Impulse-Response and CAD-Model-Based Physical Modeling in Faust

P.-A. Grumiaux, R. Michon, E. Gallego Arias and P. Jouvelot
pierreamaury.grumiaux@gmail.com, rmichon@ccrma.stanford.edu,
{emilio.gallego_arias,pierre.jouvelot}@mines-paristech.fr

Context

The Faust programming language [4] has proven to be well suited to implement physical models of music instruments using waveguides and model synthesis [1][2][3]. We developed two tools allowing to easily generate Faust modal physical models:

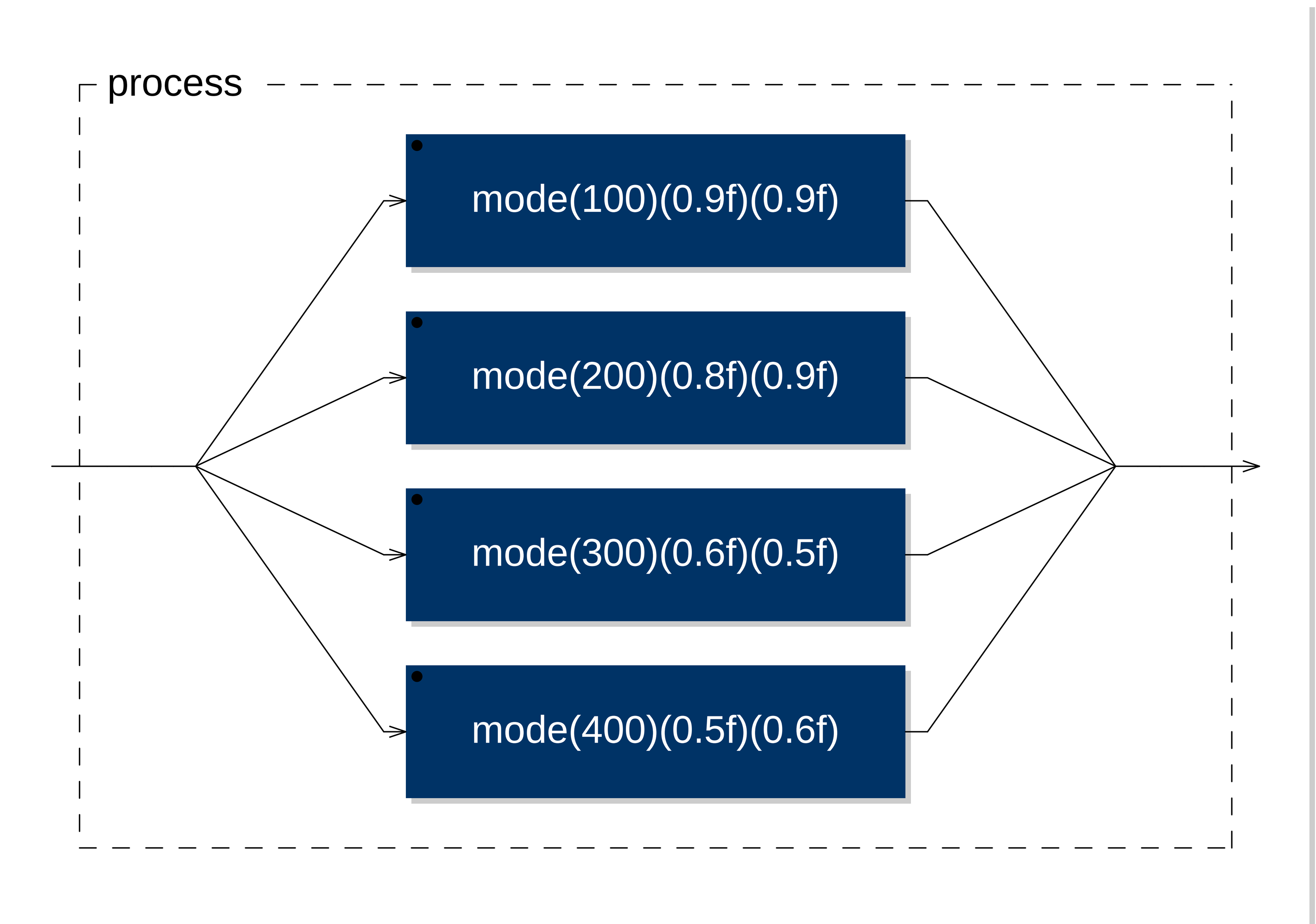
1. `ir2dsp.py` takes the audio file of an impulse response and converts it into a Faust program implementing the corresponding modal physical model;
2. `mesh2dsp.py` outputs the same type of model from a `.stl` file specifying a 3D object.

