SMS Software Manual

Music Technology Group, Audiovisual Institute
Pompeu Fabra University
[version of June 2000]

This document is a manual for version 2.0 of the SMS software. It includes indications on how to get a good analysis and a listing of the low level analysis and synthesis parameters. It should work for the different front-ends available for the software: Windows95/98/NT, Web and Unix.

1. How to get a good SMS analysis

The concept of a "good" analysis is not a simple one. If the only consideration was to reconstruct the original sound as accurate as possible, the analysis could be quite simple. However, SMS has been designed with the goal of being able to recover the perceptual characteristics of the original sound but at the same time obtaining a sound representation that would give the user flexibility to transform as many characteristics of the sound as possible.

The analysis/synthesis approach of this software is based on modeling sounds as sinusoids plus a residual component, analyzing sounds with this model and generating new sounds from the analyzed data. The analysis procedure detects sinusoids by studying the time-varying spectral characteristics, which are then subtracted from the original sound and the remaining "residual" is modeled as a time-varying spectra, or its approximation.

For the analysis to be most successful the sounds should be monophonic (single melodies), should have been recorded in a dry (non reverberant) environment, using the maximum dynamic range possible, with a high sampling rate (22,050 or 44,1000) and 16 bits resolution. Polyphonic sounds can also be analyzed but the resulting analysis might not be very useful as data to be transformed in the synthesis process. Reverberation and background sounds generate energy that is neither part of the partials nor the stochastic component and might create problems in separating the two components.

For a sound which does not follow the ideal Sinusoidal plus Residual model there are two possibilities, analyze it as a Residual-only signal with no approximation, or as a Sinusoidal Inharmonic with no Residual. With these two representations several transformations will be possible. For example, analyzing two sounds as Residual-only components the Hybridization is quite successful and we can obtain the traditional morphing, or cross-synthesis, effect.

The analysis and synthesis are frame based. They both use a constant frame size specified by the user. Specified as *FrameRate* in the analysis and as *FrameSize* in the synthesis. Thus a frame size of 128 samples in the synthesis and using a sampling-rate of 44,100 it corresponds to a frame rate of 345 frames per second. Analysis and synthesis frame rates do not have to correspond, the synthesis will do appropriate frame interpolation.

1.1 Recommended steps to be taken

1. Listen to the sound and get a feeling for it.

2. Look its time-varying spectra. Look for characteristics such as: its harmonic or not, pitch range, stability of the sound, number of partials, ...

3. Set the analysis parameters for the sinusoidal component and analyze only this component.

4. Look at the sinusoidal representation obtained from the analysis and check for obvious analysis errors such as: it has not found partials that should have been found or it has found very unstable sinusoids. Redo de analysis if necessary.

5. Synthesize the sinusoidal component and listen to it. Compare it with the original sound and make sure that it includes all the partials of the sound and that the sound quality is good. Redo the analysis if necessary.

6. Once decided that the analysis of the sinusoidal component is good enough, redo the analysis, now with the residual component.

7. Listen to the synthesis of both Sinusoidal and Residual components together. Listen to the Residual by itself, specially the not approximated one, and make sure that it does not include any partials. If the residual does not sound good enough redo the analysis by changing the parameters of the residual approximation. Increase the number of coefficients and if still does not sound good, set the residual analysis with no approximation.

8. Keep changing the parameters until you are satisfied with the synthesized sound. Once satisfied try reducing the number of sinusoids and the number of coefficients used in the residual approximation without loosing any sound quality. This will give a more compact representation easier to transform.

9. Try several drastic transformations and make sure there are no distortions produced. If that is the case you may want to try another analysis.

10. Have fun with it!.

1.2 How to get a good sinusoidal component

The Sinusoidal analysis can be tuned by the user in such a way that we can either track only stable partials of the sound or the whole spectrum can be modeled with sinusoids. We will get the maximum flexibility by tracking only the stable partials and leave as a residual component the rest of the sound.

Before analyzing a sound, we should check whether the sound is harmonic or not, and then choose either the *Harmonic* or *Inharmonic* analysis. Pseudo-harmonic sounds (like piano tones) should also be considered as harmonic. While any sound can be analyzed as *Inharmonic*, and the resynthesis without any transformation can be quite good, the number of possible transformations is more limited.

When the sound has attacks that are noisy and sharp, the analysis with the default parameters may have problems. Typically the analysis will not find the sinusoids at the very beginning of the attack. The feature of *AttachReanalysis* will try to solve the problem by reanalyzing the attacks once it has found the stable part of the sound. This is like doing the analysis backwards and thus, when the algorithm arrives at the attack, it is already tracking the main partials and can reject non-relevant spectral components appropriately, or at least evaluate them with some acquired knowledge.

The analysis window determines the time-frequency compromise. In stable sounds we should use long windows (several periods) and very smooth ones (for example, Blackman-Harris 92dB). However most sounds will have both stable portions and sharp transitions, thus a compromise will have to be taken between time and frequency resolution. When the analysis is done as *Harmonic* the actual size and type of the window, *WindowSizeHarm* and *WindowTypeHarm*, can be set to change depending on the fundamental frequency detected in every frame. With this feature we can obtain the best time-frequency compromise. In the case of inharmonic sounds there is no period, pitch, and the window size, *WindowSizeInh*, in samples, should be long enough to identify all the partials. For fast changing sounds we might have to use a Hamming window with the smallest possible window size.

