Chapter 2

Sound modeling: signal based approaches

Lecture notes 2005

2.1 Introduction

The sound produced by acoustic musical instruments is caused by the physical vibration of a certain resonating structure. This vibration can be described by signals that correspond to the time-evolution of the acoustic pressure associated to it. The fact that the sound can be characterized by a set of signals suggests quite naturally that some computing equipment could be successfully employed for generating sounds, for either the imitation of acoustic instruments or the creation of new sounds with novel timbral properties.

A wide variety of sound synthesis algorithms is currently available either commercially or in the literature. Each one of them exhibits some peculiar characteristics that could make it preferable to others, depending on goals and needs. Technological progress has made enormous steps forward in the past few years as far as the computational power that can be made available at low cost is concerned. At the same time, sound synthesis methods have become more and more computationally efficient and the user interface has become friendlier and friendlier. As a consequence, musicians can nowadays access a wide collection of synthesis techniques (all available at low cost in their full functionality), and concentrate on their timbral properties.

Each sound synthesis algorithm can be thought of as a digital model for the sound itself. Though this observation may seem quite obvious, its meaning for sound synthesis is not so straightforward. As a matter of fact, modeling sounds is much more than just generating them, as a digital model can be used for representing and generating a whole class of sounds, depending on the choice of control parameters. The idea of associating a class of sounds to a digital sound model is in complete accordance with the way we tend to classify natural musical instruments according to their sound generation mechanism. For example, strings and woodwinds are normally seen as timbral classes of acoustic instruments characterized by their sound generation mechanism. It should be quite clear that the degree of compactness of a class of sounds is determined, on one hand, by the sensitivity of the digital model to parameter variations and, on the other hand, on the amount of control that is necessary for obtaining a certain desired sound. As an extreme example we may think of a situation in which a musician is required to generate sounds sample by sample, while the task of the computing equipment is just that of playing the samples. In this case the control signal is represented by the sound itself, therefore the class of sounds that can be produced is unlimited but the instrument is impossible for a musician to control and play. An opposite extremal situation is that in which the synthesis technique is actually the model of an acoustic musical instrument. In this case the class of sounds that can be produced is much more limited (it is characteristic of the mechanism that is being modeled by the algorithm), but the degree of difficulty involved in generating the control parameters is quite modest, as it corresponds to physical parameters that have an intuitive counterpart in the experience of the musician.

An interested conclusion that could be already drawn in the light of what stated above is that the compactness of the class of sounds associated to a sound synthesis algorithm is somehow in contrast with the “playability” of the algorithm itself. One should remember that the ”playability” is of crucial importance for the success of a specific sound synthesis algorithm as, in order for a sound synthesis algorithm to be suitable for musical purposes, the musician needs an intuitive and easy access to its control parameters during both the sound design process and the performance. Such requirements often represents the reason why a certain synthesis technique is preferred to others.

Some considerations on control parameters are now in order. Varying the control parameters of a sound synthesis algorithm can serve several purposes, the first one of which is certainly that of exploring a sound space, i.e. producing all the different sounds that belong to the class characterized by the algorithm itself. This very traditional way of using control parameters would nowadays be largely insufficient by itself. As a matter of fact, with the progress in the computational devices that are currently being employed for musical purposes, the musician’s needs have turned more and more toward problems of timbral dynamics. For example, timbral differences between soft (dark) and loud (brilliant) tones are usually obtained through appropriate parameter control. Timbral expression parameters tend to operate at a note-level time-scale. As such, they can be suitably treated as signals characterized by a rather slow rate.

Another reason for the importance of time-variations in the algorithm parameters is that the musician needs to control the musical expression while playing. For example, staccato, legato, vibrato etc. need to be obtained through parameter control. Such parameter variations operate at a phrase-level time-scale. Because of that, they can be suitably treated as sequences of symbols events characterized by a very slow rate.

In conclusion, control parameters are signals characterized by their own time-scales. Controls signals for timbral dynamics are best described as discrete-time signals with a slow sampling rate, while controls for musical expression are best described by streams of asynchronous symbols events. As a consequence, the generation of control signals can once again be seen as a problem of signal synthesis.

2.2 Signal generators

In this category we find methods that directly generate the signal. We will see periodic waveform generators and noise generators.

2.2.1 Waveform generators

2.2.1.1 Digital oscillators

In many musical sounds, pitch is a characteristic to which we are quite sensitive. In examining the temporal waveform of pitched sounds, we see a periodic repetition of the waveform without great variations. The simplest synthesis method attempts to reproduce this characteristic, generating a periodic signal through a continuous repetition of a waveform. An algorithm that implements this method is called oscillator. A first method consists in computing the appropriate value of the function for every sample. Often function, as sinusoids are approximated by polynomial or rational truncated series. For example a sinusoid of frequency f can be computed by

$$s[n] = sin(\omega n) = p(\omega(n \bmod F_s))$$

where $\omega = 2\pi f/F_s$. More efficient algorithms will be presented in the next sections.

2.2.1.2 Table lookup oscillator

A very efficient approach is to precompute the samples of the waveform, store them in a table which is usually implemented as a circular buffer, and access them from the table whenever needed. If we store in the table a copy of one period of the desired waveform, when we cycle over the wavetable with the aid of a circular pointer, we generate a periodic waveform. When the pointer reach the end of the table, it wraps around and points again at the beginning of the table. Given a table of length L, the period of the generated waveform is given by $T_L = LT$, and its fundamental frequency by $f_0 = F_s/L$. If we want to change the frequency, we would need the same waveform stored in tables of different lengths. A better solution is to store many equidistant points of the (continuous) waveform in the table, and then read the value in correspondence of the desired abscissa. Obviously, the more numerous are the points in the table, the better the approximation will be. The oscillator cyclically searches the table to get the point nearest to the required one. In this way, the oscillator resample the table to generate a waveform with different frequency. The distance in the table between two samples at subsequent instants is called SI (*sampling increment*) and is proportional to the frequency f of the generated sound:

$$f = \frac{SI \cdot F_s}{L} \tag{2.1}$$

If the sampling increment SI is greater than 1, it can happen that the highest frequencies of the waveform overcome the Nyquist frequency $F_N = F_s/2$ giving rise to foldover.

M-2.1

Implement in Matlab a circular look-up from a table of length L and with sampling increment SI.

M-2.1 Solution

```
phi=mod(phi +SI,L);
s=tab[phi];
```

where `phi` is a state variable indicating the reading point in the table, `A` is a scaling parameter, `s` is the output signal sample. The function `mod(x,y)` computes the remainder of the division x/y and is used here to implement circular reading of the table. Notice that `phi` can be a non integer value. In order to use it as array index, it can be truncated, or rounded to the nearest integer. A more accurate output can be obtained by linear interpolation between adjacent table values.

2.2.1.3 Recurrent sinusoidal signal generators

Sinusoidal signal can be generated also by recurrent methods. A first method is base don the second order resonator filter (sect. 2.4.6.1.2) with the poles lying on the unit circle . The equation is

$$y[n+1] = 2\cos(\omega)y[n] - y[n-1] \tag{2.2}$$

where $\omega = 2\pi f/F_s$. With initial values $y[0] = 1$ and $y[-1] = \cos\omega$ the generator produces

$$y[n] = \cos n\omega = \cos(2\pi f T n)$$

With $y[0] = 0$ and $y[-1] = -\sin\omega$ the generator produces $y[n] = \sin n\omega$. In general if $y[0] = \cos\phi$ and $y[-1] = \cos(\phi - \omega)$ the generator produces $y[n] = \cos(n\omega + \phi)$. This property can be justified remembering the trigonometric relation $\cos\omega \cdot \cos\phi = 0.5[\cos(\phi + \omega) + \cos(\phi - \omega)]$.

Another method that combine both the sinusoidal and cosinusoidal generators is the so called coupled form described by the equations

$$\begin{aligned} x[n+1] &= \cos\omega \cdot x[n] - \sin\omega \cdot y[n] & (2.3)\\ y[n+1] &= \sin\omega \cdot x[n] + \cos\omega \cdot y[n] & (2.4)\end{aligned}$$

With $x[0] = 1$ and $y[0] = 0$ we have $x[n] = \cos(n\omega)$ and $y[n] = \sin(n\omega)$. This property can be verified considering that if we define a complex variable $\alpha[n] = x[n] + jy[n] = \exp(jn\omega)$, it results $\alpha[n+1] = \exp(j\omega) \cdot \alpha[n]$. The real and imaginary parts of this relation give equations 2.4 and 2.4 respectively.

Both methods have the drawback that coefficient quantization can give rise to numerical instability, i.e. poles outside the unitary circle. The waveform will then tend to grow exponentially or to decay rapidly into silence. To avoid this problem, a periodic re-initialization is advisable. It is possible to use a slightly different set of coefficients to produce absolutley stable sinusoidal waveforms

$$\begin{aligned} x[n+1] &= x[n] - c \cdot y[n] & (2.5)\\ y[n+1] &= c \cdot x[n+1] + y[n] & (2.6)\end{aligned}$$

where $c = 2\sin(\omega/2)$. With $x[0] = 1$ and $y[0] = c/2$ we have $x[n] = \cos(n\omega)$.

2.2.1.4 Amplitude/frequency controlled oscillators

The amplitude and frequency of a sound are usually required to be time-varying parameters. Amplitude control is needed in order to define suitable sound envelopes, or to create *tremolo* effects (quasi-periodic amplitude variations around an average value). Frequency control is needed to simulate *portamento* between two tones, or subtle pitch variations in the sound attack/release, or *vibrato* effects (quasi-periodic pitch variations around an average value), and so on.

We then want to have at our disposal a digital oscillator of the form

$$s[n] = A[n] \cdot \mathrm{tab}[\phi[n]], \tag{2.7}$$

where $A[n]$ scales the amplitude of the signal, and the phase $\phi[n]$ does not in general increase linearly in time and is computed as a function of the instantaneous frequency. Figure 2.1 shows the symbol usually adopted to depict an oscillator with fixed waveform and varying amplitude and frequency

Many sound synthesis languages (e.g., the well known *Csound*) define control signals at *frame rate*: a frame is a time window with pre-defined length (typically 5 to 50 ms), in which the control signals can be reasonably assumed to be approximately constant. This approximation clearly helps to reduce computational loads significantly.

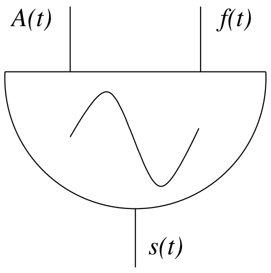


Figure 2.1: Symbol of the fixed waveform oscillator, with varying amplitude and frequency.

M-2.2

Assume that a function `sinosc(t0,a,f,ph0)` realizes a sinusoidal oscillator (`t0` is the initial time, `a,f` are the frame-rate amplitude and frequency vectors, and `ph0` is the initial phase). Then generate a sinusoid of length 2 s, with constant amplitude and frequency.

M-2.2 Solution

```
%%% headers %%%
global Fs;              %sample rate
global SpF;             %samples per Frame
Fs=22050;
ControlW=0.01;          %control window (in sec): 10 ms
SpF=round(Fs*ControlW);
Fc=Fs/SpF;              %control rate

%%% define controls %%%
slength=2;             %soundlength in seconds
nframes=slength*Fc;    %total number of frames
a=ones(1,nframes);     %constant amplitude
f=50*ones(1,nframes);  %constant frequency

%%% compute sound %%%
s=sinosc(0,a,f,0);     %sound signal
```

Note the structure of this simple example: in the “headers” section some global parameters are defined, that need to be known also to auxiliary functions; a second section defines the control parameters, and finally the audio signal is computed.

M-2.3

When the oscillator frequency is constant the phase is a linear function of time, $\phi(t) = 2\pi f t$, therefore in the digital domain ϕ can be computed as $\phi[n+1] = \phi[n] + 2\pi f/F_s$. In the more general case in which the frequency varies at frame rate, we have to understand how to compute the phase of the oscillator. The starting point is the equation

$$f(t) = \frac{1}{2\pi}\frac{d\phi}{dt}(t), \tag{2.8}$$

which simply says that the radian frequency $\omega(t) = 2\pi f(t)$ is the instantaneous angular velocity of the time-varying phase $\phi(t)$. If $f(t)$ is varying slowly enough (i.e. it is varying at frame rate), we can say that in the K-th frame the first-order approximation

$$\frac{1}{2\pi}\frac{d\phi}{dt}(t) = f(t) \sim f(T_K) + F_c\left[f(T_{K+1}) - f(T_K)\right]\cdot(t - T_K) \tag{2.9}$$

holds, where T_K, T_{K+1} are the initial instants of frames K and $K+1$, respectively. The term $F_c\left[f(T_{K+1}) - f(T_K)\right]$ approximates the derivative df/dt inside the Kth frame. We can then find the phase by integrating equation (2.9):

$$\phi(t) = \phi(T_k) + 2\pi f(T_k)(t - T_k) + 2\pi F_c[f(T_{K+1}) - f(T_K)]\frac{(t-T_K)^2}{2},$$

$$\phi((K-1)\cdot \mathsf{SpF} + n) = \phi(K) + 2\pi\frac{f(K)n}{F_s} + \pi\frac{f(K+1) - f(K)}{\mathsf{SpF}\cdot F_s}n^2, \tag{2.10}$$

where $n = 0\ldots(\mathsf{SpF} - 1)$ spans the frame. In summary, equation (2.10) computes ϕ at sample rate, given the frame rate frequencies. The key ingredient of this derivation is the linear interpolation (2.9).

Realize the `sinosc(t0,a,f,ph0)` function that we have used in M-2.2. Use equation (2.10) to compute the phase given the frequency vector `f`.

M-2.3 Solution

```
function s = sinosc(t0,a,f,ph0);

global SpF;             %samples per frame
global Fs;              %sampling rate

nframes=length(a);      %total number of frames
if (length(f)==1) f=f*ones(1,nframes); end
if (length(f)~=nframes) %check
   error('f and a must have the same length!');
end

s=zeros(1,nframes*SpF);        %signal vector (initialized to 0)
lastampl=a(1);
lastfreq=f(1);
lastphase=ph0;
for i=1:nframes                  %cycle on the frames
   naux=1:SpF;                   %counts samples within frame
   ampl=lastampl +...            %compute amplitudes within frame
        (a(i)-lastampl)/SpF.*naux;
   phase=lastphase+pi/Fs.*naux.* ... %compute phases within frame
         (2*lastfreq +(1/SpF)*(f(i)-lastfreq).*naux);
   s(((i-1)*SpF+1):i*SpF)=ampl.*cos(phase); %read from table
```

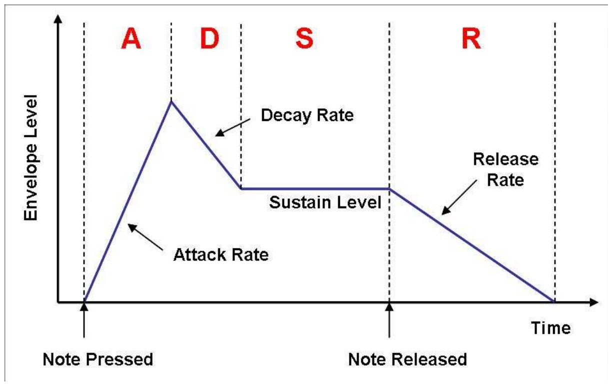


Figure 2.2: ADSR envelope described by four phases: Attack, Decay, Sustain e Release.

```
   lastampl=a(i);                             %save last values
   lastfreq=f(i);                             %of amplitude,
   lastphase=phase(SpF);                      %frequency, phase
end


s=[zeros(1,round(t0*Fs+1)) s]; %add initial silence of t0 sec.
```

Both the amplitude `a` and frequency `f` envelopes are defined at frame rate and are interpolated at sample rate inside the function body. Note in particular the computation of the `phase` vector within each frame.

2.2.1.5 Envelope generators

It is possible to use the same algorithm of table look-up oscillator to produce time envelopes. In this case, to generate a time envelope of d sec and scan only once the table, a sampling increment $SI = L/(F_s d)$ should be used. Often in sound synthesis, the amplitude envelope is described by a linearly varying function. A typical schema is the ADSR envelope (Fig. 2.2). The time envelope is described by four phases of *Attack, Decay, Sustain e Release*. When we want to change the tone duration, it is advisable to only slightly modify the attack and release, that marks the identity of the sound, while the sustain can be lengthened more freely. So the oscillator table will be read once with different sampling increments for the different parts of the generated envelope.

The use of waveform and envelope generators allows to generate quasi periodic sounds with very limited hardware and constitutes the building block of many more sophisticated algorithms.

M-2.4

Write a function that realizes a line-segment envelope generator. The input to the function are a vector of time instants and a corresponding vector of envelope values.

M-2.4 Solution

```
function env = envgen(t,a,method);      %t vector of time instants
                                        %a vector of envelope values
```

```
global SpF;                                 %samples per frame
global Fs;                                  %sampling rate

if (nargin<3)
   method='linear';
end

frt=floor(t*Fs/SpF+1);                      %times instants as frame numbers
nframes=frt(length(frt));                   %total number of frames
env=interp1(frt,a,[1:nframes],method); %linear interpolation
```

The envelope shape is specified by break-points, described as couples (time instant (sec) and amplitude). The function generates the envelope at frame rate. Notice that the interpolation function `interp1` allows to easily use cubic of $spline$ interpolations.

M-2.5

Synthesize a modulated sinusoid using the functions `sinosc` and `envgen`.

M-2.5 Solution

```
%%% headers %%%
%[...]

%%% define controls %%%
a=envgen([0,.2,1,1.5,2],[0,1,.8,.5,0],'linear');  %ADSR amp. envelope
f=envgen([0,.2,1,2],[200,250,250,200],'linear');   %pitch envelope
f=f+max(f)*0.05*...                    %pitch envelope with vibrato added
      sin(2*pi*5*(SpF/Fs)*[0:length(f)-1]).*hanning(length(f))';

%%% compute sound %%%
s=sinosc(0,a,f,0);
```

In fig. 2.3 amplitude `a` and frequency `f` control signals are shown.

2.2.2 Noise generators

Up to now, we have considered signals whose behavior at any instant is supposed to be perfectly knowable. These signals are called deterministic signals. Besides these signals, *random signals* of unknown or only partly known behavior may be considered. For random signals, only some general characteristics, called statistical properties, are known or are of interest. The statistical properties are characteristic of an entire signal class rather than of a single signal. A set of random signals is represented by a random process. Particular numerical procedures simulate random processes, producing sequences of random (or more precisely, pseudorandom) numbers.

