Different Ways to Write Digital Audio Effects Programs

Daniel Arfib
CNRS-LMA, 31 Chemin Joseph Aiguier
13402 Marseille Cedex 20
arfib@lma.cnrs-mrs.fr

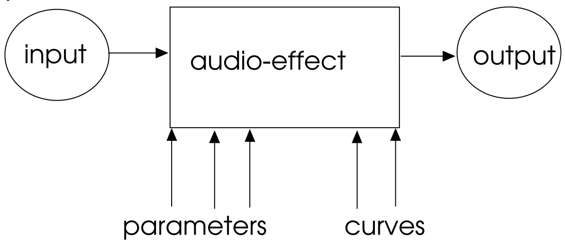
Abstract

This paper is a very basic one, where one tries to explain how one can write a digital audio effect in a non-real time situation with a very general mathematical language such as MATLAB, and how such digital audio effects can be used in the real life.

1 What is a digital audio effect?

The term « digital audio effects » has been used as an acronym for the COST action G6. So what is a digital audio effect?

.



First of all it is not a "sound effect", in the sense of a collection of sounds be them natural or artificial. A digital audio effect is a process which when applied to a sound modifies it in a way to improve it or at least make some aspect of it more evident. Some of these effects come from the analog electronics. Reverberation units for example were the good companion of mixing tables and equalizers. Then computers came and it was possible to control different units, or even do some digital processing in real-time or out of real time.

2 Real time and non real time

The implementation of digital audio effects roughly depends on one choice: real time or non real time. This does not mean slow or fast, but: do we need to hear the sound at the same time that it is processed or not.

2.1 Non real time

The non real time situation has been the only way to use computer for digital audio during years: computers were slow and unadapted in their hardware. But the question nowadays is not so much the question of processing than the one of control. It is a totally different situation for a musician to control one sound in real time or not. The non-real time allows to compose with sounds in a way which can be totally innovative and complex: for example one can draw very precise curves to control complex processes. Also one can analyze the sound characteristics in order to find the good processing. In a non real time situation, sounds are recorded and then processed. The processing time can be small enough that we can hear the result at once after the recording. It can be also very large, passing through different steps before getting an output sound. In some way the constraints are less than in real time situations.

2.2 Real time

The real-time situation is often the one of a performance. Two questions appear: how complex can be the process and how can we control it. This has a great influence on the implementation of the process: the mean processing time for one sample must be less than the running time, the time lag between the input and the must be very small in order not to hear a delay. Though the control rate can be different, the control must be responsive.

3 Three ways

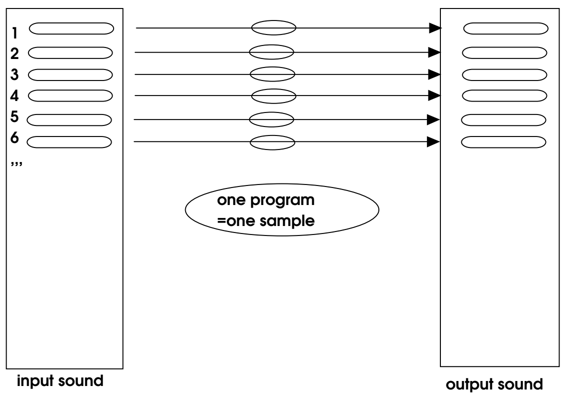
Basically three types of processing can apply:

- the sample by sample technique, where for each input sample one output sample is calculated.

- the block technique, which is very related to MusicX synthesis programs. It has been proved that it is faster to synthesis sound using buffers, especially if the program compiles or interprets instructions and itself uses languages such as C, FORTRAN or Pascal

- the vector technique, where the entire input sound is considered as a vector and the processing is done on this vector. Languages such as Matlab are strongly oriented to such a treatment and are optimised for all the vector or matrices operations.

