# Creating Modulation Matrices for Modern Synthesizers and Effects in Csound and Cabbage

Jonathan Walter[*]

Berklee College of Music
jonwaltermusic@gmail.com

**Abstract.** When designing synthesizers and effects often the most important and characteristic component is its modulation capabilities. When looking at commercial hardware such as pedals by Chase Bliss, essentially every parameter can be modulated internally, as well as controlled via MIDI and CV. Alternatively, hardware such as Sequential's Prophet 6 offers limited customizable modulation, which helps establish the synthesizer's signature sound. Internal modulation presents an interesting problem as one doesn't want to limit the modulation capabilities of the user, while also not sacrificing CPU. One solution is to build all-encompassing internal routing matrices into your plugins. The following paper will explain how to create a routing matrix for synthesizers and effects inspired by the one in Ableton's wavetable synthesizer.

**Keywords:** Csound, Cabbage, ntrpol, Modulation, Matrix, Routing, VCA, Plugins

## 1 Introduction

It is important to keep in mind the different software that a plugin may be used in and design it to be flexible enough to suit the needs of each. For instance, users of Ableton Live Suite have access to Max for Live devices that perform modulation (such as low frequency oscillators and envelope generators) that can be mapped to individual parameters of any plugin. In this case, if a plugin offers limited internal modulation options, the user can simply add as many of these devices as they need to achieve the sound they are looking to create. Other DAWs don't have this flexibility so it is important to consider how many layers of internal modulation to add. As far as external modulation, MIDI control is easily achievable through the various MIDI-related opcodes that Csound contains, and customizable MIDI control can be added to this modulation matrix. Regardless of the DAW being used, all users will be able to automate parameters, however this can be slower for the workflow of the user and can be less musical than internal routing. By integrating a routing matrix the individual sound of the plugin can be crafted.

[*] Thanks to Dr. Boulanger.

## 2 Scaling and Planning

In order to determine how to implement and scale modulation, it is important to decide on the desired effect. By default LFOs are bipolar, ranging from -1 to 1, so typically they are implemented through addition, like in vibrato.

```
alfo lfo 1, 6
kcps = kcps + (alfo * 0.1 * kcps)
```

This effect modulates the destination parameter (in this case pitch) above and below its chosen value. Alternatively LFOs can be scaled to be unipolar, with the following equation. This equation assumes the LFO amplitude to be set to 1:

```
alfo = (alfo + 1) * 0.5
```

If using an amplitude other than 1, the equation can be said to be:

```
alfo = (alfo + kamp) * (0.5 * kamp)
```

Once the LFO is unipolar, the modulator may be applied through either multiplication or addition, depending on the desired results. One use case for applying an LFO through multiplication would be to modulate volume. In this case the LFO would modulate between full volume and silence.

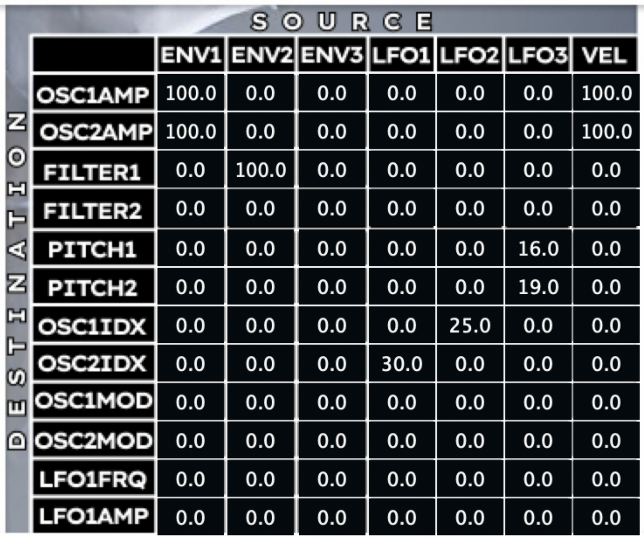
Envelope generators also typically have a range of 0 to 1 which usually does not have to be scaled. If using a standard ADSR style envelope generator, the sustain parameter can be used to scale the lowest value. More often than not envelope generators would be applied through multiplication, as they would be plugged into VCA's in a hardware system. This however is not a hard set rule and envelope generators can be scaled and/or added to parameter values if that is the desired effect. For instance if the envelope is being used to modulate pitch, it may be helpful to add a small value so that the pitch will never reach 0. Here is an example of this use-case.

```
aenv madsr iatk, idec, isus, irel
aenv = aenv + 0.01
kcps = kcps * aenv
```

Csound also offers many other opcodes that can be used for modulation such as `randh` and `loopseg`, and the same principles of scaling apply to these.