Random sequences can be used both as signals (i.e., to produce white or colored noise used as input to a filter) and a control functions to provide a variety in the synthesis parameters most perceptible by the listener. In the analysis of natural sounds, some characteristics vary in an unpredictable way; their mean statistical properties are perceptibly more significant than their exact behavior. Hence, the addition of a random component to the deterministic functions controlling the synthesis parameters

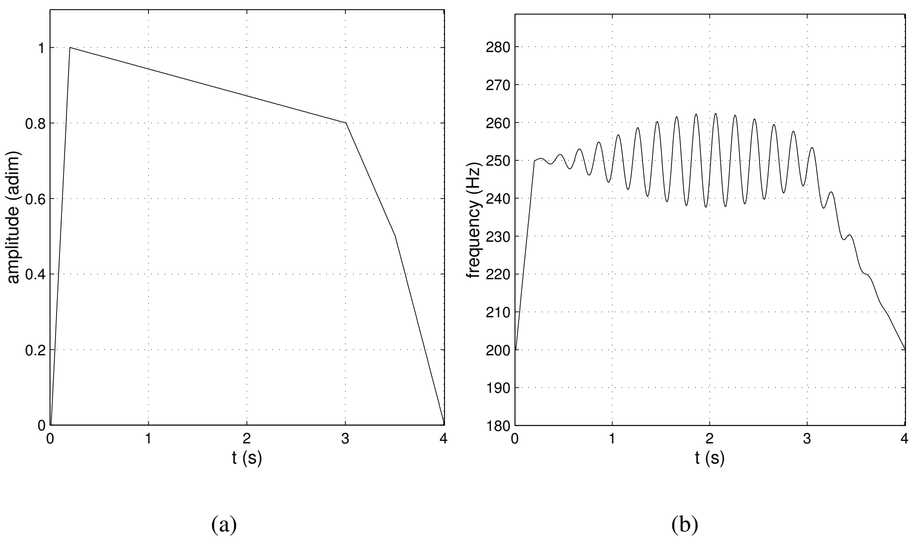


Figure 2.3: Amplitude (a) and frequency (b) control signals

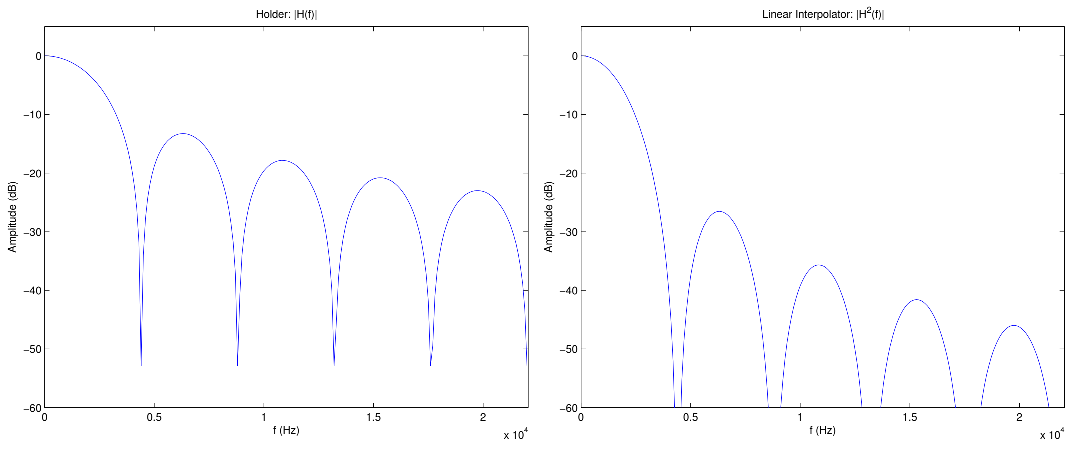
is often desirable. In general, a combination of random processes is used because the temporal organization of the musical parameters often has a hierarchical aspect. It cannot be well described by a single random process, but rather by a combination of random processes evolving at different rates. For example this technique is employed to generate $1/f$ noise.

2.2.2.1 Random noise models

2.2.2.1.1 White noise generators The spread part of the spectrum is perceived as random noise. In order to generate a random sequence, we need a random number generator. There are many algorithms that generate random numbers, typically uniformly distributed over the standardized interval $[0, 1)$. However it is hard to find good random number generators, i.e. that pass all or most criteria of randomness. The most common is the so called *linear congruential* generator. It can produce fairly long sequences of independent random numbers, typically of the order of two billion numbers before repeating periodically. Given an initial number (seed) $I[0]$ inn the interval $0 \leq I[0] < M$, the algorithm is described by the recursive equations

$$\begin{aligned} I[n] &= (\, aI[n-1] + c\,) \bmod M \\ u[n] &= I[n]/M \end{aligned} \tag{2.11}$$

where a and c are two constants that should be chosen very carefully in order to have a maximal length sequence, i.e. long M samples before repetition. The actual generated sequence depends on the initial value $I[0]$; that is way the sequence is called pseudorandom. The numbers are uniformly distributed over the interval $0 \leq u[n] < 1$. The mean is $E[u] = 1/2$ and the variance is $\sigma_u^2 = 1/12$. The transformation $s[n] = 2u[n] - 1$ generates a zero-mean uniformly distributed random sequence over the interval $[-1, 1)$. This sequence corresponds to a white noise signal because the generated numbers are mutually independent. The power spectral density is given by $S(f) = \sigma_u^2$. Thus the sequence contains all the frequencies in equal proportion and exhibits equally slow and rapid variation in time.



(a) (b)

Figure 2.4: Spectral envelope $|H(f)|$ of low frequency noise generators where a new random number is generated every $d = 10$ samples: (a) hold generator; (b) linear interpolator.

With a suitable choice of the coefficients a and b, it produces pseudorandom sequences with flat spectral density magnitude (white noise). Different spectral shapes ca be obtained using white noise as input to a filter.

M-2.6

A method of generating a Gaussian distributed random sequence is based on the central limit theorem, which states that the sum of a large number of independent random variables is Gaussian. As exercise, implement a very good approximation of a Gaussian noise, by summing 12 independent uniform noise generators.

2.2.2.1.2 Pink noise generators If we desire that the numbers vary at a slower rate, we can generate a new random number every d sampling instants and hold the previous value in the interval (*holder*) or interpolate between two successive random numbers (interpolator). In this case the power spectrum is given by

$$S(f) = |H(f)|^2 \frac{\sigma_u^2}{d}$$

with

$$|H(f)| = \left| \frac{\sin(\pi f d/F_s)}{\sin(\pi f/F_s)} \right|$$

for the holder (fig. 2.4(a)) and

$$|H(f)| = \frac{1}{d} \left[\frac{\sin(\pi f d/F_s)}{\sin(\pi f/F_s)} \right]^2$$

for linear interpolation (fig. 2.4(b)).

2.2.2.1.3 $1/f$ noise generators A $1/f$ noise, also called pink noise, is characterized by a power spectrum that fall in frequency like $1/f$

$$S(f) = \frac{A}{f} \tag{2.12}$$

To avoid the infinity at $f = 0$, this behaviour is assumed valid for $f \geq f_{min}$, where f_{min} is a desired minimum frequency. The spectrum is characterized by a 3 db per octave drop, i.e. $S(2f) = S(f)/2$. The amount of power contained within a frequency interval $[f_1, f_2]$ is

$$\int_{f_1}^{f_2} S(f)df = A \ln \left(\frac{f_1}{f_2}\right)$$

This implies that the amount of power in any octave is the same. $1/f$ noise is ubiquitous in nature and is related to fractal phenomena. In audio domain it is known as pink noise. It represents the psychoacoustic equivalent of the white noise because he approximately excites uniformly the critical bands. The physical interpretation is a phenomenon that depends on many processes that evolve on different time scales. So a $1/f$ signal can be generated by the sum of several white noise generators that are filtered through first¡order filters having the time constants that are successively larger and larger, forming a geometric progression.

M-2.7

In the Voss $1/f$ noise generation algorithm, the role of the low pass filters is played by the hold filter seen in the previous paragraph. The $1/f$ noise is generated by taking the average of several periodically held generators $y_i[n]$, with periods forming a geometric progression $d_i = 2^i$, i.e.

$$y[n] = \frac{1}{M}\sum_{i=1}^{M} y_i[n] \tag{2.13}$$

The power spectrum does not have an exact $1/f$ shape, but it is close to it for frequencies $f \geq F_s/2^M$. As exercise, implement a $1/f$ noise generator and use it to assign the pitches to a melody.

M-2.8

The music derived from the $1/f$ noise is closed to the human music: it does not have the unpredictability and randomness of white noise nor the predictability of brown noise. $1/f$ processes correlate logarithmically with the past. Thus the averaged activity of the last ten events has as much influence on the current value as the last hundred events, and the last thousand. Thus they have a relatively long-term memory.
$1/f$ noise is a fractal one; it exhibits self-similarity, one property of the fractal objects. In a self-similar sequence, the pattern of the small details matches the pattern of the larger forms, but on a different scale. In this case, is used to say that 1/f fractional noise exhibits statistical self-similarity. The pink noise algorithm for generating pitches has become a standard in algorithmic music. Use the $1/f$ generator developed in M-2.7 to produce a fractal melody.

2.3 Time-segment based models

2.3.1 Wavetable synthesis

2.3.1.1 Definitions and applications

Finding a mathematical model that faithfully imitates a real sound is an extremely difficult task. If an existing reference sound is available, however, it is always possible to reproduce it through recording.

Such a method, though simple in its principle, is widely adopted by digital sampling instruments or samplers and is called wavetable synthesis or sampling. Samplers store a large quantity of examples of complete sounds, usually produced by other musical instruments. When we wish to synthesize a sound we just need to directly play one sound of the stored repertoire.

The possibility of modification is rather limited, as it would be for the sound recorded by a tape deck. The most common modification is that of somewhat varying the sampling rate (speed) when reproducing the sound, which results in a pitch deviation. On the other hand, what makes the method interesting the most is certainly the variety of sounds available.

From the implementation viewpoint, computational simplicity and limited amount of information to be stored are two contrasting needs for samplers. In fact, in order to reduce the data to be stored, it is possible to adopt looping techniques with almost any stationary portion of sounds. One method of improving the expressive possibilities of samplers is store multiple of the sounds at different pitches, and switching or interpolating between these upon synthesis. This method, called *multisampling*, also might include the storage of separate samples for loud and soft sounds. During synthesis a linear interpolation between these sampled is performed as function of the desired loudness. Infact most instruments exibit richer spectra for louder sounds. Sometime a filter are used to control the spectral variation llowing to obtain softer sounds from a stored loud sound.

In most cases sampling techniques are presented as a method for reproducing natural sounds and is evaluated in comparison with the original instruments. This is the main reason why the most popular commercial digital keyboards, such as electronic pianos and organs, adopt this synthesis technique. Of course, sampling cannot feature all the expressive possibilities of the original instrument. Notice that sampled sounds can also be obtained synthetically or through the modification of other sounds, which is a way of widening the range of possibilities of application of samplers. From the composer's viewpoint, the use of samplers represents a practical approach to the so-called *musique concrète*. This kind of music, begun in Paris in late Forties by the main effort of Pierre Schaeffer, started to use, as basic sonic material of its musical compositions, any kind of sound, recorded by a microphone and eventually processed.

2.3.1.2 Tranformations: pitch shift, looping

The most common modification is that of varying the sampling rate (speed) when reproducing the sound, which results in a pitch deviation. In digital domain this effect is obtained by resampling the stored waveform scanning the table with a sampling increment different than 1. The algorithm is similar to the digital oscillator by table look-up (see Sect.2.2.1.1. In this case if we want to change the frequency f_{stored} of the stored sound to a new frequency f_{new}, we will read the table with a sampling increment

$$SI = \frac{f_{new}}{f_{stored}}$$

However, substantial pitch variations are generally not very satisfactory as a temporal waveform compression or expansion results in unnatural timbral modifications, which is exactly what happens with an accelerated tape recorder. It is thus necessary to allow only pitch variations of few semitones to take place for the synthetic sound to be similar to the original one. This is expecially true for sounds characterized by formants in the spectrum. For a good sampling synthesis of the whole pitch extension, many samples should be stored (e.g. three for each octave). Moreover special care should be payed to assure that adjacent sounds be similar.

M-2.9

Import a .wav file of a single instrument tone. Scale it (compress and expand) to different extents and listen to the new sounds. Up to what scaling ratio are the results acceptable?

Often it is desired to vary the sound also in function of other parameters, the most important being the intensity. To this purpose it not sufficient to change the sound amplitude by a multiplication, by it is necessary to modify the timbre of the sound. In general louder sounds are characterized by a sharper attack and by a brighter spectrum. In this case a technique could be to use a unique sound prototype (e.g. a tone played fortissimo) and then obtaining the other intensity by simple spectral processing, as low pass filtering. A different and more effective solution, is to use a set of different sound prototype, recorder with different intensity (e.g. tones played fortissimo, mezzo forte, pianissimo) and then obtaining the other dynamic values by interpolations and further processing.

This technique is thus characterized by high computational efficiency and high imitation quality, but by low flexibility for sounds not initially included in the repertoire or not easily obtainable with simple transformations. There is a trade-off of memory size with sound fidelity.

In order to employ efficiently the memory, often the sustain part of the tone is not entirely stored but only a part (or few significant parts) and in the synthesis this part is repeated (*looping*). Naturally the repeated part should not be to short, to avoid a static character of the resulting sound. For example to lengthen the duration of a note, first the attack is reproduced without modification, then the sustain part is cyclically repeated, with possible cross interpolation among the different selected parts, and finally the sound release stored part is reproduced. Notice that if we want to avoid artefacts in cycling, particular care should be devoted to choosing the points of the beginning and ending of the loop. Normally an integer number of periods is used for looping starting with a null value, to avoid amplitude or phase discontinuities. In fact these discontinuities are very annoying. To this purpose it may be necessary to process the recorded samples by slightly changing the phases of the partials.

M-2.10

Import a .wav file of a single instrument tone. Find the stationary (sustain) part, isolate a section, and perform the looping operation. Listen to the results, and listen to the artifacts when the looped section does not start/end at zero-crossings.

If we want a less static sustain, it is possible to individuate some different and significant sound segments, and during the synthesis interpolate (*cross-fade*) among subsequent segments. In this case the temporal evolution of the tone can be more faithfully reproduced.

2.3.2 Granular synthesis

Granular synthesis, together with additive synthesis, shares the idea of building complex sounds from simpler ones. Granular synthesis assumes that a sound can be considered as a sequence, possibly with overlaps, of elementary acoustic elements called grains. Granular synthesis constructs complex and dynamic acoustic events starting from a large quantity of grains. The features of the grains and their temporal location determine the sounds timbre. We can see it as being similar to the cinema, where a rapid sequence of static images gives the impression of objects in movement.

The initial idea of granular synthesis dates back to Gabor's work aimed at pinpointing the physical and mathematical ideas needed to understand what a time-frequency spectrum is. He considered sound as a sum of elementary Gaussian functions that have been shifted in time and frequency. Gabor considered these elementary functions as *acoustic quanta*, the basic constituents of a sound. In the scientific field these works have been rich in implications and have been the starting point for studying time-frequency representations. The usual Gabor expansion on a rectangular time-frequency lattice of a signal $x(t)$ can be expressed as a linear combination of properly shifted and modulated versions $g_{mk}(t)$ of a synthesis window $g(t)$

$$x(t) = \sum_m \sum_k a_{mk}\, g_{mk}(t)$$

with

$$g_{mk} = g(t - m\alpha T)e^{jk\beta\Omega t}$$

The time step αT and the frequency step $\beta\Omega$ satisfy the relationship $\Omega T = 2\pi$ and $\alpha\beta \leq 1$.

In music, granular synthesis arises from the experiences of taped electronic music. In the beginning musicians had tools that did not allow a great variation of timbre, for example fixed waveform oscillators and filters. They obtained dynamic sounds by cutting tapes into short sections and the putting together again. The rapid alternation of acoustic elements give a certain variety to the sound. The source of the sound could be electronic or live recorded sounds that were sometimes electronically processed. Iannis Xenakis developed this method in the field of analog electronic music. Starting from Gabors theory, Xenakis considers the grains as being music quanta and suggested a method of composition that is based on the organization of the grains by means of screen sequences, which specify the frequency and amplitude parameters of the grains at discrete points in time. In this way a common conceptual approach is used both for micro and macro musical structure.

2.3.2.1 Sound granulation

Two main approaches to granular synthesis can be identified: the former based on sampled sounds and the latter based on abstract synthesis. In the first case, *sound granulation*, complex waveforms, extracted from real sounds or described by spectra, occurs in succession with partial overlap with the method called Overlap And Add (OLA). In this way, it is possible both to reproduce accurately real sounds and modify then in their dynamic characteristics.

Let $x[n]$ and $y[n]$ be the input and output signals. The grains $g_k[i]$ are extracted from the input signal with the help of a window function $w_k[i]$ of length L_k by

$$g_k[i] = x[i + i_k]\; w_k[i]$$

with $i = 0 \ldots L_{k-1}$. The time instant i_k indicates the point where the segment is extracted; the length L_k determines the amount of signal extracted; the window waveform $w_k[i]$ should ensure fade-in and fade-out at the border of the grain and affects the frequency content of the grain. Long grains tend to maintain the timbre identity of the portion of the input signal, while short ones acquire a pulse-like quality. When the grain is long, the window has a flat top and it used only to fade-in and fade-out the borders of the segment.

As in additive synthesis the organization of the choice of the frequencies is very important, so in granular synthesis the proper timing organization of the grain is essential to avoid artifacts produced by discontinuities. This problem makes often the control quite difficult. An example of use is the synthesis of a time varying stochastic component of a signal. In this case it is only necessary to control the spectral envelope. To this purpose, it is convenient to employ the inverse Fourier transform of a spectrum, whose magnitude is defined by the spectral envelope and the phase are generated randomly. Every frame is multiplied by a suitable window before the overlap-and-add, i.e. the sum of the various frames partially overlapped. It is possible also to use this approach for transforming sampled sounds (sound granulation). In this case grains are built selecting short segments of a sound, previously or directly recorded, and then are shaped by an amplitude envelope. These grains are used by the

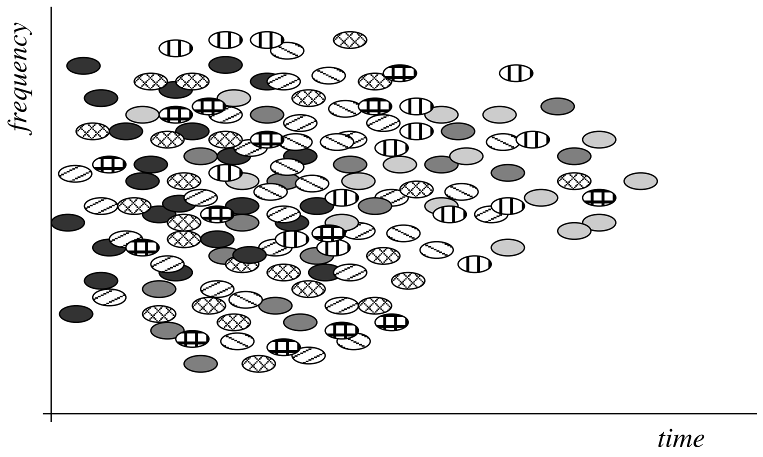


Figure 2.5: Representation of granular synthesis where grains derived from different sources are randomly mixed.

composer with a different order and time location or a different velocity. Notice that it is possible to extract grains from different sound files to create hybrid textures, e.g. evolving from one texture to another (fig. 2.5).

2.3.2.2 Synthetic grains

In abstract granular synthesis (second case), grains consist of arbitrary waveforms whose amplitude envelope is a short Gaussian function. Often frequency modulated Gaussian functions are used in order to localize the energy both in frequency and time domain. Grains are scattered on the frequency-time plane in the form of clouds. Notice that it is possible to recognize a certain similarity between this type of granular synthesis and the technique of mosaics, where the grains are single monochromatic tesseras and their juxtaposition produces a complex image.

In this case the waveform of the i-th grain (fig. 2.6) is given by

$$g_i[n] = w_i[n] \cdot \cos\left(2\pi \frac{f_i}{F_s} n + \phi_i\right)$$

where $w_i[n]$ is a window of duration N_i samples. The synthesis expression is given by

$$s[n] = \sum_i a_i \cdot g_i[n - n_i] \tag{2.14}$$

where a_i is the amplitude coefficient of the i-th grain and n_i is its time instant. Every grain contributes to the total energy around the point (n_i, f_i) of the time frequency plane.

The most important and classic type of granular synthesis, (*asynchronous granular synthesis*), is when simple grains are irregularly distributed in the time-frequency plane, e.g. they are scattered onto a mask that delimitate a particular area of the time-frequency-amplitude space. It results a cloud of micro-sounds or a sonic texture that is time-varying. Moreover the grain density inside the mask can be controlled. In this way many sound textures and natural noisy sounds can be modelled where general statistical properties are more important than the exact sound evolution. These kind of sounds can be defined as the accumulation of more or less complex sonic grains, with their proper temporal

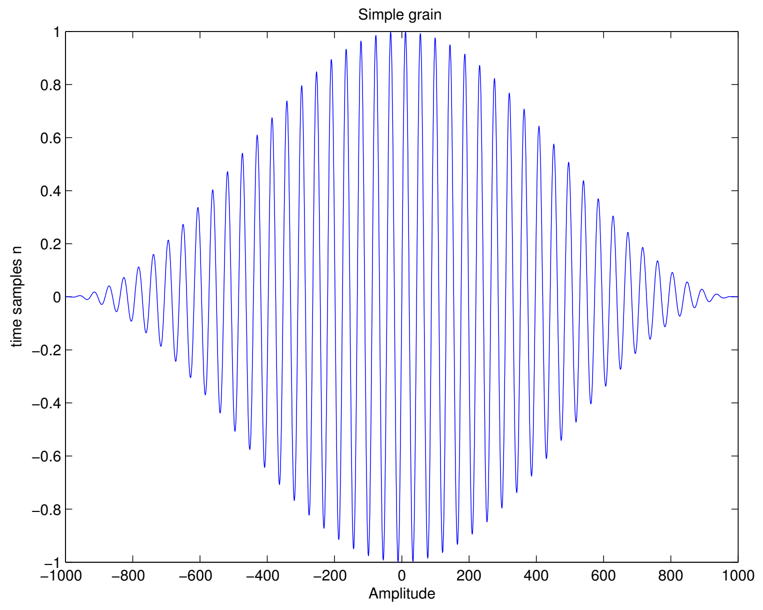


Figure 2.6: Example of a synthetic grain waveform.

and spectral variability. For example, the sound of rice falling onto a metal plate is composed of thousands of elementary ticks; the rain produces, in the same way, the accumulation of a large amount of water droplet micro-sounds. Scratching or cracking sounds made by the accumulation of thousands of complex micro-sounds not necessarily deterministic. In fact, in the real world, when multiple realizations of a same event, of a same phenomenon occur, we can expect these types of sounds.