With the *Magnitude Threshold* we can reject any spectral peaks, the candidates for partials, that fall below a given threshold. Since in every sound there is some noise floor we should set the magnitude threshold to be the magnitude of this noise floor.

A typical cause of a bad sinusoidal analysis in harmonic sounds is a wrong fundamental frequency detection. There are a lot of user parameters that can help the pitch detection algorithm. The easiest parameters to control are the *LowestPitch*, *HighestPitch* and *DefaultPitch*, with these we can restrict the range of the search algorithm.

1.3 How to get a good residual component

The residual is obtained by subtracting the detected sinusoids from the original sound. If the sinusoidal analysis has been done correctly the residual should be free of partials and it would only contain the stochastic part of the sound. If the sound was not well recorded, or it is distorted in some way, the residual might contain energy which does not correspond to the noise component of the sound and it would be difficult to model it as an stochastic signal. If no sinusoidal analysis has been performed everything is left as residual.

The residual component can either be approximated or not. The process of approximation implies that the residual is as stochastic signal. Without any approximation the residual is left as it is, representing it as a waveform or in the frequency domain as a STFT.

When approximated, the magnitude spectrum of each frame of the residual sound is approximated by a linear interpolation using a given number of coefficients, *ResCoef*. The more coefficients we use the better the modeling of the spectral characteristics will be. The maximum number of coefficients is 129, which is the size of the magnitude spectrum.

2. Analysis Parameters

InputSoundFile

type: String

default: 1

Name of the sound file to be analyzed. It supports wave, snd, aiff, and au files. Sampling rates of 22050 and 44100. Resolution of 16bits.

OutputSmsFile

type: String

default: 1

Name of the SMS file to be created. The convention is to use the extension .sms.

FrameRate

type: Number

default: 172.266

Number of analysis frames per second. This will determine the hop size of the analysis window. This value is maintained constant for the whole sound but the actual window size (in samples) can change in harmonic sounds. The default value corresponds to 128 samples at 22050).

BeginPos

type: Number

default: 0

Offset in seconds from the beginning of the input sound file InputSoundFile, where the analysis will start. 0 means beginning of the file, 1 means the end of the file. See also: EndPos.

EndPos

type: Number

default: 1

Position in seconds in the input sound file where the analysis will end. See also BeginPos.

SilenceZeroCrossing

type: Function

range: 0 to 1

default: 0.15

Minimum amount of zero-crossing of a given frame for it to be considered silence if the energy is below SilenceEnergy. By detecting silence the analysis avoids having to perform any sinusoidal analysis.

SilenceEnergy

type: Function

range: 0 to 1

default: 0.008

Maximum normalized energy of a frame for it to be considered silence if the number of zero-crossings is above SilenceZeroCrossing. It is a value from 0 to 1. If the energy factor is less than SilenceEnergy/4 the frame will be considered as silence independently of the SilenceZeroCrossing.

SineModel

type: Function

default: 1

Type of sinusoidal analysis.

0 No analysis

1 Harmonic analysis

2 Inharmonic analysis

Many analysis parameters will depend on whether the sinusoidal analysis is of type harmonic or inharmonic. The main one is that pitch detection is only performed when the sinusoidal model is a harmonic one. Being a function we can analyze different parts of a sound in different ways.

WinType

type: Function

default: 8

Type of analysis window to use in the sinusoidal analysis.

0 Hamming

1 KaiserBessel17

2 KaiserBessel18

3 KaiserBessel19

4 KaiserBessel20

5 KaiserBessel25

6 KaiserBessel30

7 KaiserBessel35

8 Blackman-Harris 62 dB

9 Blackman-Harris 70 dB

10 Blackman-Harris 74 dB

11 Blackman-Harris 92 dB

These windows cover the range from a not very smooth window, Hamming, to a very smooth one, Blackman-Harris 92 dB. Being a Function we can change the type of window in time.

WindowsInFFT

type: Number

default: 4

Size of the FFT buffer as number of windows. This paramenter control the zero padding.

HighestFreq

type: Function

range: 0 to 22050

default: 11025

Highest frequency in Hz to be searched for in the peak detection process.

MagThreshold

type: Function

range: -300 to 0

default: -100

Minimum spectral magnitude used in the peak detection process, expressed as a dB value. No partials softer than this value will be found.

nSines

type: Function

range: 0 to 400

default: 60

Maximum number of sinusoids to be used in sinusoidal analysis. In the case of harmonic analysis this will mean the number of harmonics to be searched for.

KeepSinePhases

type: Boolean

range: 0 to 1

default: 1

Specifies whether or not the phases of the original sound are kept in the sinusoidal analysis data. Depending on the context it may or may not be a significant factor in additive synthesis. In certain steady states sounds a rearrangement of phases may not be audible. Phase relationships become apparent in the perception of brilliant but short-lived attacks, grains, and transients. It is also very noticeable in low tones and in complex sounds where the phases of certain components are shifting over time. For high-fidelity reproduction of these transients and quasi-steady-state tones, phase data help reassemble short-lived and changing components in their proper order and are therefore valuable. When kept the quality of the synthesized sound will be higher but some transformations will not sound as good or will not be possible. If set to 0 the phases will not be kept.

