will have frequencies above f_s/2 and these will be aliased to a frequency below f_s/2. If the period time of the waveform is not an integral number times sampling frequency, the aliased harmonics will be located between the harmonics which could lead to an uncontrolled contribution of roughness in the voice. These aliased harmonics can easily be heard if one makes a frequency sweep. If F0 increases one can hear tones that has a decreasing frequency. The problem is, however, reduced by the fact that the source and vocal tract has a falling spectrum, but it is a problem worth being aware of when dealing with roughness in voices. With frequency-domain synthesis one simply does not include any sinusoids with frequencies higher than f_s/2. For Addsynt, this statement is not entirely true though, since Addsynt has some time-domain parameters. For example, changing the level of the signal very rapidly would lead to similar problems.

The frequency-domain synthesis has some other consequences. One of them is obviously that time-domain parameters are replaced by frequency-domain parameters. Glottal closing rate has its correspondence in spectral tilt. Jitter and shimmer are implemented as subharmonics.

Addsynt has a graphical interface towards the user and has some 20 programmable parameters. In order to simplify re-synthesis of real voices, each parameter window can hold a background picture for tracing. The picture can be copied from the clipboard or from a Windows BMP or WMF file. The background picture is movable within the parameter window in order to synchronise it with the axes (Fig. 1).

Addsynt has a feature for building batches of stimuli. Each point in a parameter window can vary between some given boundaries in a selectable number of steps. Points can vary synchronously, separately or in combinations.

Addsynt is written in Borland Delphi and the object-oriented structure makes it easy to add more parameters as new ideas come along.
Detailed description of parameters

Addsynt is a program under continuos development and this section describes the current version as of this date. More features will most probably be added in the future (Fig. 2).

Level parameters

The Addsynt voice source has three parts, harmonics, subharmonics and noise. The harmonic part consists of \(F_0\) and its overtones \(2 \cdot F_0, 3 \cdot F_0\) and so on. The subharmonic part consist of frequencies \(0.5 \cdot F_0, 1.5 \cdot F_0, 2.5 \cdot F_0\) and so on. The noise part is a sequence of square-distributed random numbers with white spectrum. This is the only part of the voice source that is generated in the time-domain.

Overall level

Acts as a volume control for the entire signal. The overall level is programmable over time.

Harmonic level

Acts as a volume control for the harmonic and subharmonic part of the signal. The harmonic level is programmable over time.

Relative subharmonic level

Sets the level of subharmonics relative to the harmonics. Each subharmonic’s level follows the level of the adjacent harmonic with higher frequency. The relative subharmonic level is programmable over time.

Noise level

Acts as a volume control for the noise generator. The noise level is programmable over time.

Fundamental frequency parameters

The Addsynt fundamental frequency is set by the \(F_0\), vibrato and flutter parameters. The \(F_0\) parameter sets the average \(F_0\). The vibrato parameters allows for a sinusoidal variation of \(F_0\). The flutter parameter is derived from a square-distributed white noise generator connected to a low-pass filter with adjustable cut-off frequency and Q value. The flutter parameter can also be used to generate wow by selecting a sufficiently low LP frequency, see Ternström & Friberg (1989). These three components are added to the actual \(F_0\) frequency.

\(F_0\)

Sets the average fundamental frequency. \(F_0\) is programmable over time.

Vibrato frequency

Sets the frequency of periodic \(F_0\) variation. The vibrato frequency is programmable over time.

Vibrato amplitude

Sets the strength of the vibrato in % of average \(F_0\). The vibrato amplitude is programmable over time.

Flutter amplitude

Sets the strength of the flutter. Since the flutter is generated by passing a noise signal through a second order LP filter, the filter characteristics will also effect the flutter strength, a higher LP cut-off frequency or Q will give more flutter. Flutter amplitude is programmable over time.

Flutter LP frequency

Sets the flutter LP filter cut-off frequency. A higher cut-off frequency gives faster fluctuations in \(F_0\). The flutter LP frequency is programmable over time.
Flutter LP Q
Sets the flutter LP filter Q value. A higher Q gives a more periodic fluctuation in F0. High Q values give F0 fluctuations similar to vibrato. The flutter LP Q value is programmable over time.

Spectral shape parameters
The level of each harmonic/subharmonic in the voice source is determined by the spectral tilt, harmonic/subharmonic spectrum and frequency response parameters. The voice source is followed by a conventional formant filter with up to 32 formants.

Spectral tilt
Sets the tilt of the voice source spectrum in dB/octave. The spectral tilt is programmable over time.

Harmonic and subharmonic spectrum
Sets the level of individual harmonics/subharmonics. The current version does not support programming of the harmonic spectrum over time.

Figure 2. Block schematic of Addsynt. Each shaded box corresponds to a user entered parameter.
**Frequency response**
Similar to harmonic spectrum, but the level set is fixed to a certain frequency rather than a partial number. The current version does not support programming of the frequency response over time.

**Formants**
Sets formant frequencies of the formant filters. The formants are programmable over time.

**Formant bandwidth**
Sets the formant bandwidth as a function of formant frequency. The current version does not support direct programming of the formant bandwidth over time.

**Phase parameters**
The phase of each harmonic/subharmonic is set by the phase parameters. The phase of each harmonic/subharmonic is set by adding the phase and phase randomness parameters.

**Harmonics and subharmonics phase**
Sets a fixed phase for each harmonic/subharmonic to add to the argument in each sinusoid generator. This phase does not vary over time. The current version does not support programming of the harmonics phase over time.

**Harmonics and subharmonic phase randomness**
Before every sample file build, each sinusoid generator is assigned a random number between 0 and $2\pi$. A fraction of these numbers can be added to the argument of each generator. If the fraction does not vary over time, the number added will be constant over the build. The fraction is set by the harmonics phase randomness parameter and is programmable over time.

**File format**
Adsynt saves the parameter lists in its own .ADS format which is much like the Windows ini-file style. The ADS files are readable with any common text editor. The sound files produced by Adsynt are in the .SMP file format, which is used by other programs, such as the SoundSwell (Soundswell Signal Workstation, AB Nyvalla DSP) and Aladdin packages from AB Nyvalla DSP.

**System requirements**
Adsynt will run on a 386 PC with 4MB RAM and Microsoft Windows 3.1, but a faster PC is strongly recommended. The minimum recommended system configuration is a 100MHz Pentium with 8MB RAM and 200MB hard disk. Adsynt runs under Microsoft Windows 3.1 and Microsoft Windows 95.

**Acknowledgements**
This work was supported by research grants from the Bank of Sweden Tercentenary Foundation. I would also like to thank Jan Gauffin, Britta Hammarberg and Stellan Hertegård for their valuable input.

**References**