3.1 Sample by sample



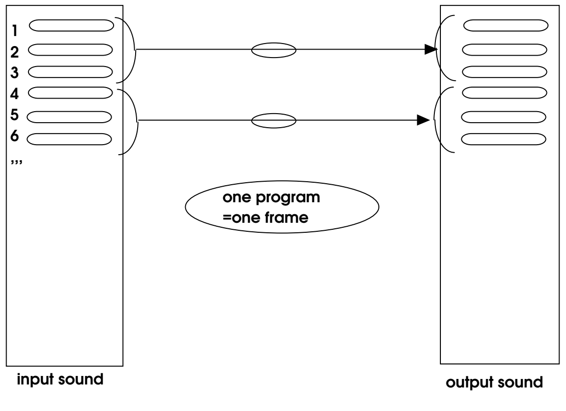
This technique can be called "analog device simulation" because the output is calculated sample by sample, giving a regular output flow. Given that the delay time between the input and the output is reduced to one sample it is very convenient for real-time situations.

The way to write such a program is to have an unique program which inputs one sample and outputs one. As an example a filter can be easily written as a "sample by sample" program, providing that some memory is kept for some provisional variables.

This does not mean that « IF instructions » cannot be taken but that the longest path inside the program must be executed in less than one sampling period. For example rectifiers or threshold gate can be implemented if the machine has a logical if. in its instructions.

This technique is mandatory in some dedicated machines. Fore example some real time machine use a fixed set of pipe line instructions and a program itself must be held under a fixed number of lines.

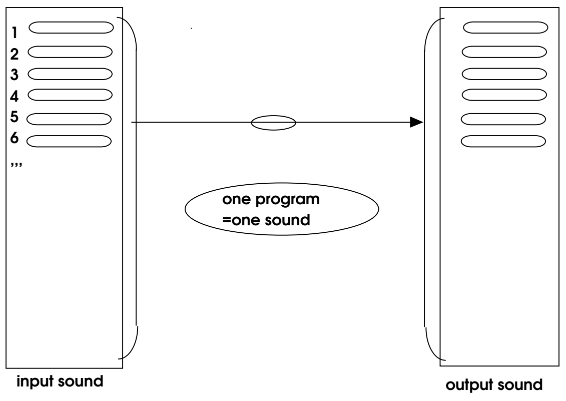
3.2 Frame by frame



this is the situation usually used to process sounds: the output sound is calculated and output frame by frame. This means that the delay between the input sample and the output sample is at least the length of the frame buffer. Compared to the sample by sample approach, this technique allows a better distribution of the computation load: some samples can require more time than other, or there can be an initialization time for some specific process.

This technique is a necessity for some techniques like the use of the sliding Fourier transform. But in this case frames are usually overlapping

3.3 Vectorized



This technique is resolutely linked to non-real time situations or the processing of individual notes. It is used in real time only for specific applications using big delays between a sound and its processed version.

It is the normal way for example to write Matlab programs. With this kind of approach, an input sound is considers as a vector. All the processing units will use a vector as an input and output. So a complete digital effect will be a series of such procedures.

The main advantage in term of processing time are:

- each procedure is independent and can be independently tested

- the processing time can be very fast as far as each procedure is well written

- an effect consists only in the list of chained procedures.

the drawbacks of such a way to write are:

- it is memory consuming: there must be space for as many vectors than what are used even as a transitory variable

- It is absolutely not suited to real-time situations because one supposes that the input sound is already known when the computation is going on.

4 The framework of a DAFX: the "do-nothing effect"

I would like to show three implementations written in the Matlab language using the simplest example: the do-nothing program.

the sound to be processed is supposed written on a hard disk in a raw format, and that the file is mono, 16 bits. The easiest way to simulate a real time process with Matlab is to have the file ready as is it should be in such a condition

4.1 sample by sample

```
%
clear;
clf;

% INPUT FILE
fid=fopen('input.son')
status=fseek(fid,0,'eof')
nbytes=ftell(fid)
% number of bytes in the file
status=fseek(fid,0,'bof')

% OUTPUT FILE
fid2=fopen('son.son','w+')
%
%
nwords= nbytes/2;
% one word is two bytes
%
% DAFX
for n=1:nwords
    bufin=fread(fid,1,'int16');
    bufout=bufin;

% here insert the effect

    fwrite(fid2,1,'int16');
end

    fclose(fid);
    fclose(fid2);
```

This is a very inefficient program in a Matlab implementation because there is one disc access by sample and also Matlab is oriented toward vector processing.