Grain duration affects the sonic texture: short duration (few samples) produces a noisy, particulate disintegration effect; medium duration (tents of ms) produces fluttering, warbling, gurgling; longer durations (hundreds of ms) produce aperiodic tremolo, jittering spatial position. When the grains are distributed on a large frequency region, the texture has a massive character, while when the band is quite narrow, it result a pitched sound. Sparse densities (e.g. 5 grains per second) give rise to a pointillistic texture.

2.3.2.3 Concatenative speech synthesis

Connecting prerecorded natural utterances is probably the easiest way to produce intelligible and natural sounding synthetic speech. However, concatenative synthesizers are usually limited to one speaker and one voice and usually require more memory capacity than other methods.

One of the most important aspects in concatenative synthesis is to find correct unit length. The selection is usually a trade-off between longer and shorter units. With longer units high naturalness, less concatenation points and good control of coarticulation are achieved, but the amount of required units and memory is increased. With shorter units, less memory is needed, but the sample collecting and labeling procedures become more difficult and complex. In present systems units used are usually words, syllables, demisyllables, phonemes, diphones, and sometimes even triphones.

- Word is perhaps the most natural unit for written text and some messaging systems with very limited vocabulary. Concatenation of words is relative easy to perform and coarticulation effects within a word are captured in the stored units. However, there is a great difference with words spoken in isolation and in continuos sentence which makes the continuous speech to sound very unnatural. Because there are hundreds of thousands of different words and proper names in each language, word is not a suitable unit for any kind of unrestricted Text To Speech system.

- Demisyllables represents the initial and final parts of syllables. One advantage of demisyllables is that only about 1,000 of them is needed to construct the 10,000 syllables of English. Using demisyllables, instead of for example phonemes and diphones, requires considerably less concatenation points. Demisyllables also take account of most transitions and then also a large number of coarticulation effects and also covers a large number of allophonic variations due to separation of initial and final consonant clusters.
- Phonemes are probably the most commonly used units in speech synthesis because they are the normal linguistic presentation of speech. The inventory of basic units is usually between 40 and 50, which is clearly the smallest compared to other units. Using phonemes gives maximum flexibility with the rule-based systems. However, some phones that do not have a steady-state target position, such as plosives, are difficult to synthesize. The articulation must also be formulated as rules. Phonemes are sometimes used as an input for speech synthesizer to drive for example diphone-based synthesizer.
- Diphones (or dyads) are defined to extend the central point of the steady state part of the phone to the central point of the following one, so they contain the transitions between adjacent phones. That means that the concatenation point will be in the most steady state region of the signal, which reduces the distortion from concatenation points. The number of units is usually from 1500 to 2000, which increases the memory requirements and makes the data collection more difficult compared to phonemes. However, diphone is a very suitable unit for sample-based text-to-speech synthesis.

A problem in concatenative synthesis compared to other methods is that data collecting and labeling of speech samples is usually time-consuming. In theory, all possible allophones should be included in the material, but trade-offs between the quality and the number of samples must be made. Moreover memory requirements are usually very high, especially when long concatenation units are used, such as syllables or words

2.3.3 Overlap And Add (OLA) method

A time-frequency representation can be seen as a series of overlapping FFTs with or without windowing. Based on the normal Fourier Transform of a short-time spectrum, we have that a frame $X_n(e^{j\omega_k}$ of the STFT is the DFT of the signal windowed segment $y_n(m) = x(m)w(n-m)$. We can reconstruct $x(m)$ by computing the inverse DFT (IDFT) of $X_n(e^{j\omega_k})$ and dividing out the window (assuming non-zero for all samples). This process gives L signal values for each window, where L is the window length. Then the window can be moved by L samples and the process repeated. However, since $X_n(e^{j\omega_k}$ is undersampled in time, it is highly susceptible to aliasing errors. Thus a more robust synthesis procedure is needed.

The idea of Overlap And Add (OLA) method is to overlap the segments reconstructed by the IDFT and multiply them by a synthesis window $v[n]$ before the summation of the overlapping sections. The synthesis algorithm is given by

$$y[n] = \sum_s v[n - sR_s]\, y_i[n - sR_s] \qquad (2.15)$$

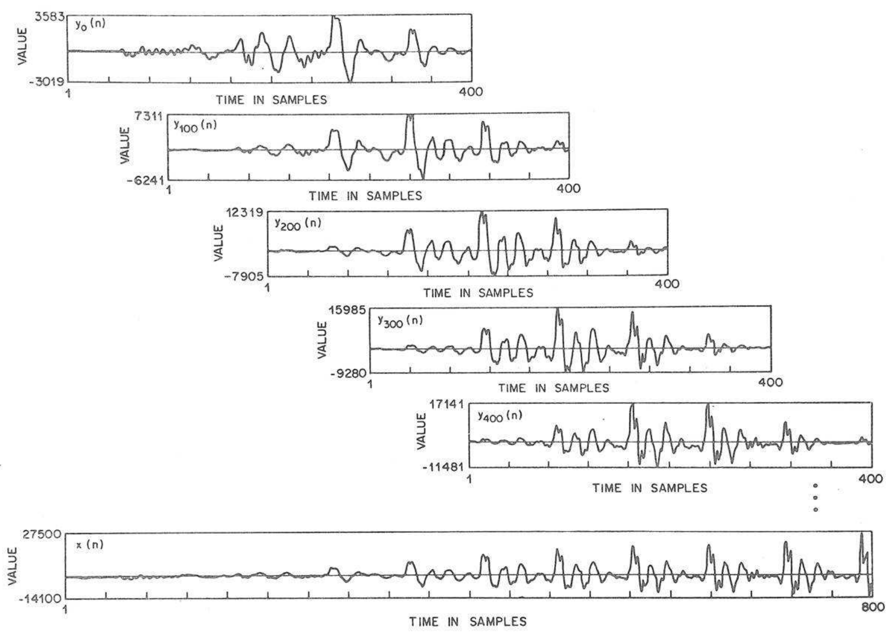


Figure 2.7: Overlap And Add (OLA) method

where s is the frame index, R_s is the synthesis hop size and $i = sR_s$ is reference instant of the segment y_i. The finite length signals y_i are computed from the inverse Fourier transform of the short-time spectra $Y[sR_s, k]$ eventually modified in amplitude and phase. A perfect reconstruction can be achieved, if the sum of the overlapping windows is unity. Normally an L- point Hamming window is used with $R_s = L/4$ (see Fig. 2.7).

Some digital sound effects can be obtained emplying this syntesis technique. A robotization effect can be obtained by putting zero phase values on every FFT before reconstruction. The ffect applies a fixed pitch onto a sound. Moreover, as it forces the sound to be periodic, many erratic and random variations are converted into robotic sounds.

If we deliberately impose a random phase on a time-frequency representation, we can have a different behavior depending on the length of the window: if the window is quite large (for example, 2048 for a sampling rate of 44100 Hz), the magnitude will represent the behavior of the partials quite well and changes in phase will produce an uncertainty over the frequency. But if the window is small (e.g. 64 points), the spectral envelope will be enhanced and this will lead to a whispering effect.

2.3.4 SOLA algorithm for time stretching

The basic idea of time stretching by time-segment processing is to divide the input sound into segments. Then if the sound should be lengthened, some segments are repeated, while if the sound should be shortened, some segments are discarded. A possible problem is amplitude and phase discontinuity at the boundaries of the segments. Amplitude discontinuities are avoided by partially overlapping the blocks, while phase discontinuities are avoided by a proper time alignment of the blocks.

For the most part, commercial time-stretching software packets used by PC users adopt the SOLA (Synchronous OverLap and Add) algorithm, which bases the synthesis phase on the sum of windowed segments of the given signal. The SOLA method, (fig. 2.8), considers the input signal $x[n]$ divided in

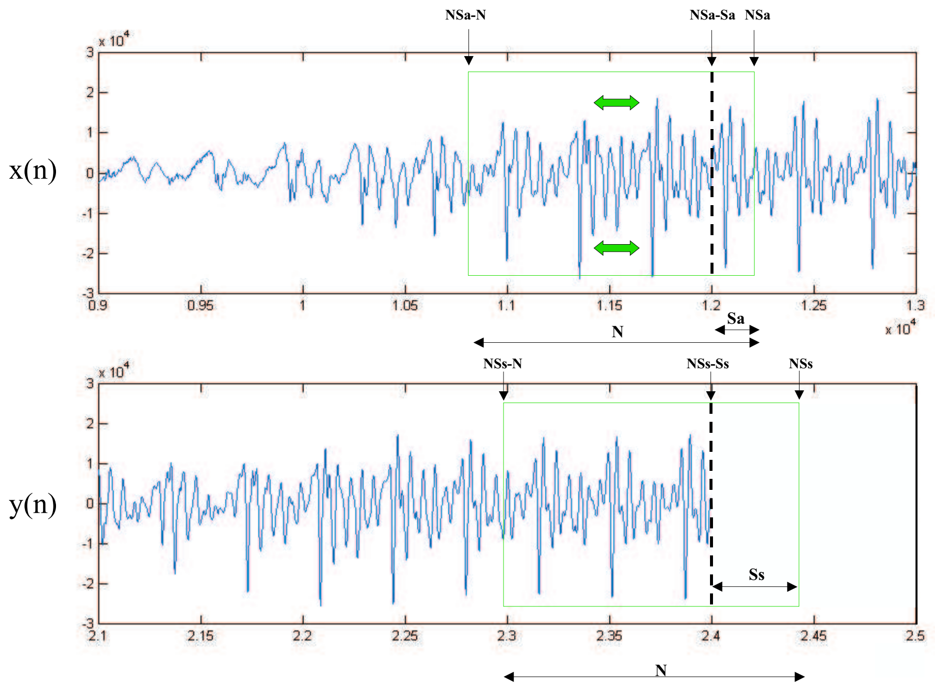


Figure 2.8: The generic SOLA algorithmic step

fixed-length frames of N samples each, taken every S_a samples (a quantity called *analysis hop size*); the output signal $y[n]$, on the contrary, is logically divided in frames of the same length (N samples), but synthesized with a different step (S_s, *synthesis hop size*): in this way the stretching factor is given by the ratio between the synthesis process speed (S_s samples per step) and the analysis process speed (S_a samples per step), $\alpha = S_s/S_a$.

After an initialization phase, where SOLA copies the first N samples from $x[n]$ to $y[n]$, to obtain a minimum set of samples to work on, $y[j] = x[j]$ for $j = 0 \dots N-1$. Let $NS_a = N + S_a$ and $NS_s = N + S_s$, during the generic step the algorithm tries to find the best overlap between the synthesis frame (fixed) and the analysis one (repositionable in a given range). When this optimal overlap if found, it copies the last S_s samples from the input frame to the end of the current output frame and mixes the $(N - S_s)$ preceding from each frame with a *linear crossfade* to obtain a gradual transition to the new packet of samples:

$$\begin{aligned} y[NS_s - S_s + j] &= x[NS_a - S_s + k_m + j] \\ &\quad \text{with } 0 \le j < S_s - 1 \\ y[NS_s - N + j] &= (1 - v[j])y[NS_s - N + j] + v[j]x[NS_a - N + j + k_m] \\ &\quad \text{with } 0 \le j < N - S_s - 1 \end{aligned}$$

where $v[j]$ is a linear function, called (*smoothing function*), for crossfading the two segments. The final effect of this method is a local replication of the waveform periods (more or less evident depending from the value of parameter α), located by a *synchronization function*, which finds during each step, as stated, the optimal overlap between frames. This local replication of waveform periods permits us to obtain a synthesized waveform with the same spectral properties, but with an altered temporal evolution.

To find the optimal value for the discrete time lag k_m, assuring the best match between the two N-sample frames, we can use one of the three most diffused techniques:

1. Computation of the minimum vectorial inter-frame distance in L_1 sense (cross-AMDF)
2. Computation of the maximum cross-correlation $r_{x,y}(k)$

$$r_{x,y}(k) = \sum_{i=0}^{L_m} x[N+k+i] \cdot y[NS_s - N + i]$$
$$-k_{max} \le k \le k_{max}, \quad L_m \doteq N - S_s$$

3. Computation of the maximum normalized cross-correlation $r_N[k]$, where every value taken from $r_{x,y}[k]$ is normalized dividing it by the product of the frame energies (one of which is varying with k).

Even if the last technique is conceptually preferable, the second one is often used for computational efficiency.

2.4 Spectrum based models

Since the human ear acts as a particular spectrum analyser, a first class of synthesis models aims at modeling and generating sound spectra. The Short Time Fourier Transform and other time-frequency representations provide powerful sound analysis tools for computing the time-varying spectrum of a given sound. In this section models that can be interpreted in the frequency domain will be presented.

2.4.1 Sinusoidal model

When we analyze a pitched sound, we find that its spectral energy is mainly concentrated at a few discrete (slowly time-varying) frequencies f_k. These frequency lines correspond to different sinusoidal components called partials. If the sound is almost periodic, the frequencies of partials are approximately multiple of the fundamental frequency f_0, ie. $f_k(t) \simeq k\, f_0(t)$. The amplitude a_k of each partial is not constant and its time-variation is critical for timbre characterization. If there is a good degree of correlation among the frequency and amplitude variations of different partials, these are perceived as fused to give a unique sound with its timbre identity.

The sinusoidal model assumes that the sound can be modeled as a sum of sinusoidal oscillators whose amplitude $a_k(t)$ and frequency $f_k(t)$ are slowly time-varying

$$s_s(t) = \sum_k a_k(t)\, \cos(\phi_k(t)) \ , \tag{2.16}$$

$$\phi_k(t) = 2\pi \int_0^t f_k(\tau) d\tau + \phi_k(0) \ , \tag{2.17}$$

or, digitally,

$$s_s[n] = \sum_k a_k[n]\, cos(\phi_k[n]) \ , \tag{2.18}$$

$$\phi_k[n] = 2\pi\, f_k[n] T_s + \phi_k[n-1] \ , \tag{2.19}$$

where T_s is the sampling period. Notice that eq. 2.19 can also be written as

$$\phi_k(n) = \phi_{0k} + 2\pi T_s \sum_{j=1}^{n} f_k(n)$$

. Equations (2.16) and (2.17) are a generalization of the Fourier theorem, that states that a periodic sound of frequency f_0 can be decomposed as a sum of harmonically related sinusoids of frequency $f_i = if_0$

$$s_s(t) = \sum_k a_k \cos(2\pi k f_0 t + \phi_k) \ .$$

This model is also capable of representing aperiodic and inharmonic sounds, as long as their spectral energy is concentrated near discrete frequencies (spectral lines).

In computer music this model is called *additive synthesis* and is widely used in music composition. Notice that the idea behind this method is not new. As a matter of fact, additive synthesis has been used for centuries in some traditional instruments such as organs. Organ pipes, in fact, produce relatively simple sounds that, combined together, contribute to the richer spectrum of some registers. Particularly rich registers are created by using many pipes of different pitch at the same time. Moreover this method, developed for simulating natural sounds, has become the “metaphorical” foundation of a compositional methodology based on the expansion of the time scale and the reinterpretation of the spectrum in harmonic structures.

2.4.2 Spectral modeling

Spectral analysis of the sounds produced by musical instruments, or by any physical system, shows that the spectral energy of the sound signals can be interpreted as the sum of two main components: a *deterministic* component that is concentrated on a discrete set of frequencies, and a *stochastic* component that has a broadband characteristics. The deterministic –or sinusoidal– component normally corresponds to the main modes of vibration of the system. The stochastic residual accounts for the energy produced by the excitation mechanism which is not turned into stationary vibrations by the system, and for any other energy component that is not sinusoidal.

As an example, consider the sound of a wind instrument: the deterministic signal results from self-sustained oscillations inside the bore, while the residual noisy signal is generated by the turbulent flow components due to air passing through narrow apertures inside the instrument. Similar considerations apply to other classes of instruments, as well as to voice sounds, and even to non-musical sounds.

In the remainder of this section we discuss the modeling of the deterministic sound signal and introduce the main concepts of additive synthesis. Later on, in section 2.4.4 we will address the problem of including the stochastic component into the additive model.

2.4.2.1 Deterministic signal component

The term *deterministic* signal means in general any signal that is not noise. The class of deterministic signals that we consider here is restricted to sums of sinusoidal components with varying amplitude and frequency. Amplitude and frequency variations can be noticed e.g. in sound attacks: some partials that are relevant in the attack can disappear in the stationary part. In general, the frequencies can have arbitrary distributions: for quasi-periodic sounds the frequencies are approximately harmonic components (integer multiples of a common fundamental frequency), while for non-harmonic sounds (such as that of a bell) they have non-integer ratios.

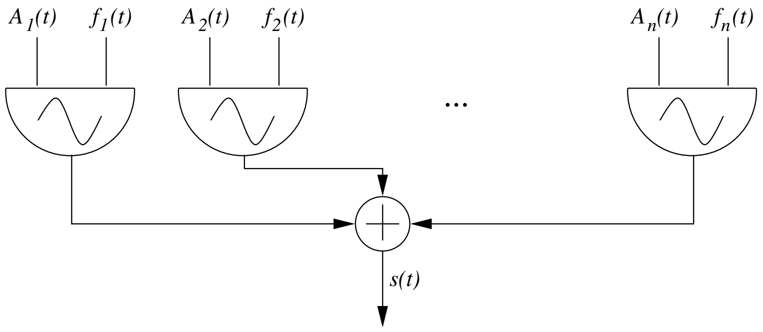


Figure 2.9: Sum of sinusoidal oscillators with time-varying amplitudes and frequencies.

The deterministic part of a discrete-time sound signal can be represented by the sinusoidal model of sect. 2.4.1. The equation is

$$s_s[n] = \sum_k a_k[n]\, cos(\phi_k[n]) \qquad \phi_k[n] = 2\pi\, f_k[n] T_s + \phi_k[n-1]\ . \tag{2.20}$$

This equation has a great generality and can be used to faithfully reproduce many types of sound, especially in a "synthesis-by-analysis" framework (that we discuss in section 2.4.3 below). However, as already noted, it discards completely the noisy components that are always present in real signals. Another drawback of equation (2.20) is that it needs an extremely large number of control parameters: for each note that we want to reproduce, we need to provide the amplitude and frequency envelopes for all the partials. Moreover, the envelopes for a single note are not fixed, but depend in general on the intensity.

On the other hand, additive synthesis provides a very intuitive sound representation, and this is one of the reasons why it has been one of the earliest popular synthesis techniques in computer music.[1] Moreover, sound transformations performed on the parameters of the additive representation (e.g., time-scale modifications) are perceptually very robust.

2.4.2.2 Time- and frequency-domain implementations

Additive synthesis with equation (2.20) can be implemented either in the time domain or in the frequency domain. The more traditional time-domain implementation uses the digital sinusoidal oscillator in wavetable or recursive form, as discussed in section 2.2.1.1. The instantaneous amplitude and the instantaneous radian frequency of a particular partial are obtained by linear interpolation, as discussed previously. Figure 2.9 provides a block diagram of such a time-domain implementation.

M-2.11

Use the sinusoidal oscillator realized in M-2.3 to synthesize a sum of two sinusoids.

M-2.11 Solution

```
%%% headers %%%
%[...]
```

[1]Some composers have even used additive synthesis as a compositional metaphor, in which sound spectra are reinterpreted as harmonic structures.