MinLengthSines

type: Function

range: 0 to 10

default: 0

Minimum length of the sinusoids in seconds. Sinusoids shorter than this value will be deleted.

MaxDropOut

type: Function

range: 0 to 10

default: 0

Maximum length in seconds of gaps in sinusoids that will be filled by interpolating their boundaries.

PartialTrackMode

type: Function

default: 1

Specifies how to create trajectories between the peaks found from frame to frame, when performing the peak continuation.

0	Peak with nearest frequency
1	Peak with nearest frequency and magnitude
2	Peak with nearest frequency and maximum magnitude

PeakContribution

type: Function

range: 0 to 1

default: 0.3

Contribution of the previous peak values of a given sinusoidal trajectory to the current “guide” values. If the value is 1, it means that the previous peak will completely define the current guide value. If the value is 0, the previous peak will not be used at all.

ResolveGuide

type: Boolean

range: 0 to 1

default: 1

Specifies whether or not to find another peak for a guide that lost it due to a conflict with another guide, when performing the peak continuation.

WindowTypeHarm

type: Function

range: 0 to 11

default: 2

Type of window used. Like the WindowSizeHarm parameter, it is defined as a function of frequency in order to change WindowTypeHarm depending on pitch.

WindowSizeHarm

type: Function

range: 0.5 to 20

default: 2.5

Size of analysis window expressed as number of pitch periods. The actual window size in seconds is this value divided by the pitch found at every given moment. It is defined as an envelope function of frequency in order to change *WindowSizeHarm* depending on fundamental frequency. X values go from 0 to 1 and y values represent the window size as a multiple of the pitch period. It covers the range from 0 to 1000Hz.

PitchDetection

type: Boolean

range: 0 to 1

default: 1

Specifies whether or not to perform pitch detection. When no pitch detection is done (0), the DefaultPitch is used as the reference pitch throughout.

PitchMode

type: Number

default: 1

Which method to use for the pitch detection

0 time domain

1 frequency domain

2 time domain controls frequency domain

LowestPitch

type: Function

range: 0 to 22050

default: 70

Lowest fundamental frequency in Hz to be searched for. No partials candidates to pitch are looked for below this value.

DefaultPitch

type: Function

range: 0 to 22050

default: 110

Default fundamental frequency in Hz. This is the frequency that is used to set the actual analysis window size when no fundamental has been found by the program, for example the first few frames. In normal situations it is convenient to give the value of the fundamental frequency of the beginning of the sound so that the program can start with a good guess. This value has to be between the LowestPitch and the HighestPitch.

HighestPitch

type: Function

range: 0 to 22050

default: 600

Highest fundamental frequency in Hz to be searched for.

MaxPitchError

type: Function

range: 0 to 1000

default: 20

Maximum error to accept a fundamental in a frame. The pitch detection algorithm uses this error as a way to measure the goodness of each fundamental candidate. If it is set very large it will find a pitch at every frame, that is, the higher the value the more tolerant the program is to accept a fundamental.

PitchContribution

type: Function

range: 0 to 1

default: 0.7

Contribution of the fundamental frequency of the current frame to the current guide frequency.

HarmonicCorrection

type: Function

range: 0 to 1

default: 0

This parameter might be useful for harmonic series that are not ideal, with a value of 0 the guides are built as multiples of the fundamental, i.e., perfect harmonic series. With a value higher than that it will correct the guide frequencies according to the lower harmonics found. This might help in stretch harmonic series such as the ones of a piano sound.

HarmonicCorrectionSize

type: Function

range: 1 to 20

default: 1

Number of partials that are used to define the HarmonicCorrection factor.

MaxFreqEnergyForHarmonic

type: Function

range: 0 to 22050

default: 3000

When the frequency of the peak with maximum amplitude is higher than MaxFreqEnergyForHarmonic, the pitch will be set to 0. This is especially useful for voice analysis: if the energy is mainly concentrated at high frequencies we can assume it is an unvoiced region of the sound, so pitch is set to zero.

MaxHarmonicAmplitudeOscillation

type: Function

range: 0 to 100

default: 25

Related to the MaxHarmonicAmplitudeVariation parameter. The pitch algorithm will discard the pitch candidates when the magnitude in dB of the peaks that try to describe their harmonic series oscillates as average more than MaxHarmonicAmplitudeOscillation.

MaxHarmonicAmplitudeVariation

type: Function

range: 0 to 100

default: 40

When looking for candidates for the pitch, the program will discard those candidates with very irregular harmonic peaks. With this parameter, when two of the first few consecutive harmonics have magnitudes in dB that differ more than MaxHarmonicAmplitudeVariation, the pitch candidate will be discarded. Useful to avoid detecting a pitch of an octave lower than the real pitch.

KeepSineFreqDeviations

type: Boolean

range: 0 to 1

default: 1

Whether or not to keep the sinusoid frequency deviations from the perfect harmonic series when the Pitch Attribute has been extracted. Thus when set to 0 only the fundamental frequency will be stored, during synthesis it will generate a perfect harmonic series.

UseHarmonicHighLowEnergyRatio

type: Function

range: 0 to 1

default: 1

WindowSizeInh

type: Function

range: 64 to 8191

default: 701

Size of analysis window expressed as number of samples when sinusoidal models has been set to inharmonic. There is no pitch information and the window size can only be set in samples. However we should find the size that is sufficient to discriminate the partials of the sound.