Faust Modal Physical Model

Linear percussion instruments can be implemented using banks of resonant bandpass filters [2]. Each filter implements one mode of the system and is configured with 3 parameters : the frequency of the mode, its gain and its resonance duration (`t60`). Its Faust version, `modeFilter` below, uses a biquad filter (`tf2`) and computes its poles and zeroes for a given frequency and `t60`.

```
modeFilter(f,t60) = tf2(b0,b1,b2,a1,a2)
with{
     b0 = 1;
     b1 = 0;
     b2 = -1;
     w = 2*PI*f/SR;
     r = pow(0.001,1/float(t60*SR));
     a1 = -2*r*cos(w);
     a2 = r^2;
};
mode(f,t60,gain) = modeFilter(f,t60)*gain;
```

Modal physical models are implemented using multiple parallel (`par` in Faust) instances of `mode` calls. The Faust-generated block diagram corresponding to such an implementation is presented below (we used arbitrary parameters here).



Such a model can be excited by a filtered noise impulse.

Future Directions

We plan to improve `ir2dsp.py` by using a better `t60` measurement algorithm. For now, the calculation is done by measuring the bandwidth for each peak, while it would be a better approach to extract it from a time-frequency representation of the signal.

Regarding `mesh2dsp.py`, we would like to try other open-source packages than Elmer to carry out FEA.

`ir2dsp.py` and `mesh2faust`

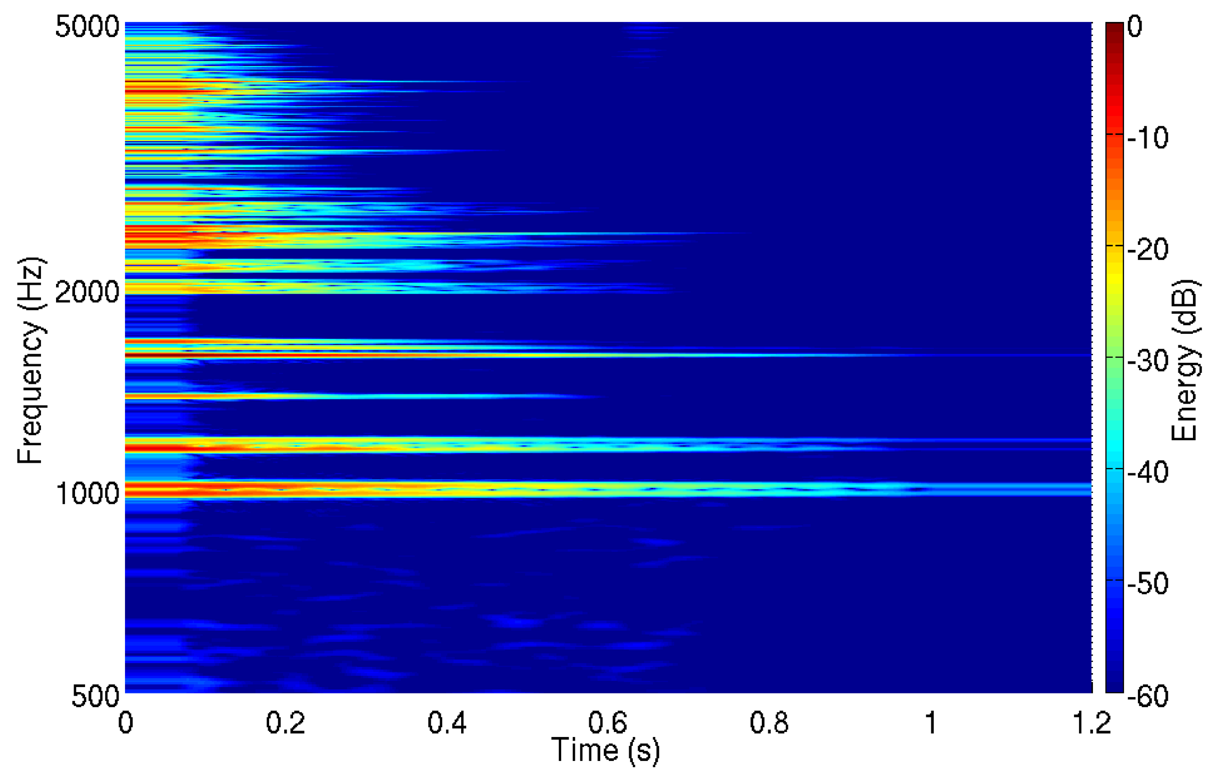
`ir2dsp.py` takes an audio file and extracts modal physical model-based information for each mode: frequency and gain, by carrying out peak detection; `t60`, by measuring bandwidth at -3 dB. A Faust file is then generated. With this tool, one can strike any object, record the resulting sound and turn it into a playable digital instrument.