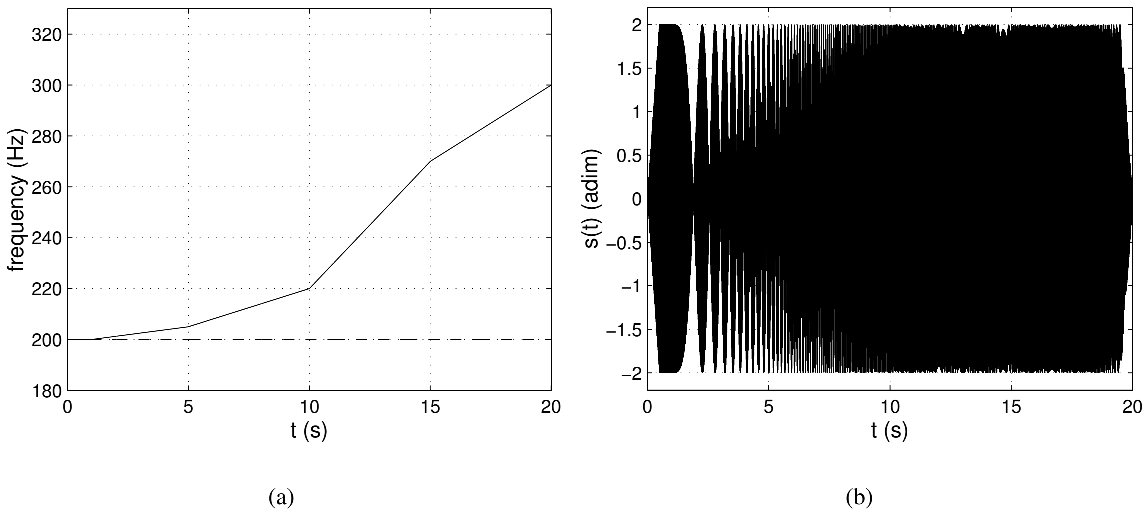


Figure 2.10: Beating effect: (a) frequency envelopes (`f1` dashed line, `f2` solid line) and (b) envelope of the resulting signal.

```
%%% define controls %%%
a=envgen([0,.5,5,10,15,19.5,20],[0,1,1,1,1,1,0]); %fade in/out
f1=envgen([0,20],[200,200]);          %constant freq. envelope
f2=envgen([0,1,5,10,15,20],...        %increasing freq. envelope
          [200,200,205,220,270,300]);

%%% compute sound %%%
s=sinosc(0,a,f1,0)+sinosc(0,a,f2,0);
```

The sinusoidal oscillator controlled in frequency and amplitude is the fundamental building block for time-domain implementations of additive synthesis. Here we employ it to look at the beating phenomenon. We use two oscillators, of which one has constant frequency while the second is given a slowly increasing frequency envelope. Figure 2.10 shows the `f1, f2` control signals and the amplitude envelope of the resulting sound signal: note the beating effect.

In alternative to the time-domain approach, a very efficient implementation of additive synthesis can be developed in the frequency domain, using the inverse FFT. Consider a sinusoid in the time-domain: its STFT is obtained by first multiplying it for a time window $w[n]$ and then performing the Fourier transform. Therefore the transform of the windowed sinusoid is the transform of the window, centered on the frequency of the sinusoid, and multiplied by a complex number whose magnitude and phase are the magnitude and phase of the sine wave:

$$s[n] = A\cos(2\pi f_0 n/F_s + \phi) \quad \Rightarrow \quad \mathcal{F}[w \cdot s](f) = Ae^{i\phi}W(f - f_0). \tag{2.21}$$

If the window $W(f)$ has a sufficiently high sidelobe attenuation, the sinusoid can be generated in the spectral domain by calculating the samples in the main lobe of the window transform, with the

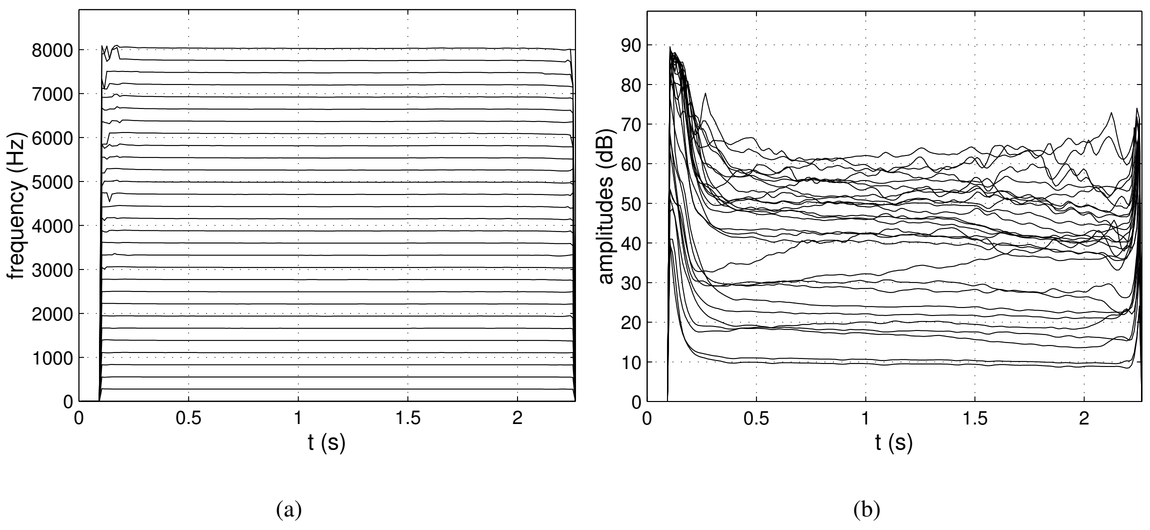


Figure 2.11: Fourier analysis of a saxophone tone: (a) frequency envelopes and (b) amplitude envelopes of the sinusoidal partials, as functions of time.

appropriate magnitude, frequency and phase values. One can then synthesize as many sinusoids as desired, by adding a corresponding number of main lobes in the Fourier domain and performing an IFFT to obtain the resulting time-domain signal in a frame.

By an overlap-and-add process one then obtains the time-varying characteristics of the sound. Note however that, in order for the signal reconstruction to be free of artifacts, the overlap-and-add procedure must be carried out using a window with the property that its shifted copies overlap and add to give a constant. A particularly simple and effective window that satisfies this property is the triangular window.

The FFT-based approach can be convenient with respect to time-domain techniques when a very high number of sinusoidal components must be reproduced: the reason is that the computational costs of this implementation are largely dominated by the cost of the FFT, which does not depend on the number of components. On the other hand, this approach is less flexible than the traditional oscillator bank implementation, especially for the instantaneous control of frequency and magnitude. Note also that the instantaneous phases are not preserved using this method. A final remark concerns the FFT size: in general one wants to have a high frame rate, so that frequencies and magnitudes need not to be interpolated inside a frame. At the same time, large FFT sizes are desirable in order to achieve good frequency resolution and separation of the sinusoidal components. As in every short-time based processes, one has to find a trade-off between time and frequency resolution.

2.4.3 Synthesis by analysis

As already remarked, additive synthesis allows high quality sound reproduction if the amplitude and frequency control envelopes are extracted from Fourier analysis of real sounds. Figure 2.11 shows the result of this kind of analysis, in the case of a saxophone tone. Using these data, additive resynthesis is straightforward.

M-2.12

Assume that the script `sinan` imports two matrices `sinan_freqs` and `sinan_amps` with the partial frequency and amplitude envelopes of an analyzed sound. Resynthesize the sound.

M-2.12 Solution

```
%%% headers %%%
% [...]

%%% define controls %%%
readsan;          %import analysis matrices sinan_freqs and sinan_amps
npart=size(sinan_amps,1);  %number of analyzed partials

%%% compute sound %%%
s=sinosc(...          %generate first partial
        0.5,sinan_amps(1,:),sinan_freqs(1,:),0);
for (i=2:npart) %generate higher partials and sum
  s=s+sinosc(0.5,sinan_amps(i,:),sinan_freqs(i,:),0);
end
```

2.4.3.1 Magnitude and Phase Spectra Computation

The first step of any analysis procedure that tracks frequencies and amplitudes of the sinusoidal components is the frame-by-frame computation of the sound magnitude and phase spectra. This is carried out through short-time Fourier transform. The subsequent tracking procedure will be performed in this spectral domain. The control parameters for the STFT are the window-type and size, the FFT-size, and the frame-rate. These must be set depending on the sound to be processed.

Note that the analysis step is completely independent from the synthesis, therefore the observations made in section 2.4.2.2 about FFT-based implementations (the window must overlap and add to a constant) do not apply here. Good resolution of the spectrum is needed in order to correctly resolve, identify, and track the peaks which correspond to the deterministic component.

If the analyzed sound is almost stationary, long windows (i.e. windows that cover several periods) that have good side-lobe rejection can be used, with a consequent good frequency resolution. Unfortunately most interesting sounds are not stationary and a compromise is required. For harmonic sounds one can scale the actual window size as a function of pitch, thus achieving a constant time-frequency trade-off. For inharmonic sounds the size should be set according to the minimum frequency difference that exists between partials.

The question is now how to perform automatic detection and tracking of the spectral peaks that correspond to sinusoidal components. In section 2.4.3.2 below we present the main guidelines of a general analysis framework, which is summarized in figure 2.12. First, the FFT of a sound frame is computed according to the above discussion. Next, the prominent spectral peaks are detected and incorporated into partial trajectories. If the sound is pseudo-harmonic, a pitch detection step can improve the analysis by providing information about the fundamental frequency information, and can also be used to choose the size of the analysis window.

Such a scheme is only one of the possible approaches that can be used to attack the problem. Hidden Markov Models (HMMs) are another one: a HMM can optimize groups of peaks trajectories according to given criteria, such as frequency continuity. This type of approach might be very valuable for tracking partials in polyphonic sounds and complex inharmonic tones.

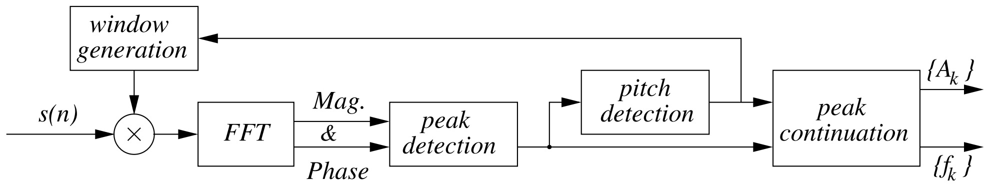


Figure 2.12: Block diagram of the sinusoid tracking process, where $s[n]$ is the analyzed sound signal and $A_k,\ f_k$ are the estimated amplitude and frequency of the kth partial in the current analysis frame.

2.4.3.2 A sinusoid tracking procedure

We now discuss in more detail the analysis steps depicted in figure 2.12.

Peak detection. The first one is detection of the most prominent frequency peaks (i.e., local maxima in the magnitude spectrum) in the current analysis frame. Real sounds are not periodic, do not have clearly spaced and defined spectral peaks, exhibit interactions between components. Therefore, the best one can do at this point is to detect as many peaks as possible and postpone to later analysis steps the decision of which ones actually correspond to sinusoidal components. The peaks are then searched by only imposing two minimal constraints: they have to lie within a given frequency range, and above a given magnitude threshold. The detection of very soft peaks is hard: they have little resolution, and measurements are very sensitive to transformations because as soon as modifications are applied to the analysis data, parts of the sound that could not be heard in the original can become audible. Having a very clean sound with the maximum dynamic range, the magnitude threshold can be set to the amplitude of the background noise floor. In order to gain better resolution in the high frequency range, the sound may be pre-processed to introduce preemphasis, which has then to be compensated later on before the resynthesis.

Pitch detection. After peak detection, many procedures can be used to decide whether a peak belongs to a sinusoidal partial or not. One possible strategy is to measure how close the peak shape is to the ideal sinusoidal peak (recall what we said about the transform of a windowed sinusoid and in particular equation (2.21). A second valuable source of additional information is pitch. If a fundamental frequency is actually present, it can be exploited in two ways. First, it helps the tracking of partials. Second, the size of the analysis window can be set according to the estimated pitch in order to keep the number of periods-per-frame constant, therefore achieving the best possible time-frequency trade-off (this is an example of a *pitch-synchronous* analysis). There are many possible pitch detection strategies, which will be presented in the next chapter.

Peak continuation. A third and fundamental strategy for peak selection is to implement some sort of peak *continuation* algorithm. The basic idea is that a set of "guides" advance in time and follow appropriate frequency peaks (according to specified constraints that we discuss in the next paragraph) forming trajectories out of them. A guide is therefore an abstract entity which is used by the algorithm to create the trajectories, and the trajectories are the actual result of the peak continuation process. The guides are turned on, advanced, and finally turned off during the continuation algorithm, and their instantaneous state (frequency and magnitude) is continuously updated during the process. If the analyzed sound is harmonic and a fundamental has been estimated, then the guides are created at the beginning of the analysis, with frequencies set according to the estimated harmonic series. When no harmonic structure can be estimated, each guide is created when the first available peak is found. In the successive analysis frames, the guides modify their status depending on the last peak values. This past information is particularly relevant when the sound is not harmonic, or when the harmonics are not locked to each other and we cannot rely on the fundamental as a strong reference for all the harmonics.

The main constraints used to assign guides to spectral peaks are as follows. A peak is assigned to the guide that is closest to it and that is within an assigned frequency deviation. If a guide does not find a match, the corresponding trajectory can be turned off, and if a continuation peak is not found for a given amount of time the guide is killed. New guides and trajectories can be created starting from peaks of the current frame that have high magnitude and are not "claimed" by any of the existing trajectories. After a certain number of analysis frames, the algorithm can look at the trajectories created so far and adopt corrections: in particular, short trajectories can be deleted, and small gaps in longer trajectories can be filled by interpolating between the values of the gap edges.

One final refinement to this process can be added by noting that the sound attack is usually highly non-stationary and noisy, and the peak search is consequently difficult in this part. Therefore it is customary to perform the whole procedure backwards in time, starting from the end of the sound (which is usually a more stable part). When the attack is reached, a lot of relevant information has already been gained and non-relevant peaks can be evaluated and/or rejected.

2.4.4 "Sines-plus-noise" models

At the beginning of our discussion on additive modeling, we remarked that the spectral energy of the sound signals has a *deterministic* component that is concentrated on a discrete set of frequencies, and a *stochastic* component that has a broadband characteritics. So far we have discussed the problem of modeling the deterministic –or sinusoidal– component. Now we have to include the stochastic component into the model.

A sinusoidal representation may in principle be used also to simulate noise, since noise consists of sinusoids at every frequency within the band limits. It is clear however that such a representation would be computationally very demanding. Moreover it would not be a *flexible* sound representation. Therefore the most convenient sound model is of the form

$$s[n] = s_s[n] + e[n] \tag{2.22}$$

where $s_s[n]$ represent the deterministic part (eq. 2.20 and $e[n]$ represents the stochastic component and is modeled separately from the deterministic part.

2.4.4.1 Stochastic analysis

The most straightforward approach to estimation of the stochastic component is through *subtraction* of the deterministic component from the original signal. Subtraction can be performed either in the time domain or in the frequency domain. Time domain subtraction must be done while preserving the phases of the original sound, and instantaneous phase preservation can be computationally very expensive. One the other hand, frequency-domain subtraction does not require phase preservation. However, time-domain subtraction provides much better results, and is usually favored despite the higher computational costs. For this reason we choose to examine time-domain subtraction in the remainder of this section. Figure 2.13 provides a block diagram.

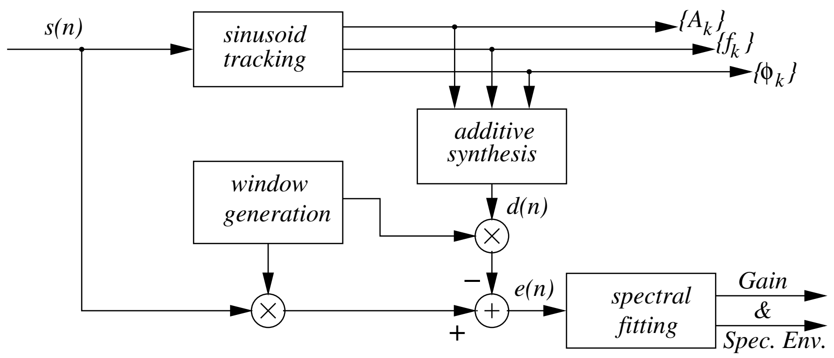


Figure 2.13: Block diagram of the stochastic analysis and modeling process, where $s[n]$ is the analyzed sound signal and $A_k,\ f_k,\ \phi_k$ are the estimated amplitude, frequency, and phase of the kth partial in the current analysis frame.

Suppose that the deterministic component has been estimated in a given analysis frame, using for instance the general scheme described in section 2.4.3 (note however that in this case the analysis should be improved in order to provide estimates of the instantaneous phases as well). Then the first step in the subtraction process is the time-domain resynthesis of the deterministic component with the estimated parameters. This should be done by properly interpolating amplitude, frequency, and phase values in order to avoid artifacts in the resynthesized signal. The actual subtraction can be performed as

$$e[n] = w[n] \cdot [s[n] - d[n]], \qquad n = 0, 1, \ldots, N-1, \tag{2.23}$$

where $s[n]$ is the original sound signal and $d[n]$ is the re-synthesized deterministic part. The difference $(s - d)$ is multiplied by an analysis window w of size N, which deserves some discussion.

We have seen in 2.4.3 that high frequency resolution is needed for the deterministic part, and for this reason long analysis windows are used for its estimation. On the other hand, good time resolution is more important for the stochastic part of the signal, especially in sound attacks, while frequency resolution is not a major issue for noise analysis. A way to obtain good resolutions for both the components is to use two different analysis windows. Therefore w in equation (2.23) is not in general the same window used to estimate $d[n]$, and the size N is in general small.

Once the subtraction has been performed, there is one more step than can be used to improve the analysis, namely, test can be performed on the estimated residual in order to assess how good the analysis was. If the spectrum of the residual still contains some partials, then the analysis of the deterministic component has not been performed accurately and the sound should be re-analyzed until the residual is free of deterministic components. Ideally the residual should be as close as possible to a stochastic signal, therefore one possible test is a measure of correlation of the residual samples.[2]

2.4.4.2 Stochastic modeling

The assumption that the residual is a stochastic signal implies that it is fully described by its amplitude and its spectral envelope characteristics. Information on the instantaneous phase is not necessary. Based on these considerations, a frame of the stochastic residual can be completely characterized by

[2] Note that if the analyzed sound has not been recorded in silent and anechoic settings the residual will contain not only the stochastic part of the sound, but also reverberation and/or background noise.

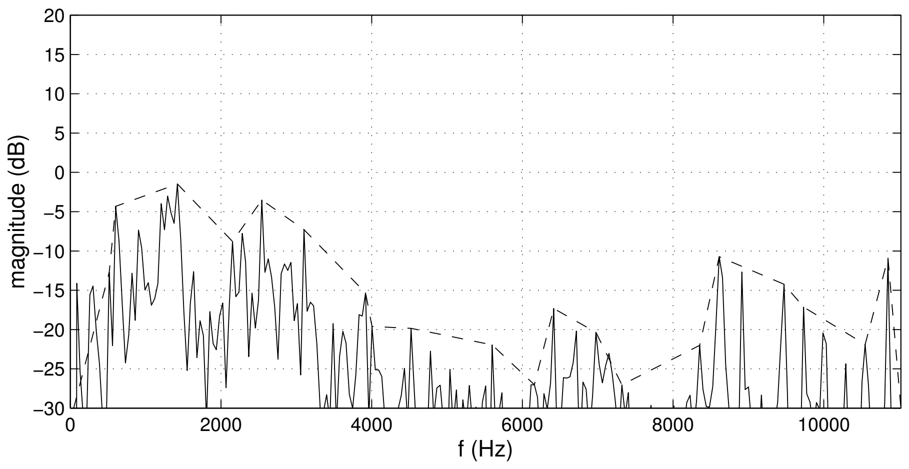


Figure 2.14: Example of residual magnitude spectrum (solid line) and its line-segment approximation (dashed line), in an analysis frame. The analyzed sound signal is the same saxophone tone used in figure 2.11.

a filter that models the amplitude and general frequency characteristics of the residual. The representation of the residual for the overall sound will then be a time-varying filter.