LowestFreq

type: Function

range: 0 to 22050

default: 20

Lowest frequency in Hz of the peaks to be detected. No partials lower than this frequency will be found.

InhFreqDeviation

type: Function

range: 0 to 100000

default: 50

Like HarmonicFreqDeviation, but for the inharmonic sinusoids.

InhFreqDeviationSlope

type: Function

range: -1 to 1

default: 0.01

Like HarmonicFreqDeviationSlope, but for the inharmonic sinusoids.

ResModel

type: Function

range: 0 to 3

default: 2

Type of model to me used for residual analysis

- 0 No analysis
- 1 Approximation
- 2 No approximation, stored as STFT
- 3 No approximation, stored as a waveform

When set to approximation, the residual sound is approximated as a stochastic signal by fitting an envelope to the magnitude spectrum of each frame and discarding the phase spectra. When set to no approximation the residual is left as it is and represented as a short-time spectrum or as a waveform.

ResCoef

type: Function

range: 0 to 129

default: 129

Number of coefficients for the stochastic representation. This number corresponds to the number of breakpoints in the magnitude spectral envelope.

ResSubstractMode

type: Function

default: 1

Specifies how to obtain the residual

- 0 Nothing
- 1 Subtract the sinusoids from the input sound
- 2 Subtract all peaks from the input sound

ResWindowSize

type: Number

default: 2

Window-size in frames of the residual window.

ResAmpCorrec

type: Boolean

default: 1

Specifies whether to activate (1) the amplitude correction that is performed in the residual to make sure that it does not have a bigger amplitude than the original sound, or to deactivate it (0).

DiskCache

type: Boolean

default: 1

Specifies whether to load the sound file InputSoundFile into memory before analysis (value 1), or to read each window frame from disk as it analyses (value 0).

CompressSines

type: Boolean

default: 0

Specifies whether or not to compresses amplitude, frequency deviations, phases and spectral shape of sinusoids to 8 bits.

CompressSpec

type: Boolean

default: 0

Specifies whether or not to compress residual data to 8 bits

DbPeakInterpolation

type: Boolean

default: 0

Specifies whether or not to use dB magnitudes (1) to find the top of the spectral peaks by interpolating the highest three spectral bins, or linear magnitude (0).

DbSineInterpolation

type: Boolean

default: 0

Specifies whether to use of dB magnitudes (1) to obtain the instantaneous magnitude when synthesizing sinusoids, or linear magnitude (0). This is for the sinusoidal synthesis that happens during the analysis.

ParabolicSineInterpolation

type: Boolean

default: 1

Specifies whether to use parabolic interpolation (1) in sinusoidal synthesis to calculate the instantaneous frequency and phase, or linear interpolation (0).

AttackReanalysis

type: Boolean

default: 0

Specifies whether to reanalyze an attack once it has stabilized. See also: See also: nAttackReanalysisFrames, AttackReanalysisLowPitchMargin and AttackReanalysisHighPitchMargin.

nAttackReanalysisFrames

type: Number

default: 10

Number of frames for the reanalysis of attacks. When reanalyzing attacks, every attack is analyzed backwards in time to obtain a more accurate result, because pitch will have been estimated better. See also: AttackReanalysis.

AttackReanalysisLowPitchMargin

type: Number

default: 0.95

Lowest fundamental frequency-ratio (in respect to the analyzed pitch) to be searched for when doing the attack reanalysis. lowest pitch = pitch * AttackReanalysisLowPitchMargin) See also nAttackReanalysisFrames, AttackReanalysis, AttackReanalysisHighPitchMargin.

AttackReanalysisHighPitchMargin

type: Number

default: 1.05

Highest fundamental frequency-ratio (in respect to the analyzed pitch) to be searched for when doing the attack reanalysis. highest pitch = pitch * AttackReanalysisHighPitchMargin See also nAttackReanalysisFrames, AttackReanalysis, AttackReanalysisLowPitchMargin.

VibratoExtract

type: Boolean

default: 0

Specifies whether or not to extract the vibrato from the pitch. See also: Synthesis parameter *VibratoWeight*.

3. Synthesis Parameters

InputSmsFile

type: String

default: 1

Name of the SMS file to be used for the synthesis.

OutputSoundFile

type: String

default: 1

Name of the sound file when saving the synthesized sound. It supports .wav, .snd, .au

SamplingRate

type: Number

default: 22050

Sampling rate of the synthesized sound. For best results it should be set to 44100 Hz.

BeginSelectionTime

type: Number

default: 0

Beginning time in seconds of the section to be used from the SMS analysis file (InputSmsFile).

EndSelectionTime

type: Number

default: 1000

Ending time in seconds of the section to be used from the SMS analysis file (InputSmsFile). Waves have to be shorter than 1000 s.

Mix

type: Boolean

default: 0

Specifies whether or not to mix the current event with the previous one, using the same output sound file (SynthesisOutput).

BeginEventTime

type: Number

default: 0

Offset from the beginning of the output sound file in seconds where the current synthesized event will be placed. This is useful when several sounds are synthesized with a score file and we want to put each one in different time positions.

WindowSize

type: Number

default: 2

Size of the synthesis window in number of frames.