## 3 Design and Implementation

This matrix design is based off of the modulation matrix from Ableton's wavetable software synthesizer, with modulation sources on the X axis and destinations on the Y axis. Within this grid each cell is filled with an `nslider` in Cabbage, displaying floating point values between 0 and 100, as shown in figure 1.



| | ENV1 | ENV2 | ENV3 | LFO1 | LFO2 | LFO3 | VEL |
|---|---|---|---|---|---|---|---|
| OSC1AMP | 100.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 100.0 |
| OSC2AMP | 100.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 100.0 |
| FILTER1 | 0.0 | 100.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| FILTER2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| PITCH1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 16.0 | 0.0 |
| PITCH2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 19.0 | 0.0 |
| OSC1IDX | 0.0 | 0.0 | 0.0 | 0.0 | 25.0 | 0.0 | 0.0 |
| OSC2IDX | 0.0 | 0.0 | 0.0 | 30.0 | 0.0 | 0.0 | 0.0 |
| OSC1MOD | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| OSC2MOD | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| LFO1FRQ | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| LFO1AMP | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

**Fig. 1.** UI application of this style of routing matrix

This routing matrix implementation interpolates between a base "off" signal of 0 or 1 (0 if the modulation is based on addition, and 1 if it is based on multiplication). This interpolation is simple to implement using Csound and Cabbage and can be done using the following lines of code:

```
kLFOtoPCH chnget "LFOtoPCH"
kLFOtoPCH = kLFOtoPCH / 100
aLFOtoPCH lfo 1, 6
aLFOtoPCH ntrpol a(0), aLFOtoPCH * (0.25 * kcps), kLFOtoPCH / 100
kcps = kcps + aLFOtoPCH
```

In this example an LFO is used to modulate pitch, which can be used to create a simple vibrato effect. The first line here receives a value from the Cabbage UI object, which is used as the "interpolation point" [2] between the base "off" signal and the full signal of the modulator. This "point" value is scaled down to 0 to 1 as otherwise it would intensify the amplitude of the modulator between the values of 1-100. The `nslider`'s have a range of 0 to 100 simply for the user's experience but this scaling could easily be bypassed by setting their range to 0 to 1 in the first place. The maximum range of the modulator signal can also be scaled at this point, which is done in this example, with the amplitude of the LFO being scaled by a fraction of the frequency of the note being played. This gives the vibrato an even level of pitch modulation across low and high frequencies. The base value of the interpolation is set to 0 because the modulation is applied through addition. This model can be scaled up simply by adding more modulators and copying the procedure above.

```
alfo lfo 1, 6
aenv madsr iatk, idec, isus, irel
kLFOtoPCH chnget "LFOtoPCH"
kENVtoPCH chnget "ENVtoPCH"
kLFOtoPCH = kLFOtoPCH / 100
kENVtoPCH = kENVtoPCH / 100
aLFOtoPCH ntrpol a(0), alfo * (0.25 * kcps), kLFOtoPCH / 100
aENVtoPCH ntrpol a(0), aenv + 0.01, kENVtoPCH / 100
kcps = (kcps + aLFOtoPCH) * aENVtoPCH
```

In this example both an LFO and an envelope are used to modulate pitch. When applying multiple modulators with varying mathematical applications (combination of addition and multiplication), it is important to pay attention to the order of operations. Typically all addition should be applied first within parentheses, followed by all multiplication, however different effects can be achieved by ordering these in different ways.