Within a given frame we therefore assume that $e(t)$ can be modeled as

$$E(f) = H(t) \cdot U(f), \tag{2.24}$$

where U is white noise and H is the frequency response of filter whose coefficients vary on a frame-by-frame basis. The stochastic modeling step is summarized in the last block of figure 2.13.

The filter design problem can be solved using different strategies. One approach that is often adopted uses some sort of curve fitting (line-segment approximation, spline interpolation, least squares approximation, and so on) of the magnitude spectrum of e in an analysis frame. As an example, line-segment approximation can be obtained by stepping through the magnitude spectrum, finding local maxima at each step, and connecting the maxima with straight lines. This procedure can approximate the spectral envelope with reasonable accuracy, depending on the number of points, which in turn can be set depending on the sound complexity. See figure 2.14 for an example.

Another possible approach to the filter design problem is Linear Predictive Coding (LPC), which is a popular technique in speech processing. However in this context curve fitting procedure on the noise spectrum (e.g., line-segment approximation) are usually considered to be more flexible approaches and are preferred to LPC. We will return on LPC in section 2.4.7.2.

The next question is how to implement the estimated time-varying filter in the resynthesis step.

2.4.4.3 Resynthesis and modifications

Figure 2.15 shows the block diagram of the synthesis process. The deterministic signal, i.e., the sinusoidal component, results from the magnitude and frequency trajectories, or their transformation, by generating a sine wave for each trajectory (additive synthesis). As we have seen, this can either be implemented in the time domain with the traditional oscillator bank method or in the frequency domain using the inverse-FFT approach.

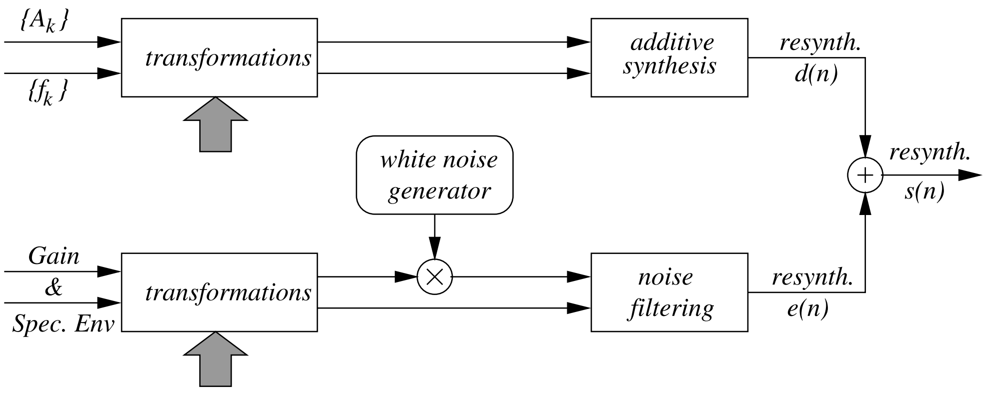


Figure 2.15: Block diagram of the sines-plus-noise synthesis process.

Concerning the stochastic component, a frequency-domain implementation is usually preferred to a direct implementation of the time-domain convolution (2.24), due to its computational efficiency[3] and flexibility. In each frame, the stochastic signal is generated by an inverse-FFT of the spectral envelopes. Similarly to what we have seen for the deterministic synthesis in section 2.4.2.2, the time-varying characteristics of the stochastic signal is then obtained using an overlap-and-add process.

In order to perform the IFFT, a magnitude and a phase responses have to be generated starting from the estimated frequency envelope. Generation of the magnitude spectrum is straightforwadly obtained by first linearly interpolating the spectral envelope to a curve with half the length of the FFT-size, and then multiplying it by a gain that corresponds to the average magnitude extracted in the analysis. The estimated spectral envelope gives no information on the phase response. However, since the phase response of noise is noise, a phase response can be created from scratch using a random signal generator. In order to avoid periodicities at the frame rate, new random values should be generated at every frame.

The sines-plus-noise representation is well suited for modification purposes.

- By only working on the deterministic representation and modifying the amplitude-frequency pairs or the original sound partials, many kinds of frequency and magnitude transformations can be obtained. As an example, partials can be transposed in frequency. It is also possible to decouple the sinusoidal frequencies from their amplitude, obtaining pitch-shift effects that preserve the formant structure.

- Time-stretching transformations can obtained by resampling the analysis points in time, thus slowing down or speeding up the sound while maintaining pitch and formant structure. Given the stochastic model that we are using, the noise remains noise and faithful signal resynthesis is possible even with extreme stretching parameters.

- By acting on the relative amplitude of the two components, interesting effects can be obtained in which either the deterministic or the stochastic parts are emphasized. As an example, the amount of “breathiness” of a voiced sound or a wind instrument tone can be adjusted in this way. One must keep in mind however that, when different transformations are applied to the

[3] In fact, by using a frequency-domain implementation for both the deterministic and the stochastic synthesis one can add the two spectra and resynthesize both the components at the cost of a sigle IFFT per frame.

two representations, the deterministic and stochastic components in the resulting signal may not be perceived as a single sound event anymore.

- Sound morphing (or *cross-synthesis* transformations can be obtained by interpolating data from two or more analysis files. This transformations are particularly effective in the case of quasi-harmonic sounds with smooth parameter curves.

2.4.5 Sinusoidal description of transients

So far we have seen how to extend the sinusoidal model by using a "sines-plus-noise" approach that explicitly describes the residual as slowly varying filtered white noise. Although this technique is very powerful, transients do not fit well into a filtered noise description, because they lose sharpness and are smeared. This consideration motivates us to handle transients separately.

One straightforward approach, that is sometimes used, is removing transient regions from the residual, performing the sines-plus-noise analysis, and adding the transients back into the signal. This approach obviously requires memory where the sampled transients must be stored, but since the transient residuals remain largely invariant throughout most of the range of an instrument, only a few residuals are needed in order to cover all the sounds of a single instrument. Although this approach works well, it is not flexible because there is no model for the transients. In addition, identifying transients as everything that is neither sinusoidal nor transient is not entirely correct. Therefore we look for a suitable transient model, that can be embedded in the additive description to obtain a "sines-plus-transients-plus-noise" representation.

2.4.5.1 The DCT domain

In the following we adopt a further modified version of the additive sound representation (2.20), in which the sound transients are explicitly modeled by an additional signal:

$$s[n] = s_s[n] + e_t[n] + e_r[n], \tag{2.25}$$

where $s_s[n]$ is the sinusoidal component, $e_t[n]$ is the signal associated to transients and $e_r[n]$ is the noisy residual. The transient model is based on a main undelying idea: we have seen that a slowly varying sinusoidal signal is impulsive in the frequency domain, and sinusoidal models perform short-time Fourier analysis in order to track slowly varying spectral peaks (the tips of the impulsive signals) over time. Transients are very much dual to sinusoidal components: they are impulsive in the time domain, and consequently they must be oscillatory in the frequency domain. Therefore, although transient cannot be tracked by a short-time analysis (because their STFT will not contain meaningful peaks), we can track them by performing sinusoidal modeling in a properly chosen frequency domain. The mapping that we choose to use is the one provided by the discrete cosine transform (DCT):

$$S[k] = \beta[k] \sum_{n=0}^{N-1} s[n] \cos\left[\frac{(2n-1)k\pi}{2N}\right], \quad \text{for } n,k = 0,1,\ldots,N-1, \tag{2.26}$$

where $\beta[1] = \sqrt{1/N}$ and $\beta[k] = \sqrt{2/N}$ otherwise. From equation (2.26) one can see that an ideal impulse $\delta[n - n_0]$ (i.e., a Kronecker delta function centered in n_0) is transformed into a cosine whose frequency increases with n_0. Figure 2.16(a) shows a more realistic transient signal, a one-sided exponentially decaying sine wave. Figure 2.16(b) shows the DCT of the transient signal: a slowly varying sinusoid. This considerations suggest that the time-frequency duality can be exploited to develop a transient model: the same kind of parameters that characterize the sinusoidal components of a signal can also characterize the transient components of a signal, although in a different domain.

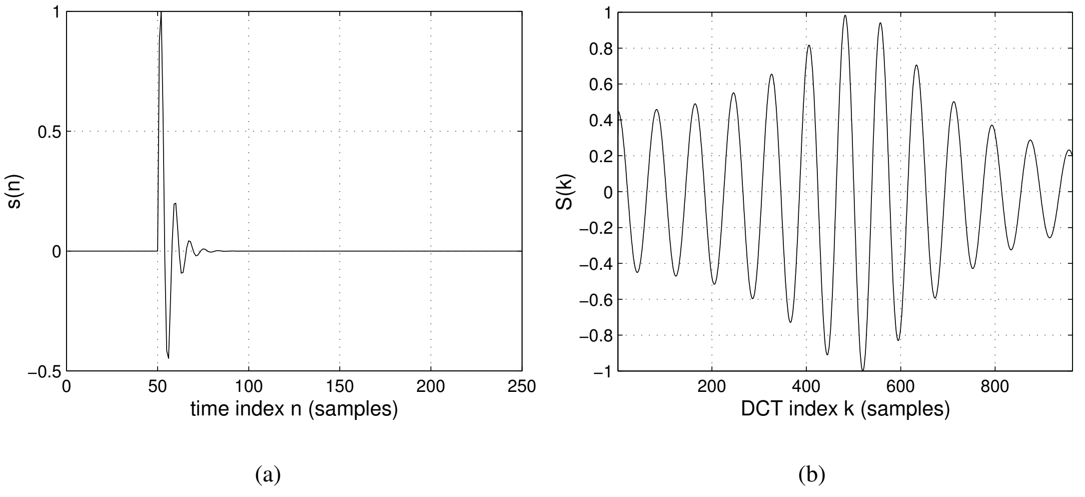


Figure 2.16: Example of DCT mapping: (a) an impulsive transient (an exponentially decaying sinusoid) and (b) its DCT as a slowly varying sinusoid.

2.4.5.2 Transient analysis and modeling

Having transformed the transient into the DCT domain, the most natural way to proceed is performing sinusoidal modeling in this domain: STFT analysis of the DCT-domain signal can be used to find meaningful peaks, and then the signal can be resynthesized in the DCT domain and back-transformed to the time domain with an inverse DCT transform (IDCT). This process is shown in figure 2.17. We now discuss the main steps involved in this block diagram.

First the input signal $s[n]$ is divided into non-overlapping blocks in which DCT analysis will be performed. The block length should be chosen so that a transient appears as "short", therefore large block sizes (e.g., 1 s) are usually chosen. The block DCT is followed by a sinusoidal analysis/modeling process which is identical to what we have seen is section 2.4.3. The analysis can optionally embed some information about transient location withing the block: there are many possible transient detection strategies, which we do not want to discuss here. Also, the analysis can perform better if the sinusoid tracking procedure starts from the end of the DCT-domain signal and moves backwards toward the beginning, because the beginning of a DCT frame is usually spectrally rich and this can deteriorate the performance of the analysis (similar considerations were done in section 2.4.3 when discussing sinusoid tracking in the time domain).

The analysis yields parameters that correspond to slowly varying sinusoids in the DCT domain: each transient is associated to a triplet $\{A_k, f_k, \phi_k\}$, amplitude, frequency, and phase of the kth "partial" in each STFT analysis frame within a DCT block. By recalling the properties of the DCT one can see that f_k correspond to onset locations, A_k is the amplitude of the time-domain signal also, and ϕ_k is related to the time direction (positive or negative) in which the transient evolves. Resynthesis of the transients is then performed using these parameters to reconstruct the sinusoids in the DCT domain. Finally an inverse discrete cosine transform (IDCT) on each of the reconstructed signals is used to obtain the transients in each time-domain block, and the blocks are concatenated to obtain the transients for the entire signal.

It is relatively straightforward to implement a "fast transient reconstruction" algorithm. Without

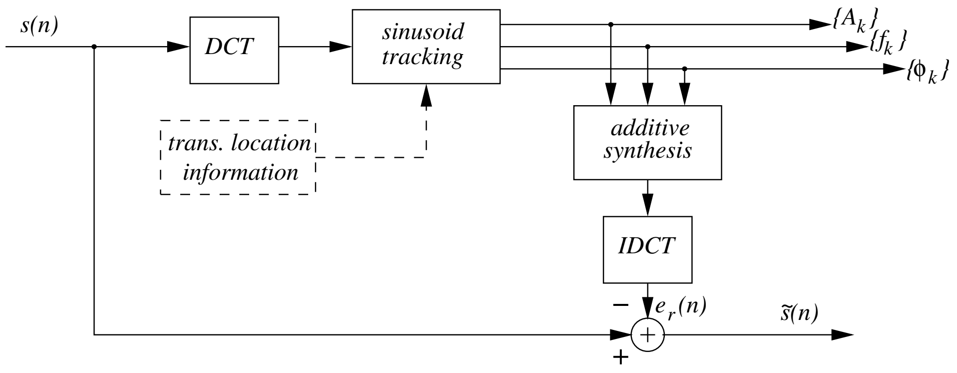


Figure 2.17: Block diagram of the transient analysis and modeling process, where $s[n]$ is the analyzed sound signal and $A_k,\ f_k,\ \phi_k$ are the estimated amplitude, frequency, and phase of the kth DCT-transformed transient in the current analysis frame.

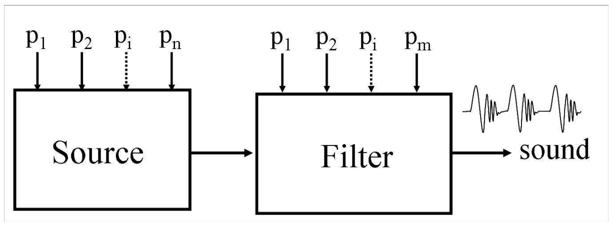


Figure 2.18: Source filter model.

entering the details, we just note that the whole procedure can be reformulated using FFT transformations only: in fact one could verify that the DCT can be implemented using an FFT block plus some post-processing (multiplication of the real and imaginary parts of the FFT by appropriate cosinusoidal and sinusoidal signals followed by a sum of the two parts). Furthermore, this kind of approach naturally leads to a FFT-based implementation of the additive synthesis step (see section 2.4.2.2).

One nice property of this transient modeling approach is that it fits well within the sines-plus-noise analysis that we have examined in the previous sections. The processing block depicted in figure 2.17 returns an output signal $\tilde{s}[n]$ in which the transient components e_t have been removed by subtraction: this signal can be used as the input to the sines-plus-noise analysis, in which the remaining components (deterministic and stochastic) will be analyzed and modeled. From the implementation viewpoint, one advantage is that the core components of the transient-modeling algorithm (sinusoid tracking and additive resynthesis) are identical to those used for the deterministic model. Therefore the same processing blocks can be used in the two stages, although working on different domains.

2.4.6 Source-filter models

Some sources can be modeled as an exciter, characterized by a spectrally rich signal, and a resonator, described by a linear system, connected in a feed-forward relationship (fig. 2.18). An example is the voice, where the periodic pulses or random fluctuations produced by the vocal folds are filtered by the vocal tract, that shapes the spectral envelope. The vowel quality and the voice color greatly depends on the resonance regions of the filter, called formants.

If the system is linear and time-invariant (at least on a short time scale), it can be described by the filter with transfer function given by

$$H(z) = \frac{B(z)}{A(z)} = \frac{\sum_i b_i z^{-i}}{\sum_k a_k z^{-k}} \tag{2.27}$$

that can be computed by a difference equation

$$s_f[n] = \sum_i b_i u[n-i] - \sum_k a_k s_f[n-k] \; . \tag{2.28}$$

where a_k e b_i are the filter coefficients and $u[n]$ e $s_f[n]$ are input and output signals. The model is also represented by the convolution of the source $u[n]$ with the impulse response of the filter

$$s_f[n] = (u * h)[n] \overset{\triangle}{=} \sum_k h(n-k)u(k) \; . \tag{2.29}$$

Digital signal processing theory gives us the tools to design the filter structure and to estimate the filter coefficients in order to obtain a desired frequency response. This model combines the spectral fine structure (spectral lines, broadband or narrowband noise, etc.) of the input with the spectral envelope shaping properties of the filter:

$$S_f(f) = U(f)\,H(f) \; . \tag{2.30}$$

Therefore, it is possible to control and modify separately the pitch from the formant structure of a speech sound. In computer music this model is called *subtractive synthesis*. If the filter is static, the temporal features of the input signal are maintained. If, conversely, the filter coefficients are varied, the frequency response changes. As a consequence, the output will be a combination of temporal variations of the input and of the filter (*cross-synthesis*).

If we make some simplifying hypothesis about the input, it is possible to estimate both the parameters of the source and the filter of a given sound. The most common procedure is linear predictive coding (LPC) which assumes that the source is either a periodic impulse train or white noise, and that the filter is all pole (i.e., no zeros). LPC is widely used for speech synthesis and modification.

2.4.6.1 Basic elements

Every spectrally rich signal can be used as input to a subtractive synthesis algorithm. The simplest one for periodic sounds is the impulse train generator which produces a sequence of unit impulses, spaced by the desired fundamental perion. Another simple generator for stochastic sources is the random noise generator, which produces a flat spectrum noise.

M-2.13

Write a function `noisegen(t0,amp)` that realizes the noise generator, and a function `buzz(t0,amp,f)` that realizes the impulse generator. The parameters `(t0,amp,f)` are initial time, amplitude envelope, and frequency envelope, respectively.

M-2.13 Solution

Hint for the noise generator: simply use the function `rand()`.

Hint for the impulse generator: use additive synthesis, and sum up all the harmonic components $\cos(2k\pi f_0 t)$ $(k \in \mathbb{N})$ of the fundamental frequency f_0, up to the Nyquist frequency $F_s/2$. Figure 2.19 shows the resulting signal, in the time and frequency domains

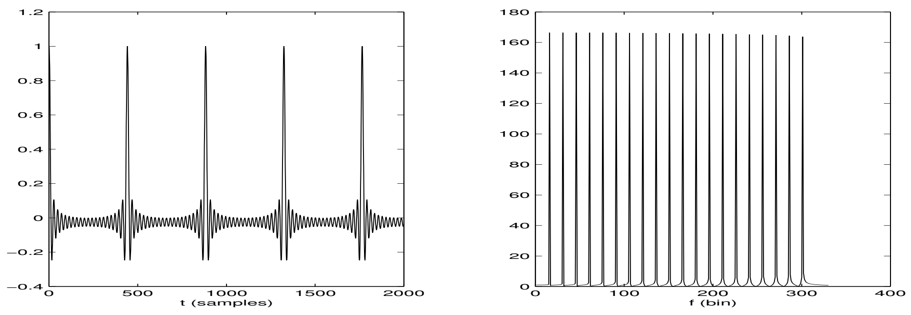


Figure 2.19: Impulse generator.

2.4.6.1.1 Simple low pass filter: first order IIR filter A simple single low pass filter is described by the equations

$$\begin{aligned} y[n] &= bx[n] + ay[n-1] \qquad 0 < a < 1, b > 0 && (2.31) \\ H(z) &= \frac{b}{1 - az^{-1}} && (2.32) \end{aligned}$$

The filter has a pole at $z = r$ and and a maximum gain of $b(1-a)$. When $a < 1$, we have a low pass filter and its selectivity increase when a tends to 1. The 3db bandwidth f_{3dB} is given by

$$f_{3dB} = \frac{1-a}{2\pi F_s} \tag{2.33}$$

The impulse response is a decreasing exponential with time constant $\tau = -1/(\ln(a)F_s)$ seconds. The time taken to decrease the amplitude below 1% (i.e. 40 dB) of the initial value is given by

$$\tau_{40dB} = \frac{1}{F_s}\frac{\log(0.01)}{\log(a)} = \frac{2}{F_s}\log(\frac{1}{a}) \tag{2.34}$$

and is referred to as the 40-dB time constant.

