- HYPERCURVE -
An hybrid curve forge in Csound

Johann Philippe, Jacopo Greco d'Alceo*

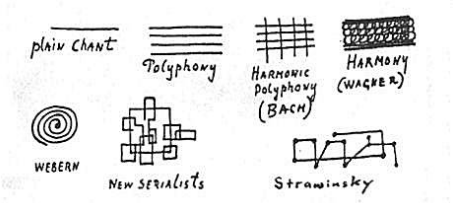
No institute
johannphilippe@lilo.org
jacopo.grecodalceo@gmail.com

Abstract. HYPERCURVE is a new library designed to combine different curve algorithms inside one function table. It has been thought as a tool for musicians looking to shape precisely their envelopes and function tables. The library is exposed to several environments, like *Csound, Faust, Lua and C++.*

Keywords: Curve, Perception, Csound, Shape, Envelope, Waveform, Control

1 Introduction

As described by Weber and Fechner[1], our sensation increases as the logarithm of a surge in energy. If we look inside the history of musical notation the idea of "line" or "curve" is almost always referred as a frequency parameter, where these geometrical terms stand for a visual counterpoint construction of harmony.



Look at the famous answer Stravinsky gave to Robert Craft [2] about a visual representation of his music.
How could we answer today?
Maybe we could add the graphical score of *Studie II* by Stockhausen where we can finally see more in detail how the sound is emerging in space-time, but we go further. Music evolution is directly consequent of notation, instruments and thinking - the more we dig into details, the more we find new paths.
This project is a study laboratory of curves in sound and music computing and it would like to propose a new and easy instrument to transform sound morphology and more.

* Thanks to the curve master François Roux, professor of electroacoustic music composition at National Conservatory of Music and Dance in Lyon.

[1] The so-called Weber–Fechner law.

2 Curves and musical expression

2.1 Curves and computer music

Since a lot of programs already give the opportunity to build and manipulate curves, an important question remains: why should someone create a new one? Most of the existing software rely on some usual algorithm without exploring the scientific world.
For example, Reaper[2] gives a lot of attention to the fade in and the fade out curves of volume and to the drawing automations. Yet, at this moment, Reaper has only six types of curve algorithm.
One of the main quality of the Csound GEN system is pointing out the difference and the advantage of constructing and experiencing a curve with ears rather than eyes - focusing on the singularity and the precision of the curve. However, GEN routines currently allow one curve algorithm per function table. But boundaries are built to be broken and that's where an open, simple and light-weight library of piecewise curves to use everywhere (i.e. in other media, too) is needed start making his own path in our discussions. Such an hybrid system has different advantages: sound samples management can be separated from curves data which can be directly manipulated with virtual function on-the-fly inside different environments and with a proper syntax. These functions give a different approach to composition and consequently different results on sound and push the limits where the most adventurous contemporary music is evolving.

2.2 A perceptual equation

Since the emancipation of the graphical score, the problem of the interpretation of curves has been transposed on a more visual question supposed to be resolved by an interpreter (Iannis Xenakis, Cornelius Cardew, etc.). Yet, by the time when computer becomes a creative tool that is integrated into the artistic process, we could hint a new vision. The deeper we dig in a mathematical and natural system, the more we discover about new ways to hear and compose. As far as today we can appreciate the basic waveforms of a synthesizer, in further researches, we could imagine the recognition of some specific and peculiar curves in a pedagogical interest (e.g. a music theory of curves) and in artistic and musical composition. Directly controlling curves suggest giving more power to the genetic material of a sound, to its expanding resources and to add details on his synthesis and direction in time and space.

3 Technical overview

HYPERCURVE is a new library designed to create piecewise hybrid curves. It can be used to combine multiple curve segments - each based on a particular

[2] The famous DAW software: `https://www.reaper.fm/`.

curve algorithm - in a single curve table. In its first version, HYPERCURVE provides 20 different curve algorithms that can be combined as several segments inside a single curve table, as well as some mechanisms used to manipulate the function table data.
These *Hypercurves* are use case agnostic: any curve can be used for any purpose: wavetable, envelope, control function, data manipulation (...). This project was created with the idea that every curve algorithm has its own signature and can have a singular impact on the way we listen, depending on which parameters it is used to control. Thus, it can be thought as a powerful and very configurable composition tool.
HYPERCURVE follows two standard:

- It focuses on 2D horizontal curves. 3D or circular curves are not handled by the library.
- Each curve base algorithm must respect its `y` starting point and destination.

