OPL3 envelope analysis

Jean Pierre Cimalando*

jp-dev@inbox.ru

May 1, 2018

Abstract

This document describes the results obtained in attempts to determine approximations for OPL3 envelope functions and their inverse, using a gray box experimental approach.

I. Introduction

The goal of this envelope analysis effort is to provide good approximations of the key-on and key-off durations, in reasonable computation time.

Existing software realize this measurement operation ahead of time (ADLMIDI). As it is a time consuming operation, it is not applicable to instruments which are modified in real time.

Experimental measurements are realized on Nuked OPL 1.8, a high fidelity emulator of the Yamaha YMF262.

II. Methods

i. Waveforms

For the sake of level computations which will follow, we need a measurement of RMS levels of the internal waves.

The internal waves, numbered 0—7, are computed in source code from *EnvelopeCalcSin* collection of functions. Running these functions in phase range 0—1023, and with envelope 0, produces a period of output normalized in the range ±4096.

The computed RMS levels are presented in 1.

Waveform	RMS level
Sine	0.705614
Half-Sine	0.384713
Abs-Sine	0.307053
Pulse-Sine	0.307053
Sine — even periods only	0.497982
Abs-Sine — even periods only	0.384261
Square	0.997680
Derived-Square	0.214195

Table 1: *Waveform RMS levels*

ii. Envelope generalities

For the envelope generator, we examine in detail the function *EnvelopeCalc*.

A previous study [Steffen Ohrendorf, 2014] has given us a formula to determine an effective envelope rate, based on several instrument parameters, and the frequency of the active note.

- ar, dr, rr: rate of every non-stationary phase of the generator (0—15)
- sl: sustain level (0—15)
- tl: output attenuation (0—63)
- ksr: envelope scaling (0—1)
- nts: impact of note frequency on envelope scaling (0—1)
- eg: whether the generator has a sustain phase (0—1)
- fnum, block: representation of note frequency (0—1023, 0—7)

For the approximations of the envelope I make, for the notion of *rate* I will refer to the effective rate, as defined by the piece of C code 1. The variable *ins_rate* is to substitute with either

*

```
unsigned eff_rate = 4 * ins_rate;
if (!ksr)
  eff_rate += block >> 1;
else
  eff_rate += (block << 1) |
    ((fnum >> (9 - nts)) & 1);
```

Figure 1: *Effective envelope rate*

```
int tlv = tl << 2;
int kslv =
  ((kslrom[fnum >> 6] << 2) -
    ((8 - block) << 5)) >>
  kslshift[ksl];
```

Figure 2: *Envelope level modifiers*

ar, *dr* or *rr*, depending on which phase of the envelope is under study.

Finally the envelope has two level modifiers which subtract from the output level, depending on *tl* and *ksl*. These are determined by the C code 2.

The envelope generator produces its output in 0—511, but reversed. I will consider the envelope output which is reversed back to normality and scaled to 0—1 range.

iii. Attack phase

To examine a particular phase of envelope generation, I ran it under all possible effective rates, resulting from all combinations of parameter values and note frequencies.

I wrote a program which took samples of envelope output, and was able to categorize the samples into their envelope phases, by examining the value of the emulator for *eg_gen*.

I have converted the sample intervals into real time, by scaling down by the native sample rate of 49716 Hz.

This gives a 2-parameter sampled function $(Time, Rate) \rightarrow Envelope$.

For the attack phase in particular, this function is an exponential shape. I can find a decent approximation by fitting this function to:

$$E(t, r) = 1 - \exp(-ab^{r+c}t) \qquad (1)$$

where t, r are time and rate respectively, and a, b, c are curve fitting constants shown in 2.

a	1.149780
b	1.189578
c	-1.771204
R-square	0.9956
RMSE	0.01699

Table 2: *Attack phase approximation*

a	0.010613
b	1.185552
c	1.013799
R-square	0.9895
RMSE	0.02902

Table 3: *Release phase approximation*

iv. Release phase

By following exactly the same method as described in iii, I obtain a sampled function for the release phase.

This one has linear shape and is approximated by the following function:

$$E(t, r) = 1 - ab^{r+c}t \qquad (2)$$

where t, r are time and rate respectively, and a, b, c are curve fitting constants shown in 3.

v. Estimating instrument durations

Given a FM instrument for OPL3, I am interested in determining two particular values.

t_{on} is defined as the duration for output amplitude to raise above or equal the target amplitude value V_t after key-on.

t_{off} is defined as the duration for output amplitude to raise below or equal the target amplitude value V_t after key-off.

We have $E(t, r)$ as the envelope function defined over time and rate. In addition, I define $E'(t, r)$ as the adjusted envelope function, subtracting level modifiers from envelope $E(t, r)$. Level modifiers have been previously explained in ii, 2.

The total amplitude level at some time point can be explained in words, as the combined amplitude of all carriers under the AM modulation of their respective envelopes. In FM synthesis a modulator has little impact on overall signal level, which is why I consider only carriers to simplify.

If N is the number of carriers, R_i is the effective rate of the i^{th} carrier, and W_i is the RMS amplitude of the i^{th} carrier's waveform, overall RMS level $V(t)$ is defined as:

$$V(t) = \sqrt{\sum_{i=1}^{N} (E'_i(t, R_i)W_i)^2} \tag{3}$$

In order to determine the duration we want, we have to substitute $V(t)$ for our desired target level, and then solve for t.

vi. Key-on duration

In order to solve 3 we can use the analytical approach, but the exponential formula for attack makes this difficult.

For a simple way to do it, we can repeatedly evaluate 3, and narrow down t with the binary search method.

With a range 0—10 for t and 16 iterations, there is approximately a maximum absolute error of 1.5×10^{-4}.

vii. Key-off duration

We can solve this analytically, by plugging $E'(t, r)$ in 3.

To make the writing more concise, let's define R'_i the slope of the i^{th} carrier as $a \times b^{r_i+c}$, and $\hat{V}_i$ the starting point of the i^{th} carrier as $Vs_i - Vm_i$, where Vs_i is the sustain level and Vm_i is the level modifier ii, 2.

This gives:

$$V(t) = \sqrt{\sum_{i=1}^{N} ((\hat{V}_i - R'_i t)W_i)^2}$$

$$V^2(t) = \sum_{i=1}^{N} (R'^2_i W^2_i)t^2 - (2R'_i \hat{V}_i W^2_i)t + \hat{V}^2_i W^2_i$$

To determine t we can have our direct answer by substituting $V(t)$ with the target level, and then solving roots of the quadratic polynomial:

$$(\sum_{i=1}^{N} R'^2_i W^2_i)t^2 + (\sum_{i=1}^{N} -2R'_i \hat{V}_i W^2_i)t + (\sum_{i=1}^{N} \hat{V}^2_i W^2_i) - V^2(t)$$

References

[Steffen Ohrendorf, 2014] Steffen Ohrendorf. (2014). OPL3 analyzed. *OPL3-FPGA*