2.4.6.1.2 Digital resonator: second order IIR filter An important filter is the second order IIR filter that is very often used for modeling resonances. It is described by:

$$H(z) = \frac{b_0}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{2.35}$$

where B is the bandwidth and f_c is the center frequency of the resonance.The filter coefficient can be derived by these (approximate) equations:

$$a_1 = -2r\cos(\omega_c), \;\; a_2 = r^2, \;\; b_0 = (1-r)\sqrt{1 - 2r\cos(2\omega_c) + r^2}, \tag{2.36}$$

where the auxiliary parameters r and ω_c are defined as $r = e^{\frac{-\pi B}{F_s}}$ e $\omega_c = 2\pi f_c/F_s$ and correspond to the magnitude and phase of the complex conjugate poles of the transfer function. The parameter b_0, gain normalization factor, is computed normalizing the magnitude at the resonance frequency, i.e. $|H(\omega_c)| = 1$. The impulse response is a sinusoidal function with exponentially decreasing amplitude

$$h([n] = \frac{b_0}{\sin\omega_c} r_n \sin(\omega_c n + \omega_c)$$

The 40-dB time constant is given by (see eq. 2.34)

$$\tau_{40dB} = \frac{1}{F_s}\frac{\log(0.01)}{\log(r)} = \frac{2}{F_s}\log(\frac{1}{r}) \qquad (2.37)$$

M-2.14

Write a function `baIIR2(fc,B)` that computes the coefficients of the 2nd order resonant filter given the center frequency f_c and the bandwidth B.

M-2.14 Solution

```
function [b,a]=baIIR2(fc,B); %computes coeff. a,b of II order cells
                             %(the no. of cells to be computed depends
                             %on the length of the vectors fc,B)
global Fs; global Fc;
nfilters=length(fc);         %no. of cells to be computed

r=exp(-(pi.*B)/Fs) a1=-(2*r.*cos(2*pi*fc/Fs))' a2=r'.^2
a0=ones(nfilters,1)

b0=(1-r).*sqrt(1-2.*r.*cos(2*2*pi.*fc/Fs)+r.*r); b0=b0';

a=[a0 a1 a2]; b=[b0 zeros(nfilters,1) zeros(nfilters,1)];
```

Note that we have followed the Octave/Matlab convention in defining the coefficients `b,a`. See the help for the function `filter(b,a,in)`

By using the sources and filter above described, it is possible to experiment the effect of subtractive synthesis with constant parameters.

M-2.15

Using the functions `buzz` and `baIIR2`, realize a parallel formant synthesizer. Use 3 2nd order IIR cells, corresponding to the first 3 vowel formant frequencies.

M-2.15 Solution

```
%%% headers %%%
%[...]

%%% define controls %%%
amp=envgen([0,.2,1,1.8,2],[0,1,.8,1,0],'linear');  % amp. envelope
f=envgen([0,.2,1.8,2],[200,250,250,200],'linear'); % pitch envelope
f=f+max(f)*0.05*...                                % add vibrato
  sin(2*pi*5*(SpF/Fs)*[0:length(f)-1]).*hanning(length(f))';

[b_i,a_i]=baIIR2([300 2400 3000],[200 200 500]); %spec. envelope /i/
[b_a,a_a]=baIIR2([700 1200 2500],[200 300 500]); %spec. envelope /a/
[b_e,a_e]=baIIR2([570 1950 3000],[100 100 800]); %spec. envelope /e/

%%% compute sound %%%
s=buzz(0,amp,f);   %impulse source
```

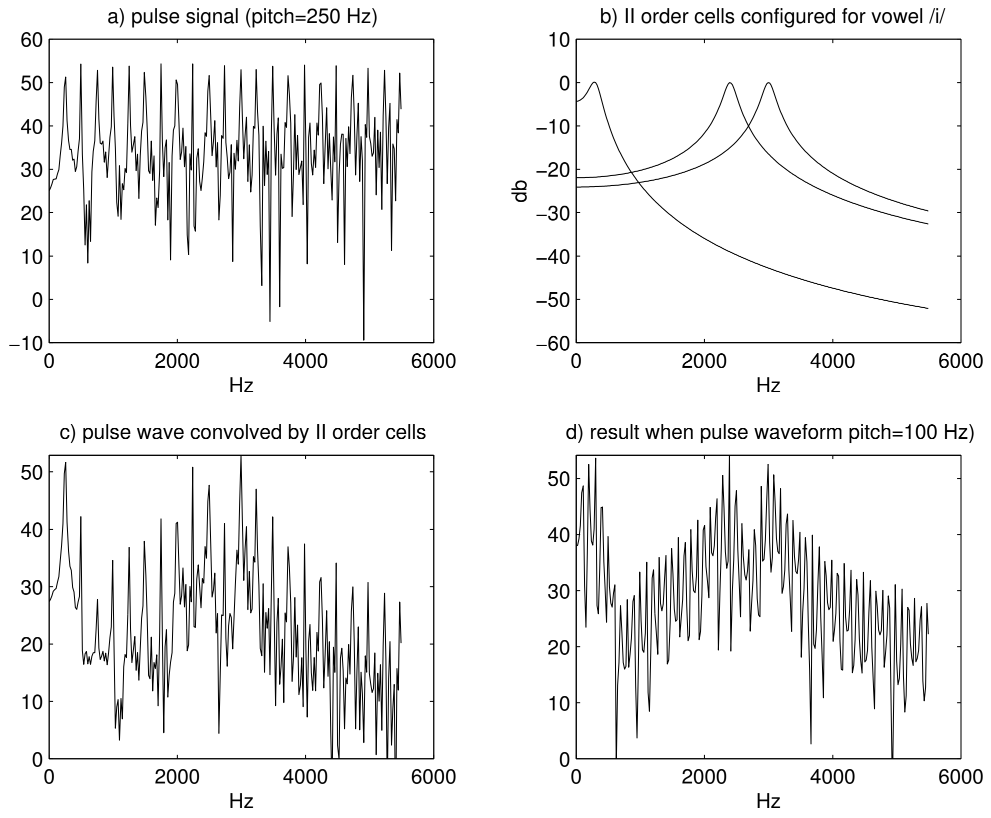


Figure 2.20: Spectrum of the original source signal, the spectral envelope defined by the filtering elements, and the spectrum of the final signal for two different pitches when a paralled synthesis structure is used.

```
si=filter(b_i(1,:),a_i(1,:),s)+...    %synthesize /i/
   filter(b_i(2,:),a_i(2,:),s)+filter(b_i(3,:),a_i(3,:),s);
sa=filter(b_a(1,:),a_a(1,:),s)+...    %synthesize /a/
   filter(b_a(2,:),a_a(2,:),s)+filter(b_a(3,:),a_a(3,:),s);
se=filter_e(b(1,:),a_e(1,:),s)+...    %synthesize /e/
     filter(b_e(2,:),a_e(2,:),s)+filter(b_e(3,:),a_e(3,:),s);
```

Note the use of the `filter` function. Figure 2.20 shows the spectrum of the original source signal, the spectral envelope defined by the filtering elements, and the spectrum of the final signal for two different pitches

2.4.6.1.3 Parametric filters Parametric filters are used quite widely in the audio industry because of their ability to amplify or dampen specific frequency components of a signal. Traditionally, these filters have been designed using analog components, however, their digital counterparts are very simple and efficient to implement. The filters bandwidth, center frequency and gain can be calculated using a few basic formulas to calculate the four required coefficients.

Parametric filters are really just second order IIR filters and have a frequency response containing a single peak or notch at a given frequency f_c. The gain at all other frequencies is roughly unity.

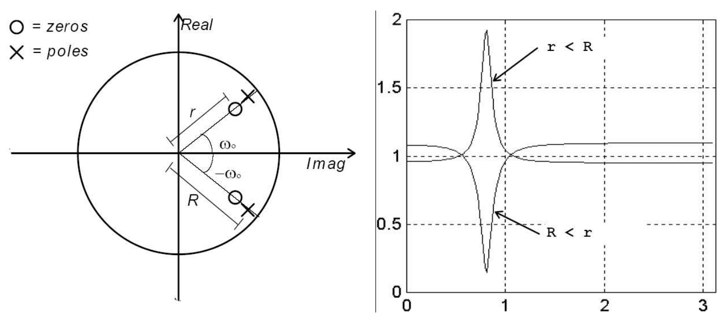


Figure 2.21: Models for speech production: (a) general model and (b) simplified

These filters require less computational power than higher order FIR and IIR filters and the amount of computational power required to calculate coefficients is minimal compared to higher order filters. Heres how the filter works : a conjugate pair of poles and a conjugate pair of zeros are arranged along a straight line from the origin of the Z-plane as shown in figure 2.21(a), where $\omega_0 = 2\pi f_c/F_S$ is the pole argument. If the poles are closer to the origin than the zeros (i.e. $R < r$), the resulting filter will be a notch filter. On the other hand (i.e. $R > r$), the zeros are closer to the origin, the resulting filter will be a peak filter. Figure 2.21(b) contains the frequency responses of both cases. The strength of the boost or cut of the respective peak and notch is determined by the closeness of r and R. Also, the width of the peak or cut is determined by the closeness or r and R to the unit circle.

The transfer function is given by

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{2.38}$$

where

$$a_0 = 1 \quad a_1 = -2R\cos(\omega_c) \quad a_2 = R^2 \tag{2.39}$$

$$b_0 = 1 \quad b_1 = -2r\cos(\omega_c) \quad b_2 = r^2 \tag{2.40}$$

2.4.6.1.4 Filters for producing periodic signals A special case is when the filter features a long delay as in

$$s_f[n] = \beta u[n] - \alpha s_f[n - N_p]\ . \tag{2.41}$$

This is a comb type filter featuring frequency resonances multiple of a fundamental $f_p = F_s/N_p$. The impulse response is a periodic repetition of impulses every $N - p$ sample.

If initial values are set for the whole delay line, for example random values, all the frequency components that do not coincide with resonance frequencies are progressively filtered out until a harmonic sound is left. If there is attenuation ($\alpha < 1$) the sound will have a decreasing envelope. Substituting α and/or β with filters, the sound decay time will depend on frequency. For example if α is smaller at higher frequencies, the upper harmonics will decay faster than the lower ones. We can thus obtain simple sound simulations of the plucked strings, where the delay line serves to establish oscillations. This method is suitable to model sounds produced by a brief excitation of a resonator,

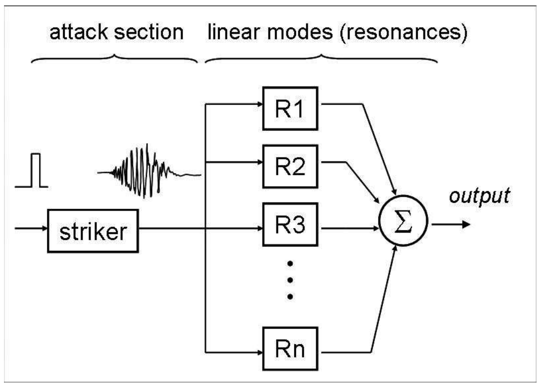


Figure 2.22: Parallel structure of digital resonators Ri for strike simulation. The parameters are striker characteristics and resonator constants.

where the latter establishes the periodicity, and the interaction between exciter and resonator can be assumed to be feedforward. This method is called *long-term prediction* or *Karplus-Strong* synthesis. More general musical oscillators will be discussed in sect. 2.5.

In speech synthesis a long term predictor allows to capture the long-term correlation in the speech signal and to represent the periodicity in speech. It is used in speech coding to synthesize the pitch periodicity. It is given by

$$H(z) = \frac{1}{1 - \sum_{k=1}^{p} b_k z^{-(D+k)}} = \frac{1}{P(z)} \tag{2.42}$$

The parameters of the predictor can be estimated using methods similar to LPC analysis (sect. 2.4.7.2).

2.4.6.1.5 Filter structures Digital signal processing theory explains how to design a linear filter according desired specifications, and many program are available to this purpose. We will not discuss here these methods.

In the first structure, the filters are grouped as a filterbank, where the input signal is fed to many simple band-pass filters. Normally the center frequencies f_{ci} are regularly spaced on the frequency axis. The bandwidth can be equal for every filter as in the channel vocoder or with a constant-Q behavior , i.e. $Q = B_i/f_{ci}$ is constant, as in the third-octave filterbank. For example to simulate struck bars, bells, drums, plucked strings

The parallel structure is quite intuitive to understand. It is similar to filterbank structure, but with center frequncies that depends on the desired signal and which can be time varying. For example to simulate struck bars, bells, drums, plucked strings the structure of fig, 2.22 can be used where a striker filter simulates the interaction of the stick with the resonating structure and teh resonaces are created by a parallel of digital resonaters tuned to the frequencies of the main modes.

The cascade structure use some specialized simpler filter connected din cascade. For example a cascade speech synthesizer can be designed using a cascade of digital resonator, one for every formant.

2.4.7 Speech modeling

This section contains the basics of speech production and speech signal processing.

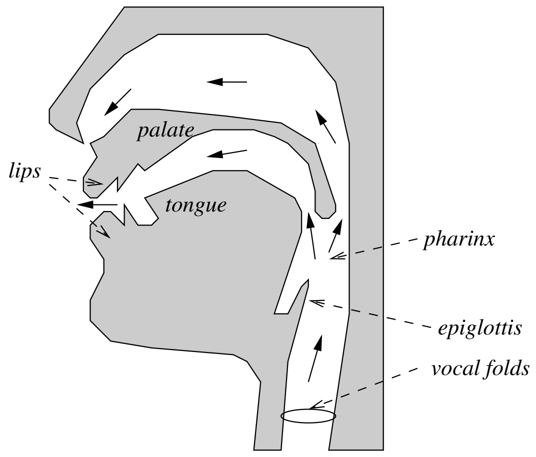


Figure 2.23: A schematic view of the phonatory system. Solid arrows indicates the direction of the airflow generated by lung pressure.

2.4.7.1 Speech production mechanism and models

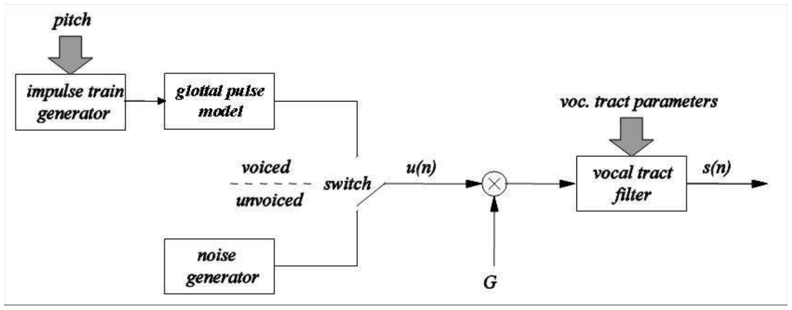
Speech is an acoustic sound pressure wave created when air is expelled from the lungs through the trachea and vocal tract (fig. fig. 2.23). This tract is composed of vocal fold opening (glottis), throat, nose, mouth and lips. As the acoustic wave passes through the vocal tract, its frequency content (spectrum) is altered by the resonances of the vocal tract; vocal tract resonances are called formants. Two types of sounds characterize speech, namely voiced and unvoiced). Voiced sounds result from a periodic excitation of the vocal tract causing oscillation of the vocal chords in a quasi-periodic manner. Technically, it is useful to think voiced speech as the response of an acoustic filter, modeling the vocal tract, to a passing pressure wave. The vocal chords oscillate in a very non-sinusoidal manner at a rate that is approximately 100Hz for adult males and 120Hz for adult females (the frequency varies from speaker to speaker as well). The periodic nature of the oscillations gives rise to harmonics and the period associated with the fundamental frequency is commonly termed its pitch1. The range of potential pitch frequencies varies from 50Hz to 250Hz for men, and from 120 to 500Hz for women. Everyone has a ”habitual pitch level”, which is a sort of ”preferred” pitch that will be used naturally on the average. Pitch is shifted up and down in speaking in response to factors relating to stress, intonation, and emotion. Intonation is associated with the pitch contour over time and performs several functions in a language, the most important being to signal grammatical structure.

In a discrete-time model, the z-domain transfer function for voiced speech $s[n)$ may be expressed by the following cascaded spectral factors:

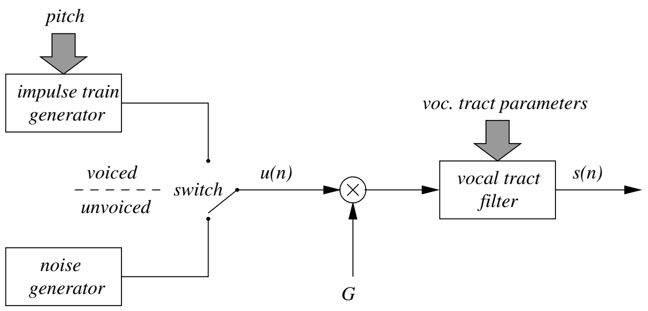
$$S(z) = g_0 X(z) \cdot [F(z) \cdot V(z) \cdot R(z)] \tag{2.43}$$

where g_0 is a constant gain term, $X(z)$ is, in this case, a periodic pulse train, $F(z)$ is a filter component due to the response of the vocal folds to pitch pulses, $V(z)$ is the vocal tract filter component, $R(z)$ is the output radiation component due to lips. The filter $F(z)$ that shapes the glottal pulses can be modeled as

$$F(z) = \frac{1}{[1 - \exp(-c/F_s)z^{-1}]^2}$$



(a)



(b)

Figure 2.24: Models for speech production: (a) general model and (b) simplified

and the radiation filter can be approximated by a differentiator

$$R(z) = 1 - z^{-1}$$

The filter $V_i(z)$ of i-th formant of center frequency f_i and bandwidth B_i has transfer function of the type of eq. (2.35). The cascade model of the vocal tract is given by

$$V(z) = \prod_{i=1}^{K} V_i(z)$$

Unvoiced sounds are due to the passage of turbulent air through a narrow constriction such as the teeth. The turbulence is traditionally modeled as white noise. Theoretically, air is forced through the constriction without vocal fold vibration. In the unvoiced case, the z-domain transfer function is

$$S(z) = g_0 X(z) \cdot [V(z) \cdot R(z)] \tag{2.44}$$

where the input $X(z)$ can be considered to be a driving noise sequence in this case. Both voiced and unvoiced sounds have the characteristic or response of the vocal tract imposed upon them. All sounds may be characterized by eq. 2.43 or 2.44; the complete transfer function characterizing the speech apparatus is defined as

$$H(z) = \frac{S(z)}{X(z)} \tag{2.45}$$

which may or may not include vocal fold response to pitch depending on whether the sound is voiced or unvoiced. Such general discrete-time model for speech is shown in figure 2.24(a). For later purposes, the above general model of speech production can be slightly simplified, dividing the operation of the speech production system as a whole into two functions, excitation and modulation, as shown in figure 2.2 where the combined spectral contributions of the glottal flow, the vocal tract and the radiation of the lips are represented by a time-varying linear digital filter (represented in figure 2.24(b)) with a system transfer function given by eq. 2.45, which can be written as

$$H(z) = \frac{S(z)}{X(z)} = g_0 \frac{1 - \sum_{m=1}^{M} b_m z^{-m}}{1 - \sum_{i=1}^{N} a_i z^{-i}} \tag{2.46}$$

Both parametric and non-parametric techniques are used to establish a model for speech signal. A parametric technique attempts to model a speech production system by estimating the parameters of the transfer function relating the input (usually unknown, but hypothesized) to the output (the speech signal). Linear parametric techniques assume an ARMA (autoregressive, moving-average) model. Linear prediction is a subtype of these autoregressive models.

2.4.7.2 Linear prediction (LPC) analysis

An ARMA(p, q) model is expressed as a linear sum of p past samples and $q + 1$ input values $x[n]$; however, in speech processing the input values (excitation) $x[n]$ are unknown, therefore much development has focused on AR(p) models. We can note that in eq. 2.46, both pole and zeros exist in the transfer function. An approximation of $H(z)$ is constituted by an all-pole model, as given by

$$H(z) = \frac{g_0}{1 - \sum_{k=1}^{p} a_k z^{-k}} \tag{2.47}$$

where g_0 is a gain scaling factor (magnitude of the glottal pulse or white noise), the coefficients a_k are time-varying (but assumed constant during the speech analysis frame) and determined from windowed sections of speech (later referred to as frames).

Transforming eq. 2.47 into the sampled time domain, we obtain the linear predictor

$$s[n] = +g_0 x[n] + \sum_{k=1}^{p} a_k s(n-k) \tag{2.48}$$

which has order p and model parameters (coefficients) a_k. Equation 2.48is the well known linear predictive coding (LPC) difference equation, which states that the value of the present output of the filter, $s[n]$, may be determined by summing the weighted present input, $g_0 x[n]$, and a weighted sum of the past p output samples.