Fig. 1. A random hybrid curve generated with HYPERCURVE

3.1 Basic principles of *HYPERCURVE*

HYPERCURVE is written in *C++*, and is available as a set of *Csound* opcode, a *Faust* FFI interface, a *Lua* module, and directly through the *C++* code. While its core is independant from frontends, *HYPERCURVE* has been ported to *Csound* using the *Csound Plugin Opcode SDK* [3].
Its design is pretty lightweight and simple, since most of the available functions in frontends are only an interface to the core library. There are three main types of components:

- `hypercurve`: an hybrid curve table

- `segment`: describes a component of an hypercurve
- `curve base`: the curve algorithm used to build a segment. Each curve algorithm function ends with `_curve` (e.g. `hc_cubic_curve`).

It has been decided to make the syntax as explicit as possible, so it can be easy to use and write complex curves. In *Csound*, *HYPERCURVE* opcodes always start with `hc_` prefix.

Any HYPERCURVE *can be described in* Csound *with the following syntax:*

```
icurve = hc_hypercurve(itable_number, isize, iystart,
            hc_segment(ifrac, idest1, hc_diocles_curve(0.51)),
            hc_segment(ifrac, idest2, hc_cubic_curve()))
```

We see that the definition of an `hypercurve` necessarily uses three main components.

- `hc_hypercurve(itable_number, isize, iystart, segment1, [segment2, ...])`
  - `itable_number`: The index of the table. Like with `ftgen`, the table number for `hc_hypercurve` can be set to 0 to allow automatic indexing.
  - `isize`: The number of samples of the curve.
  - `iystart`: The `y` starting point of the curve.
  - `segment_list`: The list of segments is currently limited to 64 segments, which should be enough for most musical purposes (or could be increased in the future).
- `hc_segment(ifrac, idest, curve_base)`
  - `ifrac`: The fractional size of the segment (between 0 and 1). Usually, the sum of all fractional sizes should be 1. Though, *HYPERCURVE* automatically checks and resizes those fractional values if the sum is not exactly 1.
  - `idest`: The destination of the segment in the `y` axis.
  - `curve_base`: The algorithm used to interpolate points of the segment.
- `hc_cubic_curve` is one of the available curve base algorithms. The available algorithms are listed in the documentation [6].

3.2 Hypercurves examples

HYPERCURVE syntax aims to be as expressive as possible. Thus, it focuses more on a textual description, rather than numeric parameters only, as used by `ftgen` and `f` statements.
The following hypercurve is composed of two segments. It can be written in Csound with the following code.

```
gicrv1 = hc_hypercurve(0, 16384, 0,
            hc_segment(0.5, 1, hc_diocles_curve(0.51)),
            hc_segment(0.5, 0, hc_tightrope_walker_curve(1.1,0.1)))
```

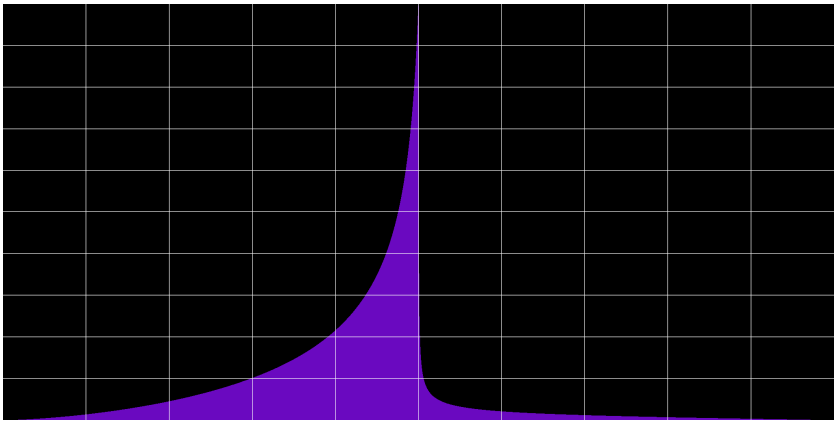


Fig. 2. Diocles cissoid and tightrope walker segments

The curve will be displayed in the terminal with a similar ASCII display as the on used for GEN function tables. It is also possible to write it as a png file with `hc_write_as_png` to get a better visual representation, as shown above.

