POLITECNICO
MILANO 1863

CMLS - Homework 1

Virtual instrument with FM synthesis

Di Palma Riccardo	(Person code: 10602207, ID: 988804)
Gargiulo Antonino Manuele	(Person code: 10829418, ID: 990594)
Morena Edoardo	(Person code: 10865449, ID: 996003)
Orsatti Alessandro	(Person code: 10680665, ID: 994757)
Perego Niccolò	(Person code: 10628782, ID: 992023)

Computer Music - Languages and Systems

Homework Assignment

POLITECNICO
MILANO 1863

Contents

1 FM Synthesis

Frequency modulation synthesis (or FM synthesis) is a form of sound synthesis in which the frequency of a waveform produced by an oscillator, also called *carrier*, is modulated by another oscillator, called *modulator*. When adding more and more carriers and modulators a lot of different signal routings are feasable. Each possible routing is called *algorithm*.
Let's consider an example in which we have only 2 oscillator, a carrier x_c and a modulator x_m, described as follows:

$$x_c(t) = A_c cos(\omega_c t) \quad (1)$$
$$x_m(t) = A_m cos(\omega_m t) \quad (2)$$

The frequency modulated signal will be:

$$x_{fm}(t) = A_c cos((\omega_c + (A_m cos(\omega_m t))\, t) \quad (3)$$

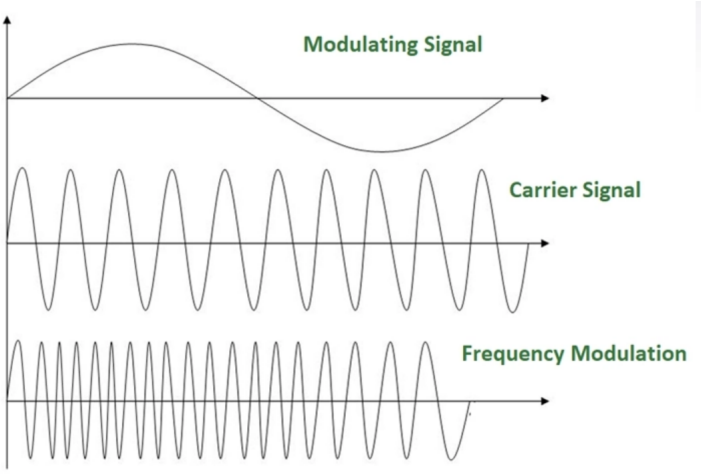

Figure 1: Effects of Frequency Modulation in time domain

1.1 Synthetic Spectrum Components

The spectrum of a frequency modulated signal has peaks in $f_c \pm kf_m$ with $k = (0, 1, 2, ...)$, where f_c is the carrier frequency and f_m is the modulator frequency.

Negative frequencies component are reflected to a positive frequency, i.e. the absolute value is taken, and their phase is reversed. This behaviour is due

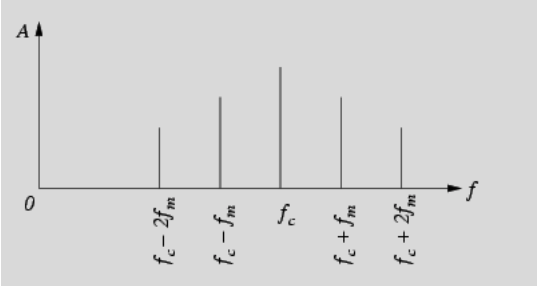

Figure 2: Effects of Frequency Modulation in frequency domain

to the equivalence $sin(-x) = -sin(x)$. In addition, as the modulator frequency increases, the space within the sidebands increases.

1.2 Modulation Ratio

With FM synthesis is possible to create both harmonic and inharmonic sounds. We define the modulation ratio R_m as the ratio between f_m and f_c:

$$R_m = \frac{f_m}{f_c}$$

To create an harmonic signal, R_m should be an integer number. In addition, the Carrier frequency does not have to be equal to the fundamental frequency. The latter is determined by the first peak in the harmonic series.

1.3 Modulation Index

The modulation index is used to control the amplitude of a modulator. This value directly influences the number of frequency components in the spectrum (sidebands), the larger is the value, the richer is the spectrum. There are different ways of defining a Modulation Index. In our implementation it is computed as:

$$I_M = \frac{A_M}{f_M}$$

where A_M is the amplitude of the modulator and f_M is its frequency.

2 SuperCollider Implementation

2.1 Operators

The basic building block of our Synthesizer is the **Operator**, which is an oscillator that produces a waveform, characterized by its type, frequency, and amplitude. There are four different types of waveform availble:

- Sine Wave
- Sawtooth Wave
- Triangular Wave
- Square Wave

The `frequency` parameter of the Operator is multiplied by the sum of two parameters, `ocRatio` and `ofRatio`, where the first stands for *Operator Coarse Ratio* and the second for *Operator Fine Ratio*. As the name suggests, these parameters allow the user to fine tune the frequency of the Operator, in terms of integers numbers (`ocRatio`) or float numbers (`ofRatio`).
The combination of one or more operators results into an **Algorithm**.

