A modulation matrix for complex parameter sets

Øyvind Brandtsegg
Music Technology
NTNU, Department of Music
NO-7491 Trondheim
oyvind.brandtsegg@ntnu.no

Sigurd Saue
Music Technology
NTNU, Department of Music
NO-7491 Trondheim
sigurd.saue@ntnu.no

Thom Johansen
Q2S Centre of Excellence
NO-7491 Trondheim
thomj@alumni.ntnu.no

ABSTRACT
The article describes a flexible mapping technique realized as a many-to-many dynamic mapping matrix. Digital sound generation is typically controlled by a large number of parameters and efficient and flexible mapping is necessary to provide expressive control over the instrument. The proposed modulation matrix technique may be seen as a generic and self-modifying mapping mechanism integrated in a dynamic interpolation scheme. It is implemented efficiently by taking advantage of its inherent sparse matrix structure. The modulation matrix is used within the Hadron Particle Synthesizer, a complex granular module with 200 synthesis parameters and a simplified performance control structure with 4 expression parameters.

Keywords
Mapping, granular synthesis, modulation, live performance

1. INTRODUCTION
Digital musical instruments allow us to completely separate the performance interface from the sound generator. The connection between the two is what we refer to as *mapping*. Several researchers have pointed out that the expressiveness of digital musical instruments really depends upon the mapping used, and that creativity and playability are greatly influenced by a mapping that motivates exploration of the instrument (see e.g. [1]). In fact, experiments presented by Hunt et al [10] indicate that "complex mappings can provide quantifiable performance benefits and improved interface expressiveness". Rather than simple one-to-one parameter mappings between controller and sound generator, complex many-to-many mappings seem to promote a more holistic approach to the instrument: "less thinking, more playing".

The importance of efficient mapping strategies becomes obvious when controlling sound generators with a large number of input parameters in a live performance context. An interesting strategy proposed by Momeni and Wessel employs geometric models to characterize and control musical material [12], inspired by research on multidimensional perceptual scaling of timbre [9][16]. They argue that high-dimensional sound representations can be efficiently controlled by low-dimensional geometric models that fit well with standard controllers such as joysticks and tablets. Typically a small number of desirable sounds are represented as high-dimensional parameter vectors (e.g. the parameter set of a synthesis algorithm) and associated with specific coordinates in two-dimensional gesture space. When navigating through gesture space new sounds are generated as a result of an interpolation between the original parameter sets weighted by their relative distance in the space. Spatial positions can easily be stored and recalled as presets, and the gesture trajectories lend themselves naturally to automation.

We have adopted their strategy in a complex digital musical instrument designed for live performance with granular synthesis [2]. This particular synthesis engine [6] offers a wide range of time-based granular synthesis techniques found in Curtis Roads book *Microsound* [15]. The synthesis model requires some 40 parameters, but in addition we add modulation and effects for a total of over 200 parameters. The parameter vector not only represents a specific sound, but also contains information on how manual expression controllers and internal modulators may influence the sound. As a result navigation in gesture space actually modifies the mapping and hence changes the instrument itself.

This concept relates to Momeni's discussion of modal and non-modal mappings [13]. The former refers to mappings where the same gesture produces a variety of different results depending on instrument mode. Modal mappings provide richer control, but introduce state-dependent actions that may confuse the performer. In our case the modes are not discrete configurations, but rather a continuum of possible instrument states controlled and interpolated from the same gesture space as the sound itself.

The key mechanism for integrating this rich behavior into the mapping is the *dynamic modulation matrix*. The general idea is to regard all control parameters as modulators of the sound generator, and to do all mapping in one single matrix. The modulation matrix defines the interrelations between modulation sources and synthesis parameters in a very flexible fashion, and also allows modulation feedback. The entire matrix is dynamically changed through interpolation when the performer navigates gesture space.

In this paper we develop the concept of modulation matrix with simple examples. We then describe a particular implementation of the matrix within the audio programming language CSound. Finally we present the Hadron Particle Synthesizer as an example of a live performance instrument combining geometric interpolation and the modulation matrix.

2. THE MODULATION MATRIX
The origin of the modulation matrix is found in the patch bays of old analogue synthesizers, where patch cords interconnected the various synthesizer modules. In 1969 the English company Electronic Music Studios (EMS) [8] introduced the VCS3 with a unique matrix patch system to replace the clutter of patch wires (see front left panel in Figure 1 under). Signal routing was accomplished by placing small pins into the appropriate slots in the matrix.

NIME'11, 30 May–1 June 2011, Oslo, Norway.

Figure 1. The EMS VCS3 synthesizer [8]

The concept is still common in software synthesizers and plug-ins. A relevant example is the Matrix Modular 3 synthesizer from Native Instruments [14] that offers a granular synthesis module, an integrated sequencer, various modulators and a modulation matrix that links them all together (Figure 2). The matrix is configurable, but there is no mechanism to dynamically interpolate from one matrix configuration to another, which is a cornerstone in what we propose.