If we assume that the input $u[n]$ is totally unknown, then the signal $s[n]$ can be predicted approximately only from a linear weighted summation of past samples

$$\tilde{s}[n] = \sum_{k=1}^{p} \tilde{a}_k s[n-k] \tag{2.49}$$

The system function of a p-th order linear predictor is the polinomial

$$P(z) = \sum_{k=1}^{p} a_k z^{-k} \tag{2.50}$$

The error between the actual value $s[n]$ and the approximation $\tilde{s}[n]$ is given by

$$e[n] = s[n] - \tilde{s}[n] = s[n] - \sum_{k=1}^{p} \tilde{a}_k s(n-k) \tag{2.51}$$

which is called the prediction error or residual error. From eq. 2.51 it can be seen that the prediction error sequence is the output of a system whose transfer function is

$$A(z) = 1 - \sum_{k=1}^{p} a_k z^{-k} \tag{2.52}$$

It can be seen by comparing equations 2.48 and 2.51 that if teh speech signal obeys the model of eq. 2.48exactly, and if $\alpha_k = a_k$, then $e[n] = g_0 u[n]$. Thus, the prediction filter $a(z)$ will be an inverse filter for the system $H(z)$ of eq. 2.47, i.e.

$$H(z) = \frac{g_0}{A(z)} \tag{2.53}$$

In LPC analysis the problem can then be stated as follows: given measurements of the signal, $s[n]$, determine the parameters $\tilde{a}_k$, $k = 1, \ldots, p$ such that the next sample can be predicted by a linear combination of the past p samples (this principle makes the corresponding predictor to be a short-term predictor, as it user the neighboring samples). The resulting parameters $\tilde{a}_k$ are then assumed to be the parameters a_k of the model system transfer function $H(z)$. The least-square approach is commonly used to determine the parameters $\tilde{a}_k$, by minimizing the total squared error with respect to each of the parameters, i.e., by minimizing the residual error energy, $E = \sum_m e^2[m]$, where the sum is extended to all the frame samples. There are efficient methods to compute the filter coefficients, namely the autocorrelation method, the covariance method, and the Burg algorithm. The gain g_0 is computed assuming that the energy in the error signal is equal to the energy in the excitation signal, i.e.

$$g^2 = \frac{\sum_m e^2(m)}{\sum_m u^2(m)} \tag{2.54}$$

The order p of the linear predictive analysis can effectively control the degree of smoothness of the resulting spectral envelope (see fig 2.25).

A significant feature (spectrum valley) of the LPC spectral modeling is the fact that the LPC spectrum matches the signal spectrum much more closely in the region of large signal energy (i.e. near the spectrum peaks) than near the regions of low energy An example is shown in fig. 2.25.

M-2.16

Write an example of an lpc analyis/synthesis procedure that analyzes a portion of speech and resynthesizes it using the model described above.

M-2.16 Solution

```
%%% headers %%%
svoce=wavread('voce.wav');     %read audio sample
% [...]                        %define Fs accordingly

%%% analysis %%%
s=svoce(8000:10000);           %select signal portion
```

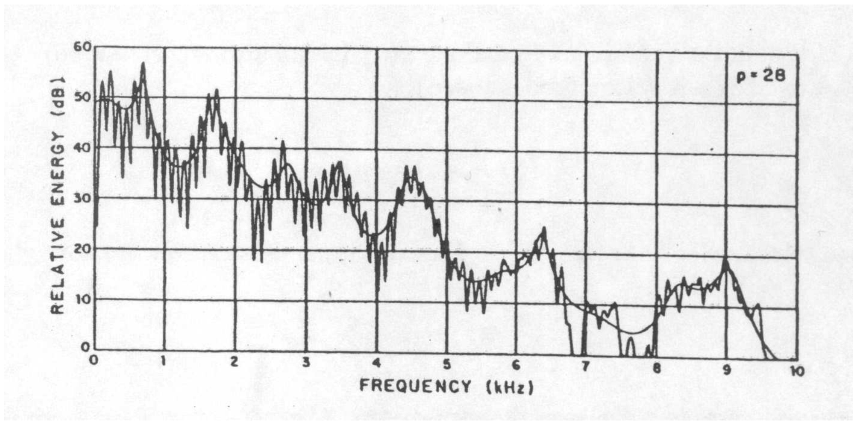


Figure 2.25: LPC spectrum with order $p = 28$ compared with the spectrum obtained by FFT analysis.

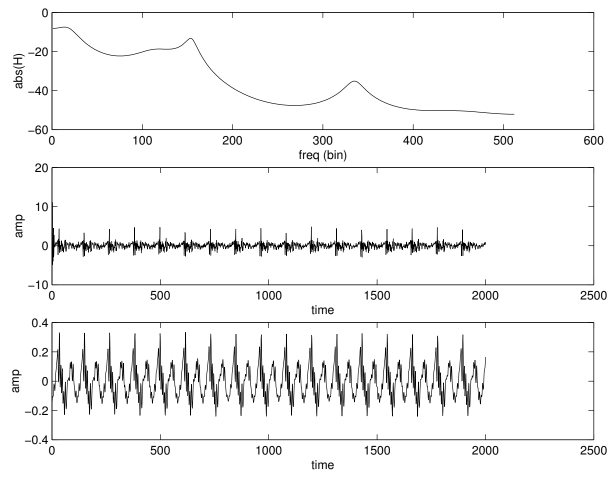


Figure 2.26: LPC analysis and synthesis.

```
Nc=10;                        %no. of lpc coeff. to be computed
[a,g]=lpc(s,Nc);              %compute lpc coeff.
freqz([g 0 0],[a]);           %plot lpc filter response

%%% synthesis %%%
u=filter([a],[g 0 0],s);      %generate glottal excitation
                              %through inverse filtering
snew=filter([g,0,0],[a],u);   %resynthesize signal portion
```

Note that we have used the native function `lpc(s,N)` , where `s` is the input signal and N is the order of the prediction filter. Figure 2.26 shows the frequency response of the formant filter, the glottal excitation, and the resynthesized waveform.

2.4.7.3 Formant tracking

The roots of $A(z)$ (i.e., the poles of the vocal tract filter) are representative of the formant peak frequencies. In other words, the angles of the roots, expressed in terms of analog frequencies can be used as an estimate of the formant frequencies. The frequencies f_i and bandwidths B_i of the poles are extracted from complex conjugate roots r_i of the corresponding polynomial as:

$$f_i = F_s \frac{\arg(r_i)}{2\pi} \tag{2.55}$$

$$B_i = F_s \frac{\log(|r_i|)}{\pi} \tag{2.56}$$

When we extract the format parameters for each frame, we find a lot of discontinuities and local

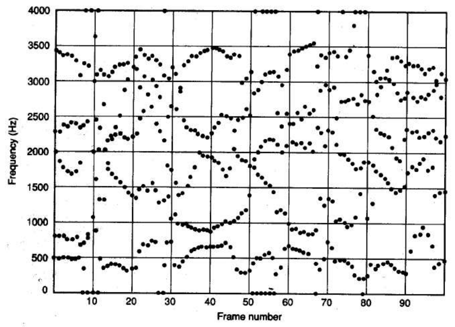


Figure 2.27: LPC analysis and synthesis.

estimation observation errors (Fig. 2.27). In order to find smooth formant trajectories, All poles are then searched through with the Viterbi algorithm (see Chapter on algorithms) in order to find the path (i.e. the formant trajectory) with the lowest cost. The cost is defined as the weighted sum of a number of partial costs: the bandwidth cost, the frequency deviation cost, and the frequency change cost. The bandwidth cost is equal to the bandwidth in Hertz. The frequency deviation cost is defined as the square of the distance to a given norm frequency, which is formant, speaker, and phoneme dependent. This requires labeling of the input before the formant tracking is carried out. Finally, the frequency change cost penalizes rapid changes in formant frequencies, and makes sure that the extracted trajectories are smooth.

M-2.17

Plot the formant frequencies as a function of the frame number, i.e., of time, in order to observe the time-evolution of the vocal tract filter. To this purpose: a) Compute the LPC coefficients for a Hamming windowed speech segment. b) Find the frequencies corresponding to the angles of the zeros of the predictor filter. c)Discard roots whose magnitude is less than 0.8, because it is quite likely that roots whose magnitude is less than 0.8 are not formants. d) Repeat the operation 100 times at lags of half the window length. e) Plot the matrix of the results.

2.4.7.4 Speech synthesis

Synthesized speech can be produced by several different methods. All of these have some benefits and deficiencies. The methods are usually classified into three groups:

- Articulatory synthesis, which attempts to model the human speech production system directly. Articulatory synthesis typically involves models of the human articulators and vocal cords. The articulators are usually modeled with a set of area functions between glottis and mouth. The articulatory control parameters may be for example lip aperture, lip protrusion, tongue tip height, tongue tip position, tongue height, tongue position and velic aperture. Phonatory or excitation parameters may be glottal aperture, cord tension, and lung pressure. The models are developed with the methods which will be seen in the Source modeling Chapter 3. Advantages of articulatory synthesis are that the vocal tract models allow accurate modeling of transients due to abrupt area changes, whereas formant synthesis models only spectral behavior.

- Formant synthesis, which models the pole frequencies of speech signal or transfer function of vocal tract based on source-filter-model.

- Concatenative synthesis, which uses different length prerecorded samples derived from natural speech. This techique is based on the methods seen in sect. 2.3. Connecting prerecorded natural utterances is probably the easiest way to produce intelligible and natural sounding synthetic speech. However, concatenative synthesizers are usually limited to one speaker and one voice and usually require more memory capacity than other methods.

The formant and concatenative methods are the most commonly used in present synthesis systems. The formant synthesis was dominant for long time, but today the concatenative method is becoming more and more popular. The articulatory method is still too complicated for high quality implementations, but may arise as a potential method in the future.

2.4.7.4.1 Formant synthesis Probably the most widely used synthesis method during last decades has been formant synthesis which is based on the source-filter-model of speech described in sect. 2.4.6. There are two basic structures in general, parallel and cascade, but for better performance some kind of combination of these is usually used. Formant synthesis also provides infinite number of sounds which makes it more flexible than for example concatenation methods.

At least three formants are generally required to produce intelligible speech and up to five formants to produce high quality speech. Each formant is usually modeled with a two-pole resonator (see sect. 2.4.6.1.2) which enables both the formant frequency (pole-pair frequency) and its bandwidth to be specified. Rule-based formant synthesis is based on a set of rules used to determine the parameters necessary to synthesize a desired utterance using a formant synthesizer. The filter $V_i(z)$ for the i-th formant of frequency f_i and bandwidth B_i is given by eq. (2.35).

A cascade formant synthesizer consists of band-pass resonators connected in series and the output of each formant resonator is applied to the input of the following one. The cascade structure needs only formant frequencies as control information. The main advantage of the cascade structure is that the relative formant amplitudes for vowels do not need individual controls. The cascade model is given by

$$V(z) = g \prod_{i=1}^{K} V_i(z)$$

A parallel formant synthesizer consists of resonators connected in parallel. The resulting filter is given by

$$V(z) = \sum_{i=1}^{K} a_i \cdot V_i(z)$$

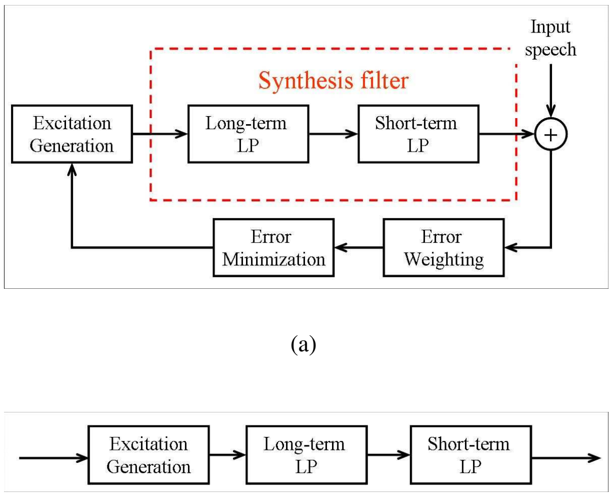
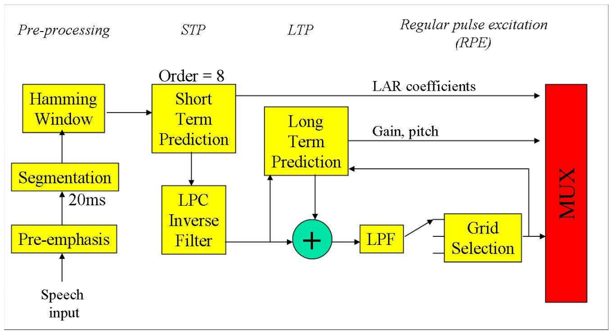


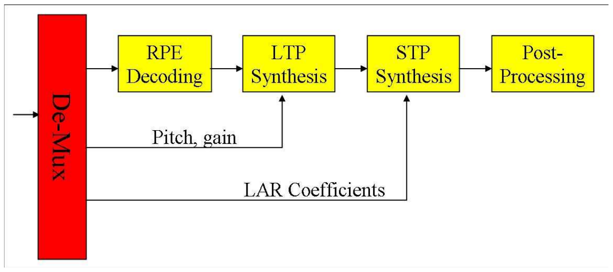
Figure 2.28: Analysis by synthesis LP coding: (a) encoder (b) decoder

Sometimes extra resonators for nasals are used. The excitation signal is applied to all formants simultaneously and their outputs are summed. The parallel structure enables controlling of bandwidth and gain for each formant individually and thus needs also more control information. The parallel structure has been found to be better for nasals, fricatives, and stop-consonants, but some vowels can not be modeled with parallel formant synthesizer as well as with the cascade one.

2.4.7.4.2 LPC synthesis and coding The LPC model is very used for speech coding and synthesis. In fact it allows to extract effective parameters and it is robust to modification. For example it is used in mobile phone communication. In fig. 2.28 it is reported the block diagram of the analysis by synthesis coding system. The encoder estimate the short term Lp coefficients (describing the spectral envelope, the pitch , gain and long term parameters, for generation of the quasi periodic excitation, plus some useful information for the residual (e.g. a code into an appropriate codebook). The GSM mobile phone encoder is shown in fig. 2.29.



(a)



(b)

Figure 2.29: Block diagram of GSM mobile phone: (a) encoder (b) decoder

2.5 Non linear processing models

The transformations seen in sect. 2.4.6, since they are linear, cannot change the frequency of the components that are present. Instead, when non linear transformations are used, frequencies can be even drastically changed and new components are created. Thus, it is possible to vary substantially the nature of the input sounds.

There are two main effects related to nonlinear transformations: spectrum enrichment and spectrum shift. The first effect is due to non linear distortion of the signal and allows for controlling the brightness of a sound, while the second is due to its multiplication by a sinusoid and moves the spectrum to the vicinity of the carrier signal, altering the harmonic relationship between the modulating signal lines. The possibility of shifting the spectrum is very intriguing in when applied to music. From simple components, harmonic and inharmonic sounds can be created, and various harmonic relations among the partials can be established. The first effect try to reproduce the nonlinearities and saturations found on real systems e.g. analog amplifiers, electronic valves. The second one instead derives from abstract mathematical properties of trigonometric functions as used in modulation theory applied to music signal. Therefore, it inherits, in part, the analogic interpretation as used in electronic music and is a new metaphor for computer musicians.

2.5.1 Memoryless non linear modelling

When a sinusoidal input sound $x[n] = \cos(2\pi f_1 Tn)$ passes through a linear system (filter) with transfer function $H(f)$, the output signal $y[n]$ is still a sinusoid with the same frequency f_1 and amplitude and phase depending on the transfer function, i.e. $y[n] = |H(f_1)| \cos(2\pi f_1 Tn + \angle H(f_1))$. Instead if it passes though a non linear amplifier, the waveform is modified and various harmonics are created. Normally we want to avoid distortions in amplifiers, but sometimes, as in amplifiers for electric guitars, we may be interested in emulating the warm sound of valves. In general the output value if a non linear system, depends on present and past values of the input and can be described by the Volterra series expansion. This kind of system are used to compensate the non linear behaviour found in real system, e.g. to linearize the loudspeakers. But is quite to complicate to use and to control. Thus is not suitable for musicians. For this reason in music signal processing often non linearities without memory, i.e. that depends only on the present input value and not on the past values, are used. In this case the system is described by a non linear curve $F(x)$, called *distortion function*, and the output is given by

$$y[n] = F(\ x[n]\ ) \tag{2.57}$$

In analog domain it is difficult to have an amplifier with a precise and variable distortion characteristic. In digital domain the distortion function can be previously computed and stored in a table. During the processing, all that is necessary is to look up the desired value in the table, with an eventual interpolation between adjacent points, as for the table lok up oscillator. In general the distortion function produces infinite partials giving rise to annoying foldover. If $F(x)$ is a polinomial or its Taylor series expansion can be truncated, the bandwidth remains limitated. However, non linear distortions in digital signals can easily surpass $F_s/2$. In this case or $x[n]$ is suitably low pass filtered before the non linear processing, or the non linear computation is done on a oversampled version of the signal.

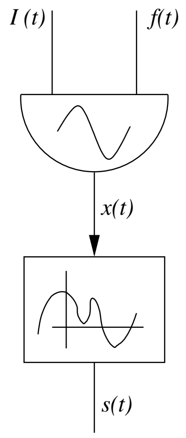


Figure 2.30: Sintesi per distorsione non lineare

For example in order to simulate the overdrive effect of guitar amplifiers, we can use the function

$$F(x) = \begin{cases} 2x & \text{for } 0 \leq x \leq 1/3 \\ \frac{3-(2-3x)^2}{3} & \text{for } 1/3 \leq x \leq 2/3 \\ 1 & \text{for } 2/3 \leq x \leq 1 \end{cases}$$

that produces a symmetrical soft clipping of the input and realizes a smooth transition of the linear behaviour for low level signal to a high saturation for high level sounds. Overdrive has a warm and smooth sound. Asymmetric functions are used for tube simulations. More nonlinear functions are employed for distortion simulation producing tones starting beyond tube warmth to buzz saw effects.

The same technique was used for sound synthesis. In this case we have a sinusoidal input and rich harmonic sound is produced using a distortion function

$$F(x) = p(a \cdot x + b)$$

where $p(x)$ is a polinomial (or a rational) function, a and b are parameters that are used to scale and shift the distortion function.

2.5.2 Synthesis by frequency modulation

This technique does not derive from models of sound signals or sound production; instead it is based on an abstract mathematical description. The definition of "FM synthesis" denotes an entire family of techniques in which the instantaneous frequency of a periodic signal (*carrier*) is itself a signal that varies at sample rate (*modulating*).

M-2.18

We have already seen in section 2.2.1.4 how to compute the signal phase when the frequency is varying at frame rate. We now face the problem of computing $\phi[n]$ when the frequency varies at audio rate. A way of approximating $\phi[n]$ is through a first-order expansion:

$$\phi[n] = \phi[n-1] + \frac{d\phi}{dt}(n-1) \cdot \frac{1}{F_s}. \tag{2.58}$$

Recalling equation (2.8), that relates phase and instantaneous frequency, we can approximate it as

$$\frac{d\phi}{dt}(n-1) = 2\pi \left[\frac{f[n] + f[n-1]}{2}\right], \tag{2.59}$$

where the frequency $f(t)$ has been approximated as the average of $f[n]$ at two consecutive instants. Using the two equations above, $\phi[n]$ is finally written as

$$\phi[n] = \phi[n-1] + \frac{\pi}{F_s}(f[n] + f[n-1]). \tag{2.60}$$

Write a function `FMosc(t0,a,f,ph0)` that realizes a FM sinusoidal oscillator (the parameters `(t0,a,ph0)` are defined as in M-2.3, while `f` is now the sample-rate frequency vector).

M-2.18 Solution

```
function s=FMosc(t0,a,f,ph0)

global SpF;             %samples per frame
global Fs;              %sampling rate

nframes=length(a);      %total number of frames

s=zeros(1,nframes*SpF); %signal vector (initialized to 0)

lastfreq=f(1);
lastphase=ph0;
for (i=1:nframes)            %cycle on the frames
   phase=zeros(1,SpF);       %phase vector in a frame
   for(k=1:SpF)              %cycle through samples in a frame
      phase(k)=lastphase+... %compute phase at sample rate
               pi/Fs*(f((i-1)*SpF+k)+lastfreq);
      lastphase=phase(k);
      lastfreq=f((i-1)*SpF+k);
   end
   s(((i-1)*SpF+1):i*SpF)=a(i).*cos(phase);
end

s=[zeros(1,round(t0*Fs+1)) s]; %add initial silence of t0 sec.
```

Compare this function with the `sinosc` function in M-2.3. The only difference is that in this case the frequency is given at audio rate. Consequently the phase computation differs.