4.2 Frame by frame

This time the input sound is read on the disk frame by frame, processed and written on the disk. The only difference with the preceding technique is there is a loop in order to read the whole file. One must be careful in the fact that the last buffer can be of a different size. One particularity of MATLAB is also that the length of the file is calculated in bytes, but the length of the buffer is calculated with the unit that is used for reading

```
%
clear;
clf;
% INPUT FILE
fid=fopen('input.son')
status=fseek(fid,0,'eof')
nbytes=ftell(fid)
% number of bytes in the file
status=fseek(fid,0,'bof')

% OUTPUT FILE
fid2=fopen('son.son','w+')
%
%
step  = input ('step in the file: ')
nframes= 1+nbytes/(2*step);
% one word is two bytes
%
% DAFX
for n=1:nframes
    bufin=fread(fid,step,'int16');
    bufout=bufin;
% insert the effect here
    plot(bufout);drawnow;
    fwrite(fid2,bufout,'int16');
end

    fclose(fid);
    fclose(fid2);
```

It is also possible to use a step from frame to frame that is different from the size of the frame. This is the case of the use of a sliding FFT algorithm. In this case the fread second argument must be the length of the frame and not the step.

4.3 Vectorised

The structure is straightforward: files have to be initialized, the input file as well as the output file. Then the input file is read in memory, the effect is applied, and the output file is written on the disc. This is the natural way to use Matlab: all the instructions of Matlab have been optimized for vector and matrix operations.

```
% THE DO-NOTHING DAFX
clear;
clf;

% INPUT FILE
fid=fopen('input.son')

% OUTPUT FILE
fid2=fopen('son.son','w+')
%

% DAFX

    bufin=fread(fid,Inf,'int16');
    bufout=bufin;
    nwords=length(bufout)
    fwrite(fid2,nwords,'int16');

%

    fclose(fid);
    fclose(fid2);
```

5 The implementation of DAFX on different machines

So the idea is to start testing an effect in MATLAB with the vectorized structure, which evident limit is the need for short sounds.

Then it is interesting to test the feasibility of a sample-by-sample or a frame by frame structure using a Matlab program.

Then it would be possible to make an implementation in a different language or a dedicated machine such as a DSP board. The matter of plug-ins and their implementation should be of importance in the development of such effects too.

6 Conclusion

One goal of COST G6 action is to collect programs that do digital effects in a Matlab-like language. These effects can touch different domains such as filtering, modulation, delay lines, non linear processing, time-frequency manipulation, spatialisation, spectral processing and so on. Each effect can be written in a simple way providing that the framework for testing is the same: namely the three proposed ways to write programs, where the do-nothing line is replaced by the effect itself.

Each effect can then be tested with different source sounds, and moreover the control parameters can be described in technical or musical terms.

7 References

A previous article of the author refers to Matlab programming oriented towards sound

Arfib, D.. *une boite à outils de traitement sonore en matlab*. rapport Laforia, Mars 94, pp68-72

At this stage of the process, these general references can help:

[1] Kahrs, Mark and Brandenburg, Karlheinz. Applications of Digital Signal Processing to Audio and Acoustics. Kluwer. 1998

[2] Moore, F. Richard. Elements of Computer Music. Prentice-Hall, New Jersey. 1990.

[3] Orfanidis, Sophocles. Introduction to Signal Processing. Prentice-Hall. New Jersey. 1996.

[4] Pohlmann, Ken. C. Advanced Digital Audio. Sams. 1991.

[5] Roads, Curtis. The Computer Music Tutorial. The MIT Press, Cambridge, Massachusetts. 1996.

[6] Roads, Curtis. The Music Machine. The MIT Press, Cambridge, Massachusetts. 1986.

[7] Roads, Curtis and John Strawn. Foundations of Computer Music. The MIT Press, Cambridge, Massachusetts. 1985.

[8] Steiglitz, Ken. A Digital Signal Processing Primer. Prentice-Hall, New Jersey. 1997.

[9] Zölzer, Udo. Digital Audio Signal Processing. J. Wiley & Sons, Chichester. 1997.

[10] Mathews M., Pierce JR Current Directions in Computer Music Research. MIT Press, Cambridge, Mass, 1989.