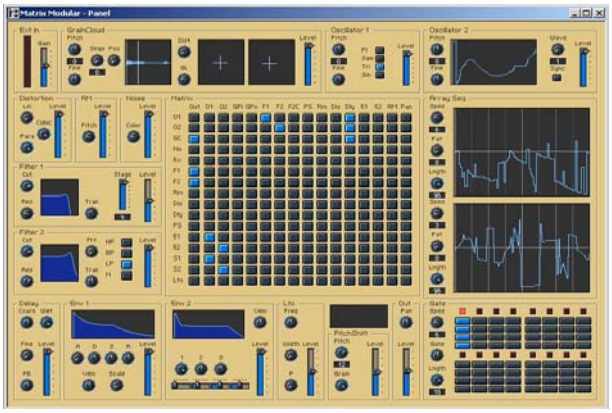

Figure 2. Native Instruments Matrix Modular 3 [14]

Software-based matrix modulation typically includes a list of modulation *sources*, a list of modulation *destinations*, and "slots" for each possible connection between source and destination. As an extension of the straightforward routing in classic patch bays software matrices often provide two controls between modulation source and destination:

- *Scaling coefficient*: The amount of modulation source reaching the destination.
- *Initial value*: An initial, fixed modulation value.

3. THE *MODMATRIX* OPCODE

Our implementation of the modulation matrix is available as an *opcode*[1] in the audio processing language CSound, with the name *modmatrix* [5]. The opcode computes a table of output values (destinations) as a function of initial values, modulation variables and scaling coefficients. The i'th output value is computed as:

$$out_i = in_i + \sum_k (g_{ki} * m_k)$$

where out_i is the output value, in_i is the corresponding initial value, m_k is the k'th modulation variable and g_{ki} is the scaling coefficient relating the k'th modulation variable to the i'th output.

In the following three sections we will provide some basic examples of *modmatrix* configurations. The modulator variables are assumed to be in the range 0.0 to 1.0 for unipolar signals (e.g. the signal from a user interface slider, called *manual expression control* here), and -1.0 to 1.0 for bipolar signals (e.g. the signal from an LFO). Amplitudes in the examples are assumed to be in range 0.0 to 1.0, and frequency values are given in Hz.

3.1 Simple modulator mapping

As a simple example of a modulation matrix we will use 2 parameters and 2 modulators. Each parameter has an initial value (the parameter value before modulation is applied), and the influence of each modulator signal to a parameter is computed by adding the modulator value to the parameter value. A scaling coefficient is applied to each modulator signal at each matrix mixing point (see Figure 3). With the specific coefficients used here the LFO will add a value of 0.4 to the oscillator amplitude when the LFO is at its peak value (if the LFO is bipolar, 0.4 will be subtracted when the LFO is at its minimum value). Oscillator frequency will also be affected by the LFO, with a maximum offset of 40Hz from the original oscillator frequency. The manual expression control affects oscillator amplitude (with a maximum offset of 0.6), and to a small degree also affects oscillator frequency (max offset of 5 Hz).

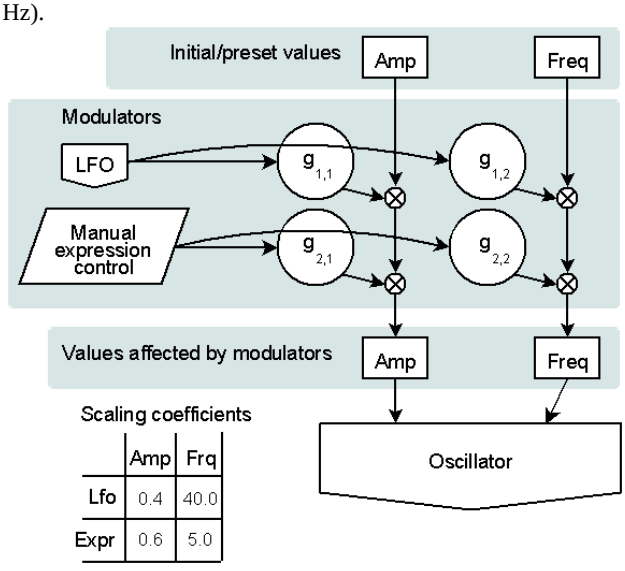

Figure 3. Parameters (Amp/Freq) for a simple oscillator modulated by one LFO and one manual expression control.