2.2 Algorithms

We have selected 4 different algorithms, each one containing 4 operators, implementing them as `SynthDef`:

- **Cascade**: Operator 1, the only carrier, is modulated by Operator 2. Operator 2's frequency is then modulated by Operator 3, which is modulated by Operator 4 itself.
- **Parallel**: 2 carriers, Operator 1 and Operator 3, are modulated by Operator 2 and Operator 4 respectively.
- **Triple Carrier**: 3 carriers, Operator 1, Operator 2 and Operator 3, all modulated by Operator 4.
- **Triple Modulator**: 1 carrier, Operator 1, is modulated by Operator 2, Operator 3 and Operator 4.

In each `SynthDef`, all the operators, depending on their function (carrier or modulator) and on the connections between them, can be characterized by different parameters. If the operator is a modulator, its amplitude is obtained by multiplying the frequency with its modulation index value; if the operator is a carrier, its frequency is modulated with the one(s) of the respective modulator(s). Every operator has a switch in order to turn it on and off, and if a modulator is on, but the respective carrier is off, no sound can be heard at the output.

The resulting signal of each Synth is multiplied by a parametric Envelope, and then is sent to a `bus`. For the algorithms that employ more than one carrier, the SuperCollider `Mix` function is used to send the resulting sum of signals on the same `bus`. An additional Synth called `master` receives the output from the active algorithm and process it with a **Low Pass Filter** (if enabled by the user) before sending it to the actual output `bus`.
The *UGen graphs* of all the `SynthDef`s are available along with the code and this report.

2.3 Normalization

Managing the amplitude normalization with respect to the algorithm topology has proved to be a quite complex procedure. The first thing to take into consideration is the fact that in case of multiple carriers playing at the same time (*parallel* and *triple carrier* algorithms) the amplitude must be normalized in the final routing sum stage. To solve this problem in a dynamic fashion, we implemented a division based on the sum of the `switch` parameters of the operators (1 if on, 0 if off). In this way we take into account the fact that we may want to switch off one or more carriers without compromising the actual volume or normalization process. The division takes place directly on the `amp` argument of the oscillator.

We wanted our synthesizer to be polyphonic. This faced us with the necessity of dealing also with this kind of amplitude normalization problem. Indeed, if the sum of the signals exceed magnitude 1, distortion takes place. Thanks to the global variable `voiceCount`, we can keep track of how many synths are playing simultaneously. Every time a note is played, and therefore a synth is

generated, we set the amplitude of all the carrier operators according to the value contained in the model and displayed in the GUI. This is done by also taking into account how many synths are playing in that very moment. In the case in which the amplitude knob is used while playing, the value of the carrier's amplitude is dynamically updated, while fulfilling the normalization task.

2.4 MIDI

To offer MIDI control for the user, we initialized the `MIDIClient` and requested the connection to all MIDI devices. We defined two `MIDIdef` objects:

- `noteOn`: listener of the "key pressed" event. Everytime a key is pressed, a new synth, playing the corresponding frequency, is generated and stored in the `notes` array, in which all currently playing synths are placed based on their `noteNum` parameter received in the MIDI message.
- `noteOff`: listener of the "key released" event that sets the gate to 0, triggering the release of the envelope, and deletes the synth from the `notes` array.

3 GUI

In our application the user can choose between the 4 different previously described algorithms through a drop down menu. The GUI also presents a *Scope* and a *FreqScope* button, that open a Stethoscope and a Frequency Analyzer respectively, which can be used to visualize what the synths are playing. In addition, the *Mouse LPF* button grants the user the possibility to activate a Low Pass Filter. Both the cutoff frequency (from 20 to 20k Hz) and the resonance (from 0 to 1) values can be operated simply moving the mouse pointer from left to right and from bottom to top of the screen respectively, exploiting the `MouseX` and `MouseY` SuperCollider objects.

In the bottom left section of the GUI the envelope shape designer is found. The user can choose between 4 different starting envelope shapes (ADSR, ASR, Triangle, Perc) and then visually modify levels and times. It is also possible

to set different envelope duration times (1, 2, 3 or 4 seconds).

In the right section of the GUI the parameters regarding all 4 operators can be tweaked. This is possible using 4 different knobs:

- **Coarse Knob**: the frequency of the operator will be multiplied by the number set by this knob. Integer values between 1 and 30 can be picked.
- **Fine Knob**: represents the decimal part of the number selected by the Coarse Knob. Floating point values between 0 and 1 can be picked.
- **Amplitude Knob**: sets the amplitude of a carrier. Floating point between 0 and 1 can be picked.
- **ModIndex Knob**: if the operator is a modulator the Amplitude Knob is substituted by a ModIndex Knob. This knob controls the amplitude of a modulator. Floating point between 1 and 6 can be picked.

The user can also choose, for each operator, between 4 different waveforms (Sinusoidal, Saw, Triangular, Square), and can switch them on and off by clicking on their name.