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

Pierre-Amaury Grumiaux[1], Romain Michon[2], Emilio Gallego Arias[1], and Pierre Jouvelot[1]
[1]MINES ParisTech, PSL Research University, France
[2]CCRMA, Stanford University, USA

Abstract

We present a set of tools to quickly implement modal physical models in the FAUST programming language. Models can easily be generated from any impulse response or 3D graphical representation of a physical object.

This system targets users with little knowledge in physical modeling and willing to use this type of synthesis technique in a musical context.

Keywords

Physical Modeling Synthesis, Faust Language, Digital Signal Processing

1 Introduction

The FAUST programming language has proven to be well suited to implement physical models of musical instruments [Michon and Smith, 2011] using waveguides [Smith, 2010] and modal synthesis [Adrien, 1991].

In this short paper, we present two Python scripts[1] allowing to easily generate FAUST modal physical models: `ir2dsp.py` and `mesh2dsp.py`.

- `ir2dsp.py` takes an audio file containing an impulse response as its main argument and converts it into a FAUST file implementing the corresponding modal physical model.
- `mesh2dsp.py` outputs the same type of model but takes an `.stl`[2] file containing the specification of any 3D object designed with a CAD[3] program as its main argument.

FAUST programs generated by `ir2dsp.py` and `mesh2dsp.py` are ready to use and can be compiled to any of the FAUST targets (standalone applications, plug-ins, etc.).

After briefly describing these two tools, we'll evaluate them and provide directions for future works.

2 Faust Modal Physical Model

Any linear percussion instrument can be implemented using a bank of resonant bandpass filters [Smith, 2010]. Each filter implements one mode (a sine or cosine function) of the system and can be configured by providing three parameters: the frequency of the mode, its gain, and its resonance duration (*T60*).

Such a filter can be easily implemented in FAUST using a biquad filter (`tf2`) and by computing its poles and zeros for a given frequency (`f`) and *T60* (`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;
```

The `modeFilter` function can be easily applied in parallel in FAUST using the `par` operator to implement any modal physical model:

```
model =
      _ <:
      par(i,nModes,
          mode(freq(i),t60(i),gain(i)))
      :> _;
```

The FAUST-generated block diagram corresponding to this code, with `nModes = 4`, `freq(i)= 100*(i+1)`, and `(t60(i),gain(i))` as successively (0,9,0.9), (0.8,0.9), (0.6,0.5) and (0.5,0.6), can be visualized in Figure 1.

This type of model can be easily excited by a filtered noise impulse (see Figure 2). The cutoff frequency of the lowpass and highpass filters

[1] `https://github.com/rmichon/pmFaust/` – All URLs in this paper were verified on 07/04/2017.
[2] STereoLithography.
[3] Computer-Aided Design.

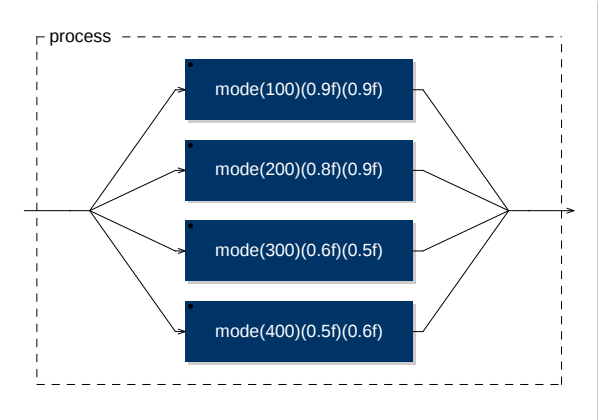

Figure 1: Block diagram of a FAUST modal physical model.

can be used to excite specific zones of the spectrum of the model and to choose the "excitation position." Since this system is linear, the same behavior could be achieved by scaling the gain of the different modes, but the filter approach that we use here will better integrate to our modular physical modeling synthesis toolkit, briefly presented in §6.

Figure 2: Excitation generator algorithm used to drive our modal physical models.

3 ir2dsp.py

`ir2dsp.py` takes an audio file containing an impulse response as its main argument. After performing the Fast Fourier Transform (FFT) on it, modes information is extracted by carrying out peaks detection. The *T60* of each mode is computed by measuring its bandwidth at -3 dB.

Modes information is formatted by `ir2dsp.py` to be plugged to a generic modal FAUST physical model similar to the one described in §2. The output of the Python program is a ready-to-use FAUST file implementing the model.

The goal of this tool is not to create very accurate models but rather to be able to strike any object (e.g., a glass, a metal bar, etc.), record the resulting sound, and turn it into a playable digital musical instrument.

4 mesh2dsp.py

The output of `mesh2dsp.py` is the same as `ir2dsp.py` (see §3), but it takes a `.stl` file as its input instead of an impulse response. `stl` is a common format supported by most CAD programs to export the description of 3D objects.

After converting the provided `.stl` file into a mesh, `mesh2dsp.py` performs a Finite Element Analysis (FEA) using Elmer [4] Various parameters such as the Young Modulus, the Poisson Coefficient, and the density of the material of the object must be provided to carry out this task.

The result of the analysis is a set of eigenvalues and mass participations for each mode. Eigenvalues are then converted to mode frequencies and mass participations to mode gains. Unfortunately, this technique doesn't allow to calculate the *T60* of the modes which can be configured by the user directly from the FAUST program.

5 Evaluation

To evaluate the accuracy of `ir2dsp.py`, we recorded the impulse response of a can and generated its corresponding modal physical model. Figure 3 shows the spectrogram of the impulse response of the can and Figure 4 the spectrogram of the impulse response of the physical model generated by `ir2dsp.py`. `ir2dsp.py` was configured to detect peaks at a minimum value of -20 dB and at least 100 Hz spaced from each other. We see that the synthesized sound is pretty close to the recorded version. *T60*s are not perfectly accurate since they were calculated by measuring the bandwidth of the mode. Tracking their evolution in the time domain would provide better results; thus we plan to use this technique in the future instead.

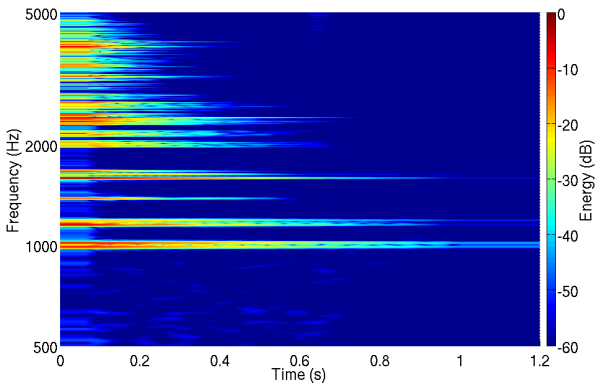

Figure 3: Spectrogram of an impulse response of a can

[4] `https://www.csc.fi/web/elmer/`

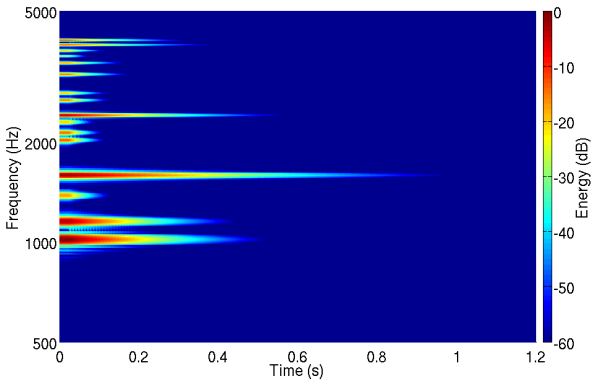

Figure 4: Spectrogram of the output of the modal model generated with `ir2dsp.py` from a can IR

`mesh2dsp.py` was tested with the geometric 3D model of a solid bar and provided good results.

6 Future Work

This work has been carried out as part of a larger project on designing a physical modeling toolkit for the FAUST programming language. `ir2dsp.py` and `mesh2dsp.py` will be integrated to it.

We plan to improve `ir2dsp.py` by using a better *T60* measurement algorithm. Indeed, the *T60* of each mode is currently computed by measuring its bandwidth after taking the FFT of the entire impulse response. A better approach would be to extract this information from a time-frequency representation of the signal (i.e., spectrogram), which would be more accurate.

Finally, we would like to try other open-source packages than Elmer to carry out the FEA in `mesh2dsp.py` to get better results and to smooth its integration in our FAUST physical modeling toolkit.

7 Conclusion

We presented a series of tools allowing to design at very high level ready-to-use physical models of musical instruments. Models can be generated from impulse responses or 3D graphical representations of physical objects.

While the models generated by this system are far from being accurate, we believe that it provides a convenient way for composers and musicians to design expressive custom instruments usable in a musical context.

8 Acknowledgements

Our thanks go to Yann Orlarey for his help with the use of Faust.

References

Jean-Marie Adrien. 1991. The missing link: Modal synthesis. In *Representations of Musical Signals*, chapter The Missing Link: Modal Synthesis, pages 269–298. MIT Press, Cambridge, USA.

Romain Michon and Julius O. Smith. 2011. Faust-STK: a set of linear and nonlinear physical models for the Faust programming language. In *Proceedings of the 14th International Conference on Digital Audio Effects (DAFx-11)*, Paris, France, September.

Julius Orion Smith. 2010. *Physical Audio Signal Processing for Virtual Musical Instruments and Digital Audio Effects*. W3K Publishing.