
CUT dB BOOST
CUT dB BOOST
CUT dB BOOST
PRE-IN
IN
SHELF
IN
IN
INPUT
Hz
Hz
kHz


Acustica
MAGENTA II
INPUT
OUTPUT
POWER
HIGH
LOW


Acustica
CUT/BOOST
INPUT TRIM
INPUT
FREQUENCY
SHELF
OUTPUT
dB/OCTAVE
BYPASS
PRE-IN
IVORY / IAE-2
MASTERING STEREO EQUALIZER


CORE 6
Pure Class A
EQUALIZER
AMBER
Acustica
INPUT
OUTPUT


HP
LOW
MID 1
MID 2
HIGH
INPUT TRIM
VCA MAIN OUT
LP
POWER
MIC PRE
OUTPUT
Acustica
Honey


Core6 Tech Notes

1. Introduction

We have recently introduced a class of plug-ins based on a new Engine, whose performance is outstanding compared to our previous technology, in terms of both CPU and latency. Many users are wondering whether this new technology can run any of the libraries created before, where this new efficiency stems from, and how everything works in technical terms.

Acqua as well as Nebula products are not derived by compiling a different code depending on needs: they rather embody a general-purpose engine that runs “scripts” described by a proprietary meta-language. These scripts describe the whole object in terms of both graphics and sound.

As we will explain later, this structure does not affect the final performance: a single unified engine allows to transparently access a series of enhancements carried out over the years and build on what has been done with other products in terms of code reuse and calculation efficiency.

This document stems from the need to be clear about what we do, describe what distinguishes us from others manufacturers, and explain why we are so proud of our work and see it as a little technological miracle.

2. Under The Skin

It all starts with opening Amber2: its bands begin to run and we start to work.

Easy, isn't it?

Today it is possible to open hundreds of instances, with a slightly higher but perfectly real-time-manageable latency, and slightly higher loading times. We need to grasp what amazing complexity lies behind these seemingly simple operations. Follow me in my journey.

First of all, it is very important to understand the difference between the information that needs to be managed by our engine and by a traditional engine.

3. FIR vs IIR

The amount of information to be stored and used for real-time calculations in a traditional way is quite small:

Code:

```
//--------------------------------
//  [code]
//--------------------------------

//fc -> cutoff frequency
//pi -> 3.14285714285714
//srate -> sample rate

//===============================================
//  shared for both lp, hp; optimizations here
//===============================================
wc=2*pi*fc;
wc2=wc*wc;
wc3=wc2*wc;
wc4=wc2*wc2;
k=wc/tan(pi*fc/srate);
k2=k*k;
k3=k2*k;
k4=k2*k2;
```

```
sqrt2=sqrt (2);
sq_tmp1=sqrt2*wc3*k;
sq_tmp2=sqrt2*wc*k3;
a_tmp=4*wc2*k2+2*sq_tmp1+k4+2*sq_tmp2+wc4;

b1=(4*(wc4+sq_tmp1-k4-sq_tmp2))/a_tmp;
b2=(6*wc4-8*wc2*k2+6*k4)/a_tmp;
b3=(4*(ec4-sq_tmp1+sq_tmp2-k4))/a_tmp;
b4=(k4-2*sq_tmp1+wc4-2*sq_tmp2+4*wc2*k2)/a_tmp;
```

However, the situation can be even more complicated – developers often solve very complex equations to determine the different coefficients.

Several approaches can be adopted: the one described in this book (and used by a number of developers) is very interesting:

http://www.native-instruments.com/fileadmin/ni_media/downloads/pdf/VAFilterDesign_1.0.3.pdf

Most of the traditional methods share a basic common feature: a limited amount of information to be managed, in terms of coefficients/bytes. In a few words, a lot of operations are performed cycling on a small set of state variables.

Acustica has always followed a different path for Nebula and Acqua series:

- Use FIR filters, for both the fundamental and harmonics
- Have a dynamic approach, where possible
- Have an approach based only on sampling
- Improve the original models by also making hybrid and improved implementations, yet still using a brute-force approach based on convolution (never reduce filters to minimum-phase filters, never reduce them to IIR coefficients).

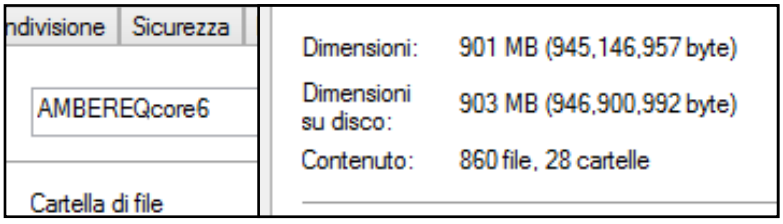
The engine development has taken 10 years of work; it looks more like a real-time graphic rendering engine (like those normally used in 3D videogames) than a traditional plug-in.