FrameSize

type: Number

default: 128

The default value corresponds to an analysis frame rate of 172 at 22,050Hz or 345 at 44,100Hz.

Overlap

type: Number

default: 50

Overlap factor of synthesis windows as a percentage. By default the synthesis frame-rate is half of the *WindowSize*.

DiskCache

type: Boolean

default: 1

Whether to load the SMS file *InputSmsFile* on memory before synthesis (value 1) or to read each frame from disk as it synthesizes (value 0). Important to set to 1 for real time synthesis.

SynthesisOutput

type: Number

default: 1

Output of the synthesis. This value is derived by taking or adding up any of the following bits:

1 Save as a sound file

2 Write directly to the DAC output buffer

4 Save as an SMS file

OutputSmsFile

type: SMSFile

default: 1

Name of the SMS file when saving the synthesized sound as an SMS file. See also *SynthesisOutput*.

Type

type: Function

default: 7

Which components to be used for synthesis.

1 only sinusoidal

2 only spectral residual component

3 sinusoidal plus spectral residual

4 only wave residual component

5 sinusoidal plus wave residual

6 wave plus spectral residual

7 sinusoidal plus wave residual plus spectral residual

Amp

type: Function

range: 0 to 5

default: 1

Scaling of the overall sound amplitude. A value of 1 does not modify the original amplitude, a value of 2 generates a sound with twice the amplitude of the original sound.

AmpSine

type: Function

range: 0 to 5

default: 1

Amplitude scaling to be applied to the sinusoids of the sound. This scaling is multiplied by *Amp*.

AmpSineWeightBypass

type: Boolean

range: 0 to 1

default: 1

Specifies whether or not to bypass the frequency dependent change in amplitude of sinusoids.

AmpSineWeight

type: Function

range: -10 to 10

default: 0

Interpolation between *AmpSineWeightShape1* and *AmpSineWeightShape2*

AmpSineWeightShape1

type: Function

range: 0 to 5

default: 1

Determines how much each sinusoid is affected by the amplitude transformation given by *AmpSine*. A Y value of 1 means that the sinusoids affected by it are completely modified, and a value of 0.5 for the same sinusoids means that the sinusoids are only affected by 50% of the change. This parameter is used to filter the sinusoidal component of the sound and its effect is quite clear when *AmpSine* is set to 1.

AmpSineWeightShape2

type: Function

range: 0 to 5

default: 0

Behaves like *AmpSineWeightShape1*.

AmpSineOdd

type: Function

range: 0 to 5

default: 1

Amplitude scaling applied to the odd sinusoids. The final amplitude change of the odd sinusoids is product of *Amp*, *AmpSine*, *AmpSineWeight*, by AmpSineOdd. This is especially useful for harmonic sounds, when a sinusoid corresponds to a harmonic partial.

AmpSineEven

type: Function

range: 0 to 5

default: 1

Amplitude scaling applied to the even sinusoids. The final amplitude change of the even sinusoids is product of *Amp*, *AmpSine*, AmpSinesWeight and AmpSineEven. This is especially useful for harmonic sounds, when a sinusoid corresponds to a harmonic partial.

AmpSpec

type: Function

range: 0 to 5

default: 1

Amplitude scaling applied to residual component.

AmpSpecWeightBypass

type: Boolean

default: 1

Whether or no to use the AmpSpecWeigth.

0 Use AmpSpecWeight

1 Don't use AmpSpecWeight

AmpSpecWeight

type: Function

range: -10 to 10

default: 0

Interpolation between *AmpSpecWeightShape1* and *AmpSpecWeightShape1*.

AmpSpecWeightShape1

type: Function

range: 0 to 5

default: 1

Determines how much each residual coefficient is affected by the amplitude transformation given by *AmpSpec*. The function covers all the coefficients independently of the range specified by the X coordinate. A Y value of 1 means that the coefficient affected by it are completely modified, and a value of 0.5 for the same coefficients means that the coefficients are only affected by 50% of the change. This parameter is used to filter the residual component of the sound and its effect is quite clear when *AmpSpec* is set to 1.

AmpSpecWeightShape2

type: Function

range: 0 to 5

default: 0

Like *AmpSpecWeightShape1*. See *AmpSpecWeight*.

AmpSineList

type: List

default: 1

List of changes in amplitude of each individual sinusoid. Useful for modifying a few partials.

FreqSine

type: Function

range: 0 to 5

default: 1

The frequency scaling to be applied to the sinusoids of the sound.

SameSpectralEnv

type: Boolean

default: 0

Whether or not to maintain the same spectral shape when frequency is changed.

FreqSineWeight

type: Function

range: 0 to 5

default: 1

Determines how much each sinusoid is affected by the frequency transformation given by *FreqSine*. This is an frequency scaling factor applied on top of the one given by *FreqSine*. A Y value of 1 means that the sinusoids affected by it are completely modified, and a value of 0.5 for the same sinusoid means that the sinusoids are only affected by 50% of the change.

FreqSineOdd

type: Function

range: 0 to 5

default: 1

Frequency scaling applied to the odd sinusoids. The final frequency change of the odd sinusoids is the product of *FreqSine*, *FreqSineWeight* and FreqSineOdd. This is especially useful for harmonic sounds, when a sinusoid corresponds to a harmonic partial.