`mesh2dsp.py` gives the same output, using a `.stl` file (describing a 3D object) as input, as follows:

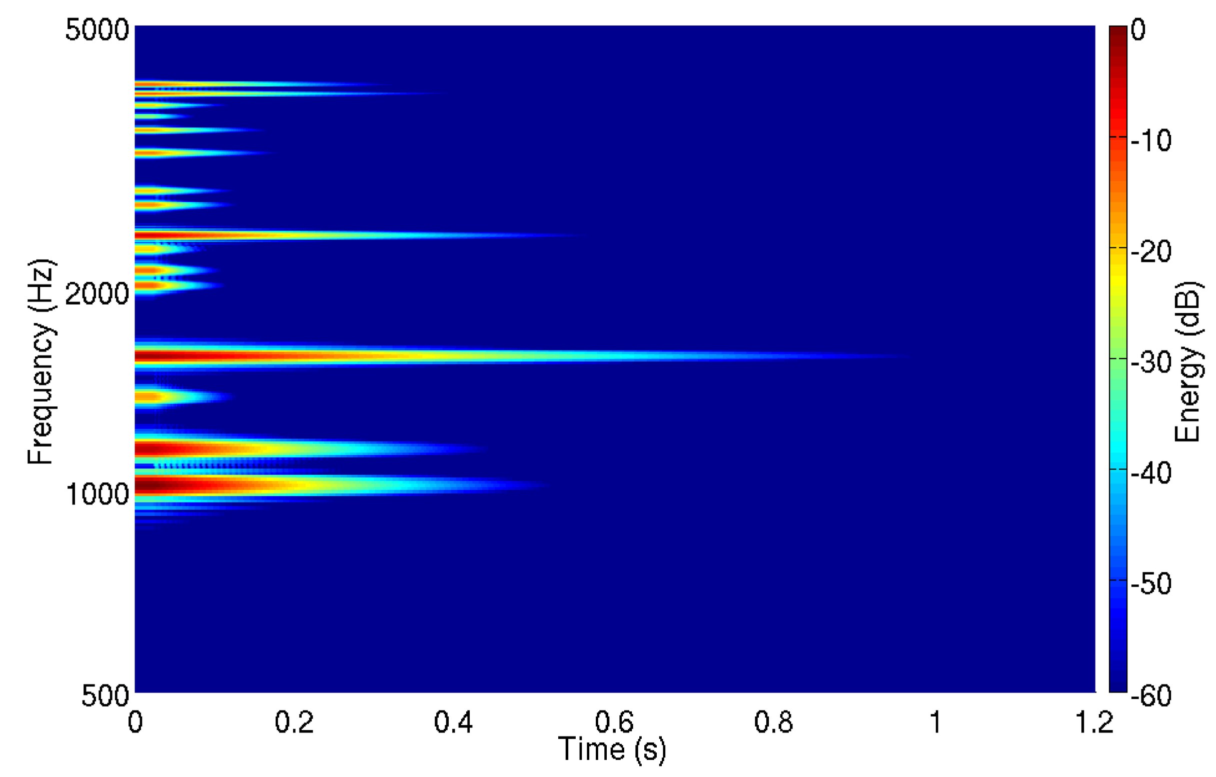
- conversion of the input object to a mesh;
- Finite Element Analysis (FEA) using the Elmer package, with the Young modulus, Poisson coefficient and density of the material as parameters:
- frequency and gain computation from eigenvalues and mass participation for each mode;
- `t60` values input (these values cannot be computed by this method unfortunately, so they are user-provided parameters).

Evaluation

Spectrograms of (a) the recording of the IR of a can and (b) its `ir2dsp.py`-generated modal physical model:



(a)



(b)

The original and synthesized sound representations are relatively close (but see Future Directions).

Artifacts

Source code available at: `https://github.com/rmichon/pmFaust/`

References

[1] R. Michon, J. O. Smith. *Faust-STK: a set of linear and nonlinear physical models for the Faust programming language*. In Proceedings of the DAFx-11 Conference, 2011

[2] J. O. Smith. *Physical Audio Signal Processing for Virtual Musical Instruments and Digital Audio Effects*. W3K Publishing, 2010

[3] J.-M. Adrien. *The Missing Link: Modal Synthesis*. In "Representations of Musical Signals", MIT Press, 1991

[4] Y. Orlarey, D. Fober, S. Letz. *Syntactical and Semantical Aspects of Faust*. Soft Computing, 2004

Project funded by ANR FEEVER. Linux Audio Conf., St-Etienne, May 18-21, 2017.