Explaining why a FIR approach is more useful than an IIR approach would fall outside the scope of this document: we can say that Acustica is currently a market leader in the FIR approach to analog modeling (paying attention not only to the equalization curves, but also to the harmonics and dynamic behaviour of the model).

4. Huge Data Management

Let's start from the amount of data collected: the sampling session usually needs tens of GB of space.

After the deconvolution process is applied, the data related to a single sampling frequency are obtained. For example, Amber at 96Khz needs about 1 GB on disk:



The huge amount of data is reduced using compression algorithms and various optimization techniques at 500 MB for the main native frequencies (44, 48, 88.2, 96Khz).

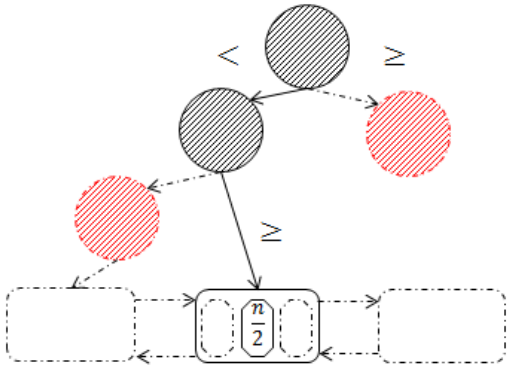
- Compression through Google Snappy was introduced in April 2015. The first product to use it will be Honey (all other products will follow later)
- Different optimization algorithms have been introduced to Nebula over the years
- The rate conversion algorithm was improved in May 2015. We previously used high-quality external algorithms, but we then realized they were inaccurate with impulse responses, due to decimation processes that, when uncontrolled, can damage significant samples and result in interpolation errors.

5. A Database Inside a Plug-in

When a band is enabled, the following operations are carried out in a split second:

- A file related to the enabled band is decrypted. This file actually represents a compressed folder bigger than 100 megabytes and containing in turn hundreds of files.
- The files protected by executables applying copy protection are repaired and restored. The current protection algorithm is called Scorpion, and it was made extremely performing in April 2014 with Trinity2, allowing to process hundreds of thousands of files per second.
- If the rate is not correct, a sample rate conversion must be done for all samples needing it. The algorithm was developed for Nebula in 2006 and has been improved over the years. The aim is to process thousands of WAV files at the same time without wasting time in building and destroying new data structures.
- The compressed folder was upgraded thanks to an algorithm introduced in September 2014. The O2 trees are particularly performing with range queries: the insertion and deletion of a node is performed at the average rate of 1 million operations per second, while range queries are execut-

ed at the rate of 50 million operations per second for each core. This enhancement was developed and improved by Acustica in 2012.



- Identical files are shared between the various instances. This improvement was introduced to Core3 in 2009.
- An XML file representing the preset structure is parsed. This file is often bigger than 100 KB. Processing an XML file at the current speed of the engine requires special precautions: for example, we had to create our own class to manage the strings, in order to avoid the allocation of new memory areas where possible. The strategy is the same as in the Google Performance Tools, yet applied to our specific case. Over the years, the development has achieved the state of the art, doing its job better than any third-party alternatives tested for this task.
- The impulses/Volterra kernels are scanned, interpreted and synchronized. This is a run-time operation, as it can vary based on the sample rate.
- A balanced tree is built to allow the maximum operating speed of the vectorial engine, for the purpose of deriving the final Volterra kernel based on the user parameters and other internal variables such as dynamics, time, or the output value of modular engine components.
- Further improvements have recently been made:
- The previous preset is disabled during the loading process. The bypass state, when on, is managed through a cross-fade that reduces undesired noises. The development was completed for Aquamarine in January 2015.
- The loading is achieved using multithreading techniques (particularly useful in the host start-up phase). In a few words, each band is loaded at the same time as the others. This was developed for Magenta in November 2014, and is based on a complex synchronization mechanism between processes.