FreqSineEven

type: Function

range: 0 to 5

default: 1

Frequency scaling applied to the even sinusoids. The final frequency change of the even sinusoids is the product of *FreqSine*, *FreqSineWeight* and FreqSineEven. This is especially useful for harmonic sounds, when a sinusoid corresponds to a harmonic partial.

FreqSineStretch

type: Function

range: -0.99 to 5

default: 0

Modification of the distribution of the sinusoids of the original sound by stretching or compressing them. A value of 1 leaves the sinusoids in its original place, a value of 2 stretches progressively all the sinusoids in such a way that the lowest sinusoid remains in its own place, but as we going up the sinusoids as being transposed higher and higher. The last sinusoid gets transposed an octave higher than its original frequency. This is especially useful for harmonic sounds, when a sinusoid corresponds to a harmonic partial.

FreqSineShift

type: Function

range: 0 to 10000

default: 0

Linear shift in Hz applied to all sinusoids. This adds a constant frequency value to all the sinusoids, either positive or negative value. If applied to a harmonic sound the result will be inharmonic.

FreqSineList

type: List

default: 1

List of changes in frequency of each individual sinusoid. Useful for modifying a few partials.

AmpSineModAmp

type: Function

range: 0 to 100

default: 0

Amplitude of the modulation as a percentage applied to the amplitude of the sinusoids.

AmpSineModFreq

type: Function

range: 0 to 100

default: 0

Frequency in Hz of the modulation applied to the amplitude of the sinusoids.

FreqSineModAmp

type: Function

range: 0 to 100

default: 0

Amplitude of the modulation as a percentage applied to the frequency of the sinusoids.

FreqSineModFreq

type: Function

range: 0 to 100

default: 0

Frequency in Hz of the modulation applied to the frequency of the sinusoids.

SpecModAmp

type: Function

range: 0 to 100

default: 0

Amplitude of the modulation as a percentage applied to the residual amplitude.

SpecModFreq

type: Function

range: 0 to 100

default: 0

Frequency in Hz of the modulation applied to the residual amplitude.

InputSmsHybridFile

type: SMSFile

default: 1

Name of the SMS file to use to hybridize with the original SMS file.

Hybridize

type: Boolean

default: 0

Specifies whether or not to perform hybridization.

HybBeginSelectionTime

type: Number

default: 0

Beginning time in seconds of the section to be used from the SMS hybridization file (InputSmsHybridFile).

HybEndSelectionTime

type: Number

default: 1000

End time in seconds of the section to be used from the SMS hybridization file (InputSmsHybridFile).

HybridizeEnv

type: Boolean

range: 0 to 1

default: 1

Specifies whether or not the hybridize , as an envelope through time. See also: *Hybridize*.

HybFrameAttTemporalIntp

type: Boolean

range: 0 to 1

default: 1

Specifies whether or not to perform temporal interpolation of the hybrid frame attributes. A value of 0 will make the synthesis faster.

HybSynchronizeTime

type: Function

range: 0 to 1

default: 0

Specifies point of synchronicity in normalized time (from 0 to 1) between InputHybridSmsFile and *InputSmsFile*. The syntax of HybSynchronizeTime function is x0 y0 x1 y1 x2 y2

HybSineAmp

type: Function

range: 0 to 1

default: 0

Interpolation factor between the sinusoids magnitude of the *InputSmsFile* (0) and of the *InputSmsHybridFile* (1).

HybSineSpectralShape

type: Function

range: 0 to 1

default: 1

Interpolation factor between the residual spectra of the *InputSmsFile* (0) and the *InputSmsHybridFile* (1). When *HybSineShapeWeight1* and *HybSineShapeWeight2* are used, this is an interpolating function between the two.

HybSineShapeWeight1

type: Function

range: 0 to 1

default: 0

Before applying the sinusoids spectral shape of the *InputSmsHybridFile* on the sinusoid spectral shape of the sound, these two spectra are normalized and compressed by applying this compression-envelope as a way to control the relative contribution of each one in the final output as a function of frequency. This envelope can have any value between 0 and 1. A value of 1 compresses the corresponding spectrum magnitude point of the original sound and does not modify the spectral value of *InputSmsHybridFile*, therefore the only contributing spectral value is the one of *InputSmsHybridFile*. A value of 0 produces the opposite, and a value of 0.5 leaves the two spectral magnitude values as they are.

HybSineShapeWeight2

type: Function

range: 0 to 1

default: 1

Behaves like *HybSineShapeWeight1*. We can interpolate between the two of them by using *HybSineSpectralShape*.

HybSinePitch

type: Function

range: 0 to 1

default: 1

Interpolation factor between the pitch of the *InputSmsFile* (0) and the *InputSmsHybridFile* (1).

HybSineFreq

type: Function

range: 0 to 1

default: 1

Interpolation factor between the frequency deviations of sinusoids of the *InputSmsFile* (0) and the *InputSmsHybridFile* (1).

HybSpecAmp

type: Function

range: 0 to 1

default: 0

Interpolation factor between the residual magnitude of the *InputSmsFile* (0) and the *InputSmsHybridFile* (1).

HybSpecSpectralShape

type: Function

range: 0 to 1

default: 1

Interpolation factor between the residual spectra of the *InputSmsFile* (0) and the *InputSmsHybridFile* (1). When *HybSpecShapeWeight1* and *HybSpecShapeWeight2* are used, this is an interpolating function between the two.