## 4 Enhancements and Modifications

Some modifications can be made to better the user's experience and make the plugins act more musically, such as scaling each `nslider` exponentially. This can be useful in many musical applications when a smaller amount of modulation produces the best results, and allows for more precision within this lower range. There are a couple ways of doing this, either before or after receiving the values from the channel. The following example plugs the interpolation point value into an exponential equation.

```
kLFOtoPCH chnget "LFOtoPCH"
kLFOtoPCH = kLFOtoPCH / 100
kLFOtoPCH = ((20^kLFOtoPCH)-1)/(20-1)
```

The value of "20" can be replaced in both locations with a higher number for a steeper curve, or a lower number (limited to greater than 1) for a flatter curve. Please note this number cannot be equal to or less than 1, as this will result in dividing by a zero or a negative number. This can also be done by adjusting the skew value in the range argument of the `nslider`. This is the fourth parameter of the argument, and setting it to 0.5 creates an exponential response, and it can be set in this way[1]:

```
nslider channel("LFOtoPCH") range(0, 100, 100, 0.5, 0.1)
```

The exponential response may feel more natural in the first example, as it applies to the slope of parameter control within a DAW and does not affect the way the UI objects interact visibly.

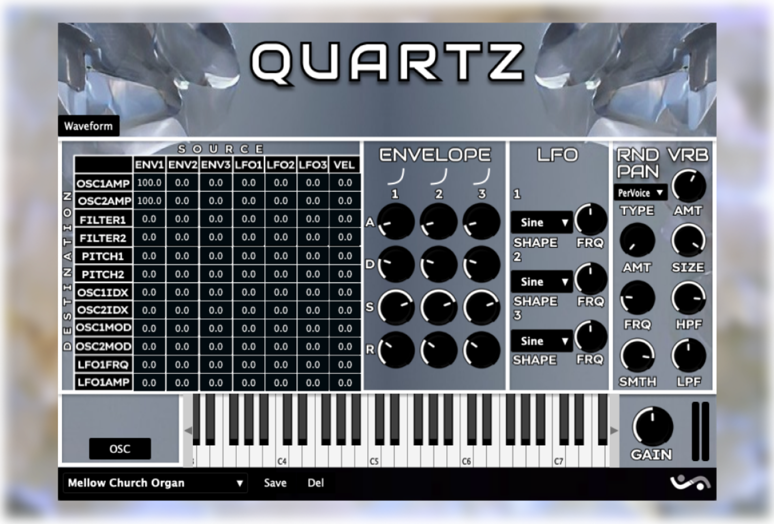
## 5 Conclusion

This matrix implementation can also be adapted in many ways. MIDI controls such as velocity and aftertouch can be added to the matrix using Csound's wide range of MIDI related opcodes and plugging their values into the same interpolation process as shown before. In this example aftertouch is routed to modulate the cutoff frequency of a low-pass filter.

```
kAFTtoFLT chnget "AFTtoFLT"
midichannelaftertouch kafter
kAFTtoFLT ntrpol k(0), kafter * 20, kAFTtoFLT / 100
aout butterlp ain, kctf + kAFTtoFLT
```

This matrix can also be adapted to create macros, modulating multiple parameters from the same source and adjusting the amount of modulation with one knob. It is worth noting the matrix in Ableton's wavetable synthesizer is capable of both normal and inverted modulation, allowing the user to select a value between -100 and +100 for the modulation point value. The matrix design here is able to do this but more scaling is needed, and has not been explored yet.

The plugin shown in figure 2 implements this style of routing matrix into an FM synthesizer. [1]



SOURCE

| DESTINATION | ENV1 | ENV2 | ENV3 | LFO1 | LFO2 | LFO3 | VEL |
|---|---|---|---|---|---|---|---|
| OSC1AMP | 100.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| OSC2AMP | 100.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| FILTER1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| FILTER2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| PITCH1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| PITCH2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| OSC1IDX | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| OSC2IDX | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| OSC1MOD | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| OSC2MOD | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| LFO1FRQ | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| LFO1AMP | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

Mellow Church Organ

**Fig. 2.** Routing matrix within an FM synthesizer plugin

[1] Two plugins using this matrix design can be downloaded here: www.jon-walter.com/icsc

There are limitless ways to customize this matrix design to suit the needs of a specific plugin and add as many or as few sources and destinations as the creator desires.

## References

1. Cabbage Docs, `https://cabbageaudio.com/docs/sliders/`
2. Csound Manual, `http://www.csounds.com/manual/html/ntrpol.html`