Let's try a more complex one, with three segments.

```
gicrv2 = hc_hypercurve(0, 16384, 0,
            hc_segment(0.1, 1, hc_tightrope_walker_curve(1.1,0.1)),
            hc_segment(0.4, 0.2, hc_gaussian_curve(1, 1)),
            hc_segment(0.5, 0, hc_kiss_curve()))
```

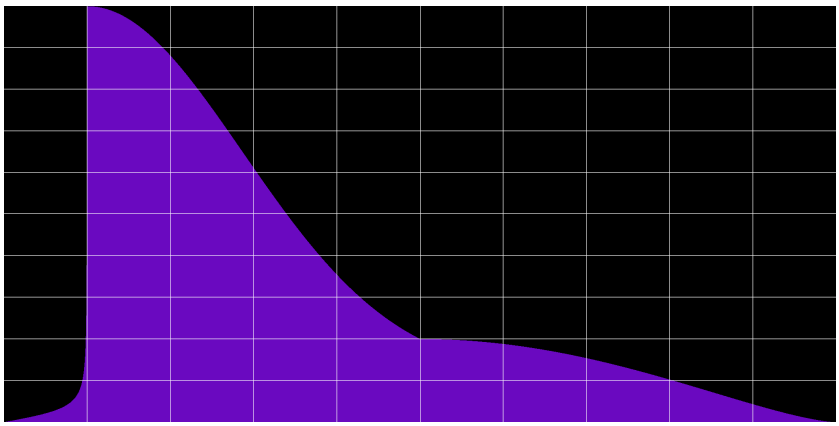


Fig. 3. Tightrope walker, gauss and kiss segments

Each `curve_base` is different and may take a different set of parameters. Those arguments are described in the documentation [6].

These curves, once created, can be used as any *function table* in *Csound*. Any *Csound* opcode using GEN function tables can be passed an *hypercurve* instead. A simple use case would be to use them directly as envelopes, reading the curve with `tablei` opcode.

```
kenv = tablei:k( linseg(0, p3, 1), gicrv2, 1)
asig = vco2(0.3, p4) * kenv
outs asig, asig
```

When an hypercurve goes out of scope (if it is defined inside an instrument for example), its memory will be freed by Csound `FtDelete`.

3.3 Curve manipulation tools

While HYPERCURVE was first thought as a way to create hybrid and finely shaped curves, it quickly appeared that it could benefit from manipulation tools, allowing to combine, concatenate, rotate and mirror curves. HYPERCURVE provides standard mathematical operators `+ - * /`, which are implemented in Csound as opcodes: `hc_add, hc_sub, hc_mult, hc_div`. Those opcodes will create a new hypercurve where each sample is the result of the operation on the two source curves.

In order to manipulate the scale of a curve, *HYPERCURVE* provides the `hc_normalize` opcode, which might be useful when using unsafe curve algorithms (such as `hc_cubic_spline_curve` or `hc_polynomial_curve`). It also provides a way to invert or mirror a curve algorithm with `hc_mirror` and `hc_invert`.

```
imirrored = hc_hypercurve(0, 1024, 0,
                hc_segment(1, 1, hc_mirror(hc_cubic_curve())))
```

In this example, the `hc_mirror` function is applied to the curve algorithm to allow the cubic segment to be convex instead of concave. This function applies a symmetry through a linear axis joining the starting point of the segment and its destination. The `hc_invert` opcode applies a vertical symmetry.

This part of the library allowing to manipulate curves will be extended in the near future, as we currently search for new intuitive and creative ideas and algorithms to transform hypercurves.

4 Future of *HYPERCURVE*

The future development of this project is focused on two axis:

– Add more curve algorithms to provide a more complete and wide tool. This axis has itself three branches. First, we plan to implement more existing curve algorithms. Secondly, we would work on new curves approximations

based on real world. Last but not least, some more curves could just be abstractly created, using a graphic calculator to invent some new shapes and curve behaviors.
– Add more manipulation functions, in order to make it a real musical curve forge. Some abstract manipulations are in the road map, like a virtual three-dimensional transformation, and some utility functions.

There also are a few pending interrogations, mostly related on the core design of *HYPERCURVE* in *Csound* implementation. Among those, one could mention the automatic normalization of hypercurves in *Csound*, that would prevent dangerous behaviors - since some curve algorithms behavior can be difficult to predict (for example user defined polynomial curves or cubic spline curve).

5 The References Section

References

1. Fechner, G.: Element der Psychophysik, Breitskopf and Härtel, Leipzig (1860)
2. Craft, R., Stravinsky, I.: Conversations with Igor Stravinsky., Faber & Faber, (2011)
3. Csound Plugin Opcode SDK, `https://github.com/csound/opcode_sdk`
4. Mathcurve website, `https://mathcurve.com/`
5. Hypercurve Github, `https://github.com/johannphilippe/hypercurve`
6. Hypercurve Documentation, `https://github.com/johannphilippe/hypercurve/tree/main/doc`