HybSpecShapeWeight1

type: Function

range: 0 to 1

default: 0

Before applying the magnitude spectra of the *InputSmsHybridFile* on the residual spectra of the sound, these two spectra are normalized and compressed by applying this compression-envelope as a way to control the relative contribution of each one in the final output as a function of frequency. This envelope can have any value between 0 and 1. A value of 1 compresses the corresponding spectrum magnitude point of the original sound and does not modify the spectral value of *InputSmsHybridFile*, therefore the only contributing spectral value is the one of *InputSmsHybridFile*. A value of 0 produces the opposite, and a value of 0.5 leaves the two spectral magnitude values as they are.

HybSpecShapeWeight2

type: Function

range: 0 to 1

default: 1

Behaves like *HybSpecShapeWeight1*. We can interpolate between the two of them by using *HybSpecSpectralShape*.

HybSpecPhase

type: Function

range: 0 to 1

default: 1

Interpolation factor between the phase of the residual spectrum of the *InputSmsFile* (0) and the *InputSmsHybridFile* (1).

HybWaveAmp

type: Function

range: 0 to 1

default: 0

Interpolation factor between the magnitude of the residual spectrum of the *InputSmsFile* (0) and the *InputSmsHybridFile* (1).

HybWaveform

type: Function

range: 0 to 1

default: 1

Interpolation factor between the magnitude of the residual waveform of the *InputSmsFile* (0) and the *InputSmsHybridFile* (1).

AttSineUsePitch

type: Boolean

range: 0 to 1

default: 0

Specifies whether to use *AttSinePitch* or not.

AttSinePitch

type: Function

range: 0 to 20000

default: 0

Frequency in Hz for pitch attribute. This is the pitch function we want the synthesized sound to have. When the value is 0 it uses the original pitch or the corresponding interpolated value. *AttSinePitchTransposition* and *AttSinePitchShift* are applied on top of this. Useful for harmonizing.

AttSinePitchShift

type: Function

range: -20000 to 20000

default: 0

Frequency in Hz to be added to the pitch attribute of the sinusoids.

AttSinePitchTransposition

type: Function

range: 0 to 100

default: 1

Frequency scaling to be applied to the pitch attribute of the sinusoids.

AttSinePerfectHarmonic

type: Boolean

range: 0 to 1

default: 0

Specifies whether or not use a perfect harmonic series as partials. A 0 means use the original frequencies and 1 means to use a perfect harmonic series.

AttUseSineAmp

type: Boolean

range: 0 to 1

default: 0

Specifies whether or not to use *AttSineAmp*.

AttSineAmp

type: Function

range: -200 to 0

default: -30

AttSineAmpChange

type: Function

range: -200 to 200

default: 0

AttUseSpecAmp

type: Boolean

range: 0 to 1

default: 0

Specifies whether to use *AttSpecAmp* or not.

AttSpecAmp

type: Function

range: -200 to 0

default: -30

AttSpecAmpChange

type: Function

range: -200 to 200

default: 0

AttWaveAmp

type: Function

range: 0 to 5

default: 1

AttSineShapeShift

type: Function

range: -20000 to 20000

default: 0

Frequency in Hz to be added to the sinusoids spectral shape attribute. Frequency in Hz to shift the spectral shape of the sinusoids

AttSineShapeShiftWithPitch

type: Function

range: -20000 to 20000

default: 0

This value is multiplied by the frequency of the pitch, and is used to shift the spectral shape of the sinusoids.

Enhance

type: Function

range: 0 to 400

default: 0

Number of harmonics to generate. If the value is higher than the number of harmonics analyzed it will artificially generate new harmonics according with EnhanceSlope and EnhanceUpGain.

EnhanceSlope

type: Function

range: -2000 to 2000

default: 0

Slope of a straight-line function of frequency (dB/Hz) to be applied over the harmonics generated by *Enhance*. This is the "a" in a line written in the form y = a * x + b, with its origin in the last point of the spectral shape. See also: *EnhanceUpGain*.

EnhanceUpGain

type: Function

range: -200 to 200

default: 0

Offset of a straight line to apply over the harmonics generated by *Enhance* . It is the "b" in a line written in the form y = a * x + b with its origin in the last point of the spectral shape. See also: *EnhanceSlope*.

ResCombFilter

type: Boolean

range: 0 to 1

default: 0

Specifies whether or not to apply a comb filter to the residual using the pitch of the current frame as delay. (With this parameter set, the residual merges better with harmonics in the morphed sound)

ResCombFilterDelayCont

type: Function

range: 0 to 1

default: 0.5

Amount of delayed signal with respect to the original signal, as a value from 0 to 1. Maximum effect will be a value of 0.5. This is only used when the residual is a waveform, when it is stored in the frequency domain there is no control. (no need to touch it)

ResWaveFIR

type: Boolean

range: 0 to 1

default: 0

Specifies whether or not to apply a low-pass filter on the residual when stored as a waveform. It is useful to smooth glitches that may happen in v/u transition.

PhaseAlign

type: Boolean

range: 0 to 1

default: 1

Specifies whether phase alignment should be used when synthesizing.