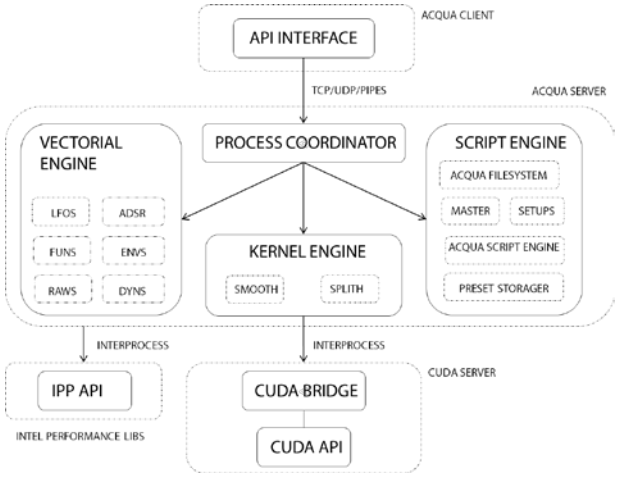
6. The Vectorial and Kernel Engine

Now, let's see how audio processing works:

- Vectorial Engine: at time intervals defined by the PROG RATE, all sample trees are valued (impulse responses/Volterra kernels). This time, corresponding to a sort of Frame Rate, usually varies from 2 to 50 milliseconds (with peaks of a few tens of microseconds for compressors). In Amber2, the PROG RATE was set at 20 milliseconds. Each band needs an independent instance where the following operations are carried out:

    - Control sources are valued
    - The various internal synthesis modules of the engine are valued (LFO, envelope followers, envelopes, functions, and so on). Some of these modules are used to control the mapping of some user variables. For instance, the development of Amber1 in November 2014 has called for new particular functions to accurately map the sample mixing by following the logarithmic trend of the gain control. Special functions map the correct sample with the control position.
    - When they are not used, some of these structures are disabled and not valued. Almost all modules are equipped with an internal cache. This improvement was completed in April 2015 for Aquamarine Murano.
    - Balanced trees are valued, running the sample merge/interpolation. In particular, the result of a sub-tree can be unvalued (put into cache memory) if the control variables are not changed within a margin of tolerance. This optimization was implemented for Nebula in 2009.

- The trees are modulated with the previous result, in order to make the result more fluid and limit undesired zip-noises while changing the parameters/values. A complex technique called SMOOTH2 was introduced to Nebula in 2008.
- Some tree portions are valued at different frame rates. It is possible to value the sample merge which is not currently being executed more slowly than the samples involved in the synthesis process. This enhancement, called SPOTOFF, was introduced in July 2014 while developing Prime products.
- The vectors related to harmonics with dynamic variability are valued at asynchronous and different time scales compared to the fundamental, thus limiting the number of useless operations and saving further clock cycles. This feature was introduced to Acqua products in November 2014 with Magenta.



– Kernel Engine: audio processing is performed by the Kernel Engine of each internal instance:

- A routing engine was first developed for Trinity in February 2014.
- The various instances are connected together by a complex set of connectors that allow to perform basic operations on the audio signal (separate a channel, use the MS decomposition, and so on). CONNs were introduced to Acqua products in December 2014 with Aquamarine and are used, for example, in Amber to keep out particular instances called GHOST, which we will describe later.
- Each Volterra order needs convolution operations. Convolution is sometimes performed directly, sometimes through FFT. Partitioned and optimized convolution for harmonics is at the heart of all our products and was implemented in 2006. Listing the small improvements and little corrections made over the years would lead me to write another document.
- VECTOR SYNCs were introduced with Ivory, in March 2015. They represent a milestone that we called “Core6”. The Kernel Engine of each instance is disabled and the Vectorial Engine alone is valued. The instance is called GHOST, as it does not actively take part in the audio processing. The result of a tree partial section is forwarded to the main instance called MASTER, which brings together data from several sources and creates the final tree to be submitted to the Kernel Engine. This correction allows to reduce latency, as only the master creates audio delay.

- The master processes the various partial trees at a different rate, depending on the data availability in the different instances. The concept of KERNEL RATE is introduced, which allows to reduce CPU consumption up to one tenth of the original value. Therefore Core6, used for equalizers, allows to improve not only latency but also resource consumption, increasing the number of possible instances.
- The KERNEL RATE is improved to become asynchronous towards the Kernel and Vectorial Engine, so as to enhance the general reactivity of the product. This improvement was introduced with Versatile EQ in May 2015.
- The concept of MIX RATE is introduced to make automations and the product's reactivity more fluid to the user control, making the master asynchronous and variable based on the workload. This optimization was introduced with Amber2 in May 2015.

Next time you open Amber... think of all this!

Enjoy your Mix!