Although early realizations of FM synthesis were implemented in this fashion, in the next sections we will follow an equivalent "phase-modulation" formulation. According to such formulation, the FM

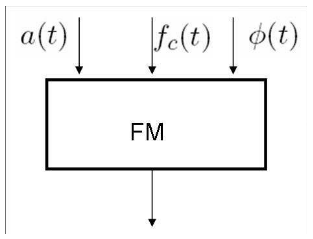


Figure 2.31: FM basic computing module

oscillator is written as:

$$s(t) = \sin[2\pi f_c t + \phi(t)], \tag{2.61}$$

where $\phi(t)$ is the input modulating signal and f_c is the carrier frequency.

For sound synthesis, an amplitude envelope $a(t)$ should be applied. Thus we define a basic FM module that computes

$$y(t) = \textbf{FMmodule}\ [a(t), f_c(t), \phi(t)]) = a(t) \cdot \sin[2\pi f_c(t)t + \phi(t)] \tag{2.62}$$

and is often represented as in fig. 2.31. The algorithm is given by

$$\varphi[n] \;=\; \varphi[n-1] + \frac{2\pi}{F_s} f_c[n] + \phi[n] \tag{2.63}$$

$$y[n] \;=\; a[n] \cdot \sin(\varphi[n]) \tag{2.64}$$

where $\varphi[n]$ is a state variable representing the instantaneous phase of the oscillator. Notice that when the oscillator is implemented by a wavetable, the phases φ and ϕ vary in the interval $0 \dots (L-1)$ and the algorithm becomes

$$\varphi_L[n] \;=\; \varphi_L[n-1] + \frac{L}{F_s} f_c[n] + \phi_L[n] \tag{2.65}$$

$$y[n] \;=\; a[n] \cdot \text{tabsin}(\varphi_L[n] \bmod L) \tag{2.66}$$

2.5.2.1 Simple modulation

When the modulating signal $\phi(t)$ is a sinusoid with amplitude I (modulation index) and frequency f_m, i.e.

$$\phi(t) = I \sin(2\pi f_m t)$$

the simple modulation gives

$$s(t) \;=\; \sin\left[2\pi f_c t + I \sin(2\pi f_m t)\right] \tag{2.67}$$

$$\;=\; \sum_{k=-\infty}^{\infty} J_k(I) \sin\left[2\pi(f_c + k f_m)t\right] \tag{2.68}$$

where $J_k(I)$ is the Bessel function of first kind and k-th order computed in I. From equation 2.68 we can see that the resulting spectrum is composed of partials at frequencies $|f_c \pm k f_m|$ with amplitude given by $J_k(I)$. Notice that negative frequencies, being sine waves, are folded changing the sign. Apparently there are infinite partials, so theorically the signal bandwidth is not limited. However it is practically limited. In the Bessel function behaviour, only few low-order functions are significant for small index values. When the index increases, the number and the order of significant function increase too. Usually in the bandwidth definition of the FM signal, The number M of lateral spectral lines (sidebands) greater than $1/100$ of the nonmodulated signal is given by $M(I) = I + 2.4 \cdot I^{0.27}$, that can be approximates as $M(I) = 1.5 * I$. In this way, varying I it is possible to directly control the bandwidth around f_c. The resulting effect is similar to a low pass filter with varying cut-off frequency. Moreover the amplitude of the partials varies in a smooth way, maintaining constant the overall signal energy.

For the synthesis we can use two basic FM modules in cascade

$$y(t) = \textbf{FMmodule}\ [a(t), f_c(t), \textbf{FMmodule}\,[I(t), f_m(t), 0]\,] \tag{2.69}$$

2.5.2.2 Spectra $|f_c \pm kf_m|$

Equation 2.68 shows a spectrum with lines at frequencies $|f_c \pm kf_m|$, with $k = 0, 1, \ldots$. This kind of spectra are characterized by the ratio f_c/f_m, sometimes also called c/m ratio. When this ratio is rational, it can be expressed as an irreducible fraction $f_c/f_m = N_1/N_2$ with N_1 and N_2 as integers that are prime between themselves. In this case the resulting sound is periodic, since all the partials are a multiple of a fundamental frequency f_0 according to integer factors

$$f_0 = \frac{f_c}{N_1} = \frac{f_m}{N_2}, \tag{2.70}$$

and $f_c,\ f_m$ coincides with the N_1-th and N_2-th harmonic:

$$f_c = N_1 f_0, \qquad f_m = N_2 f_0. \tag{2.71}$$

If $N_2 = 1$, all the harmonics are present and the sideband components with $k < -N_1$, i.e. with negative frequency, overlap some components with positive k. If $N_2 = 2$, only odd harmonics are present, and sidebands superimpose, after foldunder. coincide. If $N_2 = 3$, the harmonics that are multiple of 3 are missing. In general the N_1/N_2 ratio can be considered as an index of the harmonicity of the spectrum. The sound is more harmonious intuitively, when the N_1/N_2 ratio is simple anf formally when the $N_1 \cdot N_2$ is smaller.

The ratios can be grouped in families. All ratios of the type $|f_c \pm kf_m|/f_m$ can produce the same components that f_c/f_m produces. Only the partial coinciding with the carrier f_c changes. Remember that $f_c = N_1 f_0$. For example the ratios 2/3, 5/3, 1/3, 4/3, 7/3 and so on belong to the same family. Only the harmonics that are multiple of 3 are missing (see $N_2 = 3$) and the carrier is respectively the second, fifth, first, fourth, seventh harmonic. The ratio that distinguish a family is defined in normal form when it is $\leq 1/2$. In the previous example, it is $1/3$. Each family is characterized by a ratio in normal form. Similar spectra can be produced using ratios from the same family. We can notice that the denominator N_2 characterizes the spectrum.

When the f_c/f_m quotient is irrational, the resulting sound is aperiodic and hence inharmonic. This possibility is used to easily create inharmonic sounds as bells. For example if $f_c/f_m = 1/\sqrt{2}$, the sound contains partials with frequency $f_c \pm k\sqrt{2}$. No implied fundamental pitch is audible. A similar behaviour can be obtained with complex ratios as $f_c/f_m = 5/7$.

Of particular interest is the case of an f_c/f_m ratio approximating a simple rational value, that is,

$$\frac{f_c}{f_m} = \frac{N_1}{N_2} + \epsilon. \tag{2.72}$$

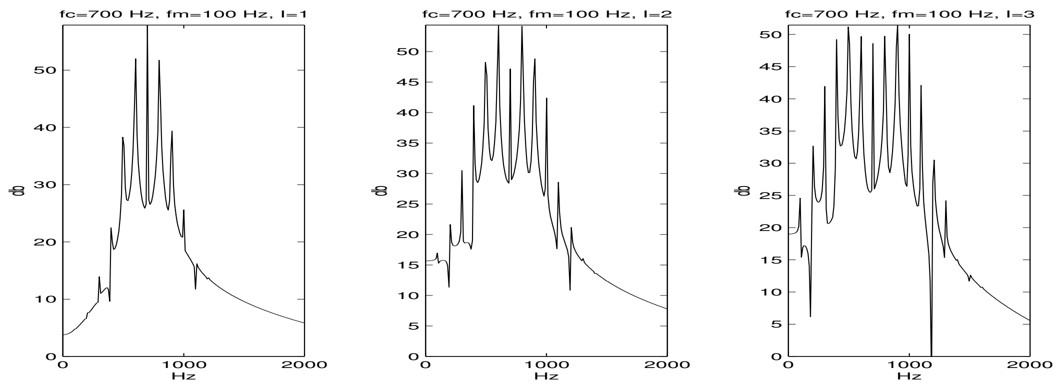


Figure 2.32: Spectrum of a simple modulation with $f_c = 700$ Hz, $f_m = 100$ Hz modulation index I varying from 1 to 3

Here the sound is no longer rigorously periodic. The fundamental frequency is still $f_0 = f_m/N_2$ and the harmonics are shifted from their exact value by $\pm\epsilon f_m$. Thus a small shift of the carrier frequency, does not change the pitch, even if it slightly spread the partials and makes the sound more lively. Notice that the same shift of f_m changes the pitch.

M-2.19

Synthesize a frequency modulated sinusoid, in the case of sinusoidal modulation. Plot the signal spectrum for increasing values of the modulation index.

M-2.19 Solution

```
%%% headers %%%
Fs=22050;          % sampling frequency

%%% define controls %%%
fc=700;            %carrier freq.
fm=100;            %modulating freq.
I=2;               %modulation index
t=0:(1/Fs):3;      %time vector (in s)

%%% compute sound %%%
s=sin(2*pi*fc*t+I*sin(2*pi*fm*t));
```

Figure 2.5.2.2 shows the signal spectrum for 3 values of the modulation index. Note that as the index increases the energy of the carrier frequency is progressively transferred to the lateral bands, according to the predicted behaviour.

2.5.2.3 Compound modulation

There are many variation of the basic scheme. If the modulating signal is composed of N sinusoids, the following relation hold:

$$s(t) \quad = \quad \sin\left[2\pi f_c t + \sum_{i=1}^{N} I_i \sin(2\pi f_i t)\right] \tag{2.73}$$

$$= \sum_{k=-\infty}^{\infty} \prod_{i=1}^{N} J_k(I_i) \sin\left[2\pi(f_c + \sum_{k_i} k_i f_i)t\right] \qquad (2.74)$$

The generated sound will have componets of frequency $|f_c \pm k_1 f_1 \pm \ldots \pm k_N f_N|$ with amplitudes given by the product of N Bessel functions. A very complex spectrum results. If the relations among the frequencies f_i are simple, that is, if the sum of the modulating waves is periodic with frequency f_m, then the spectrum is of the type $|f_c \pm k f_m|$. The frequency f_m can be computed as the greatest common divisor among the modulating frequencies f_i $(i = 1, \ldots, N)$. Otherwise the sonorities are definitely inharmonic and particularly noisy for high indexes.

For example Schottstaedt uses two modulators to simulate the piano tones, setting $f_1 \simeq f_c$ and $f_2 \simeq 4f_c$. It results that we can compute $f_m \simeq f_c$ and it results a picht $f_0 = f_m \simeq f_c$. In this way the small inharmonicity of the piano strings is simulated. Moreover modulation indexes decrease for higher f_0 values In this way the lower tones are richer in harmonics that higher ones.

M-2.20

Synthesize a frequency modulated sinusoid in the case of composite modulation. Plot the signal spectrum.

M-2.20 Solution

```
%%% headers %%%
%[...]

%%% define controls %%%
fc1=700;      %carrier freq.
fm=700;       %modulating freq. 1
fm=2800;      %modulating freq. 2
I1=1;         %modulation index 1
I2=1;         %modulation index 2
t=0:(1/Fs):3;%time vector (in s)

%%% compute sound %%%
s=sin(2*pi*fc*t+...        %sound signal
      I1*sin(2*pi*fm1*t)+I2*sin(2*pi*fm2*t));
```

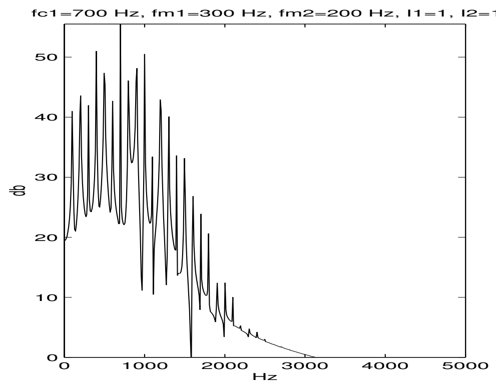
Figure 2.33 shows the spectrum of an FM oscillator with sinusoidal carrier and a composite modulation made of two sinusoids. Note that in the first case the simple ratio between f_{m1} and f_{m2} produces a spectrum of the form $|f_c \pm k f_m|$ (in which $f_m = 100$ Hz, max. common divisor between f_{m1} and f_{m2}, is the fundamental). In the second case, the values $f_{m1} = f_c$ and $f_{m2} = 4f_c$ are chosen in such a way that the fundamental coincides with f_c and that upper partials are harmonic (since $f_{m1} = f_c$ coincides with the max. common divisor between f_{m1} and f_{m2}).

Figure 2.34 shows a double modulator with $f_1 \simeq f_c$ and $f_2 \simeq 4f_c$, with increasing deviations from the exact values multiple of f_c.

2.5.2.4 Nested and feedback modulation

When a sinusoidal modulator is phase modulated by another sinusoids we have

$$\phi(t) = I_1 sin(2\pi f_1 t + I_2 \sin(2\pi f_2 t))$$



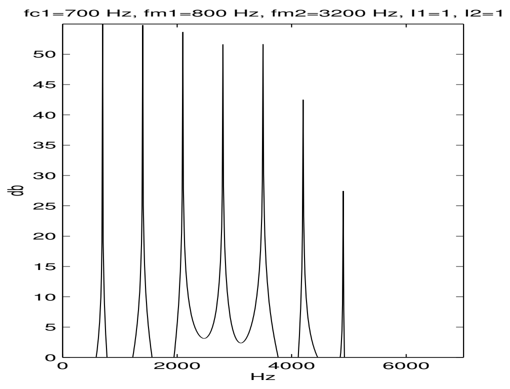
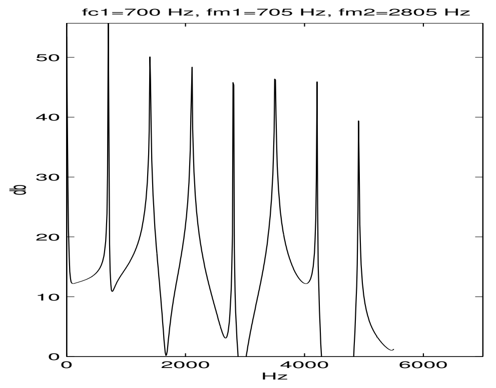


Figure 2.33: Two examples of compound modulation made of two sinusoids.



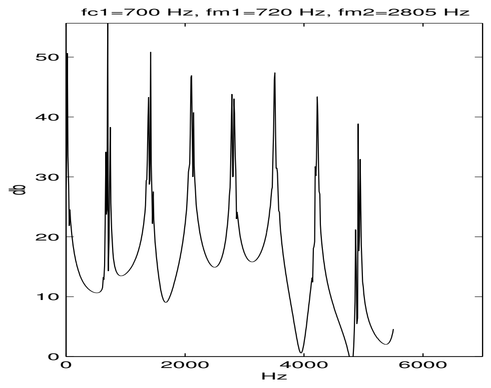


Figure 2.34: Quasi harmonic sounds produced by double modulators with $f_1 \simeq f_c$ and $f_2 \simeq 4f_c$.

and the resulating signal is thus defined by:

$$\begin{aligned} s(t) &= \sin[2\pi f_c t + I_1 \sin(2\pi f_1 t + I_2 \sin(2\pi f_2 t))] \\ &= \sum_k J_k(I_1) \sin[2\pi(f_c + kf_1)t + kI_2 \sin(2\pi f_2 t)] \\ &= \sum_k \sum_n J_k(I_1) \cdot J_n(kI_2) \sin[2\pi(f_c + kf_1 + nf_2)t] \end{aligned}$$

The result can be interpreted as if each partial produced by the modulator f_1 were modulated in his turn by f_2 with modulation index kI_2. The spectral structure is similar to that produced by two sinusoidal modulators, but with larger bandwidth.

As final variation of the basic technique let us consider the case that the past output value is used as modulating signal. This is the so called feedback modulation. This method is described by

$$\begin{aligned} \phi[n] &= \beta s[n-1] \\ s[n] &= \sin\left[2\pi \frac{f_c}{F_s} n + \phi[n]\right] \end{aligned}$$

where β is the feedback factor and acts as scale factor or feedback modulation index. For increasing values of β the resulting signal is periodic of frequency f_c and chages, in a continuous way, from a sinusoid to a saw-tooth waveform. The resulting spectrum has a increasing number of harmonics and it results

$$s(t) = \sum_k \frac{2}{k\beta} J_k(k\beta)\sin(2\pi k f_c t)$$

2.5.2.5 Discussion

Basically FM synthesis is a versatile method for producing many types of sounds. As of yet, however, no algorithm has been found for deriving the parameters of an FM model from the analysis of a given sound, and no intuitive interpretation can be given to the parameter choice as this synthesis technique does not evoke any previous musical experience of the performer. Its main qualities, i.e. great timbral dynamics with just a few parameters to control and to low computational costs, are progressively losing popularity when compared with other synthesis techniques which, though more expensive, can be controlled in a more natural and intuitive fashion. The FM synthesis, however, still preserves the attractiveness of its own peculiar timbral space and, though it is not particularly suitable for the simulation of natural sounds, it offers a wide range of original synthetic sounds that are of considerable interest for computer musicians.

2.5.3 Multiplicative synthesis

The simplest nonlinear transformation consists of the multiplication of two signals. In analog domain it is often called ring modulation (RM) and it is quite difficult to produce in a precise way. Let $x_1(t)$ e $x_2(t)$ be two input signals, the resulting signal is

$$s(t) = x_1(t) \cdot x_2(t) \tag{2.75}$$

and its spectrum $Sf)$ is obtained from the convolution of the two input signal spectra, i.e. $Y(f) = X_1(f) * X_2(f)$.

Usually one of the two input signals is sinusoidal with frequency f_c and is called carrier

$$c(t) = \cos(2\pi f_c t + \phi_c) \tag{2.76}$$

and the other input is the transformed signal and is called modulating signal $m(t)$ (or modulator). Equation 2.75 can be rewritten as

$$s(t) = m(t) \cdot c(t) = m(t) \cdot \cos(2\pi f_c t + \phi_c) \tag{2.77}$$

and the resulting spectrum is

$$S(f) = \frac{1}{2}\left[M(f - f_c)e^{j\phi_c} + M(f + f_c)e^{-j\phi_c}\right] \tag{2.78}$$

The spectrum of $s(t)$ is composed of two copies of the spectrum of $m(t)$: a lower sideband (LSB), reversed in frequency and an upper sideband(USB). The two sidebands are symmetric around f_c. When the bandwidth of $m(t)$ is greater that f_c, part of the LSB extends to the negative region of the frequency axis, and this part is folded around the origin (foldunder). Notice that the phase has to be taken into account while summing components of identical frequencies.

Let consider a sinusoidal carrier (2.76) and a periodic modulating signal of frequency f_m with N harmonics

$$m(t) = \sum_{k=1}^{N} b_k \cos(2\pi k f_m t + \phi)$$

We obtain

$$s(t) = \sum_{k=1}^{N} \frac{b_k}{2} \left[\cos\left[2\pi(f_c + kf_m)t + \phi_k\right] - \cos\left[2\pi(f_c - kf_m)t + \phi_k\right]\right] \tag{2.79}$$

The multiplicative synthesis causes every harmonic spectral line kf_m to be replaced by two spectral lines, one in the LSB and the other in the USB, with frequency $f_c - kf_m$ e $f_c + kf_m$. Notice that the spectral lines of LSB with frequency $kf_m > f_c$, i.e. with $f_c/f_m < k \leq N$, are folded around zero. The resulting spectrum has components of frequency $|f_c \pm kf_m|$ with $k = 1, \ldots, N$, where the absolute value is used to take into account the possible foldunder. The acoustic and perceptual properties of this kind of spectra will be discussed in sect. 2.5.2.2. If the carrier has many sinusoidal components, we will obtain two sidebands around each component and the resulting audio effect is less intuitive.

A variant of this method is amplitude modulation

$$s(t) = [1 + \delta\, m(t)] \cdot c(t) \tag{2.80}$$

where δ is the amplitude modulation index, that control the amplitude of the sidebands. The spectrum is as in eq. 2.78 plus the carrier spectral line

$$S(f) = C(f) + \frac{\delta}{2}\left[M(f - f_c)e^{j\phi_c} + M(f + f_c)e^{-j\phi_c}\right]$$

Contents