PhaseAlignUseAnalysis

type: Boolean

range: 0 to 1

default: 1

When *PhaseAlign* is set, this specifies whether the phase alignment that is used when synthesizing, is the one resulting from the analysis, or a synthetic one.

PhaseAlignRandomDeviation

type: Function

range: 0 to 6.28319

default: 0.785398

When *PhaseAlignUseAnalysis* is not set, we use synthetic phase alignment. This parameter specifies how much randomness will be used for the synthetic phase alignment

PhaseAlignOnlyPeriodBegin

type: Boolean

range: 0 to 1

default: 1

This parameter specifies that only the beginning of the periods should be phase aligned.

PhaseAlignFrequencyPhaseChange

type: Function

range: 0 to 22050

default: 2000

It is possible to specify a different phase alignment until and from a given frequency PhaseAlignFrequencyPhaseChange. All sinusoids with a frequency lower than PhaseAlignFrequencyPhaseChange will be aligned according to *PhaseAlignLeft*, all sinusoids with a frequency higher than PhaseAlignFrequencyPhaseChange will be aligned according to *PhaseAlignRight*.

Harmonizer1

type: Boolean

range: 0 to 1

default: 0

When set, adds a pitch shifted version to the sound.

Harmonizer1Amp

type: Function

range: 0 to 100

default: 1

Amplitude of the pitched shifted sound.

Harmonizer1PitchTransposition

type: Function

range: 0 to 100

default: 1

Transposition of the pitched shifted sound.

Harmonizer1PhaseAlign

type: Boolean

range: 0 to 1

default: 1

When set, the pitched shifted sound will be synthesized with phase alignment. See also *PhaseAlign*.

Harmonizer1PhaseAlignUseAnalysis

type: Boolean

range: 0 to 1

default: 1

When *PhaseAlign* is set, this specifies whether the phase alignment that is used when synthesizing, is the one resulting from the analysis, or a synthetic one. See also *PhaseAlignUseAnalysis*.

Harmonizer2

type: Boolean

range: 0 to 1

default: 0

See *Harmonizer1*

Harmonizer2Amp

type: Function

range: 0 to 100

default: 1

See *Harmonizer1*

Harmonizer2PitchTransposition

type: Function

range: 0 to 100

default: 1

See *Harmonizer1*

Harmonizer2PhaseAlign

type: Boolean

range: 0 to 1

default: 1

See *Harmonizer1*

Harmonizer2PhaseAlignUseAnalysis

type: Function

range: 0 to 1

default: 1

See *Harmonizer1*

Harmonizer3

type: Boolean

range: 0 to 1

default: 0

See *Harmonizer1*

Harmonizer3Amp

type: Function

range: 0 to 100

default: 1

See *Harmonizer1*

Harmonizer3PitchTransposition

type: Function

range: 0 to 100

default: 1

See *Harmonizer1*

Harmonizer3PhaseAlign

type: Boolean

range: 0 to 1

default: 1

See *Harmonizer1*

Harmonizer3PhaseAlignUseAnalysis

type: Function

range: 0 to 1

default: 1

See *Harmonizer1*

TimeStretch

type: Function

range: 1e-04 to 1000

default: 1

Time stretch factor applied to the SMS data. A value of 0.5 shortens the sound to half its duration, and a value of 2 stretches the duration to twice its original value. This modification does not affect the sound pitch. See also: *NoUnvoicedTimeStretching*.

NoUnvoicedTimeStretching

type: Number

default: 0

Specifies to leave the unvoiced sounds (consonants) intact, and only apply the *TimeStretch* on the vowel-like sounds.

QuantizeTimeToFrame

type: Boolean

range: 0 to 1

default: 1

Specifies whether or not to quantize the time to the frame times.

DbSineShapeFilter

type: Function

range: -100 to 0

default: 0

Filter to be applied to the spectral shape of the sinusoids attribute.

DbSpecShapeFilter

type: Function

range: -100 to 0

default: 0

Filter to be applied to the spectral shape of the residual attribute.

VibratoWeight

type: Function

range: 0 to 10

default: 1

Interpolation between the pitch with vibrato removed (0) and the original pitch with vibrato (1). By using a value 1, an exaggerated vibrato. See also Analysis parameter VibratoExtract.

4. Examples

Example of a text file used in an analysis of a sound:

```
InputSoundFile cello.snd
OutputSmsFile cello.sms
SineModel 1
nSines 80
PitchDetection 1
ResModel 4
AttackReanalysis 1
LowestPitch 200
HighestPitch 300
DefaultPitch 260
```

Example of a text file used in the synthesis of several sounds:

```
InputSmsFile cello.sms
OutputSoundFile cello-syn.snd
SynthesisOutput 1
Type 7
FrameSize 128
SamplingRate 22050

BeginEventTime 4.5
InputSmsFile cello.sms
OutputSoundFile cello-syn.snd
Mix 1
SynthesisOutput 1
Type 7
FrameSize 128
SamplingRate 22050
AmpSpec 0
AmpSineList 1 1 2 0 3 0 100 0

BeginEventTime 9.0
InputSmsFile cello.sms
OutputSoundFile cello-syn.snd
Mix 1
SynthesisOutput 1
Type 7
FrameSize 128
SamplingRate 22050
AmpSpec 0
AmpSineList 1 0 2 1 3 0 100 0
```