3.2 Modulator mapping with feedback

In this example, the parameter set has been extended to include amplitude and frequency for the LFO, and we enable modulator feedback in the matrix (see Figure 4). The modulator mappings to oscillator amplitude and frequency are the same as in the previous example. The LFO output affects its own frequency (by a maximum deviation of 7 Hz), and the manual expression control affects the LFO amplitude by a maximum offset of 0.4. Modulator feedback is known to be used in systems such as the

[1] An opcode is a basic CSound module that either generates or modifies signals.

Sytrus softsynth from Image Line [11]. Modulator feedback implies that some modulation sources may take the role as modulation destinations as well and we get self-modifying behavior controlled by the modulation matrix. As with any other kind of feedback, modulator feedback must be applied with caution.

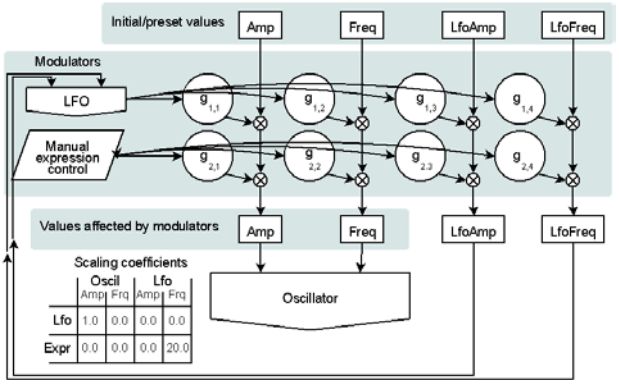

Figure 4. Modulation matrix with 4 parameters and 2 modulators. As some of the parameters are used in the synthesis of modulator signals, we have modulation feedback.

3.3 Dynamically modified mapping

As we operate with scaling coefficients stored in a table, we can dynamically alter the mapping in the modulation matrix by manipulating the values in the coefficient table. We can of course explicitly write values to the table, but more interesting: we can interpolate between different mapping tables.

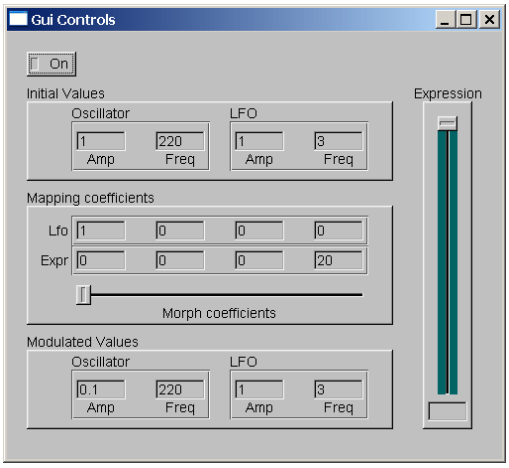

Figure 5. User interface for a simple synthesizer with a morphable modulation matrix [4].

In the following example [4] we will create a simple synthesizer with a modulation matrix as shown in Figure 4. We will enable interpolation between different tables of modmatrix coefficients, thereby morphing the mapping of the modulators. Our synthesizer GUI will have two user control sliders, one *expression* slider and one *morphing* slider. When moving the morphing slider, the modulation mapping will be dynamically changed. This changes how the LFO affects the sound, and it also changes the effect of the expression slider. To simplify our example all parameter values (oscillator amp and frequency, LFO amp and frequency) are fixed. Both the initial and the modulated values of these four parameters are shown in the user interface display (see Figure 5). A numerical example is shown in Figure 6.

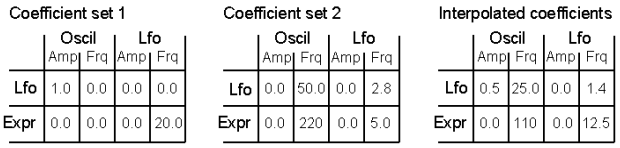

Figure 6. Interpolation between two tables of modulation matrix coefficients.

3.4 Implementation details

Due to the potentially large number of modulation sources for which modmatrix is designed to be used, optimization is a big concern. Fortunately, modulation matrices are typically sparsely populated, since only a small number of sources will be connected to any particular destination. Together with the fact that a given matrix is usually stationary between preset morphing states, this is an obvious way to improve performance.

There are several ways to deal with sparse matrices algorithmically, but most of these do not efficiently utilize the SIMD[2] capabilities present in the CPUs of all modern computers, so we have decided to apply a straightforward, but efficient method instead. Each time the synthesis environment knows that a preset interpolation, or any other activity altering the modulation matrix, is complete, it will signal modmatrix that the matrix is going into a temporarily constant state. The matrix will then be scanned for properties which can be eliminated, being for example entire rows and/or columns containing zeroes. A new modulation matrix will then be built lacking the redundant entries in question. From then on, the usual multiply and accumulate operations needed by a modulation matrix will be performed, skipping modulators and modulation targets which need not be taken into account. All modulation will be performed using this reduced matrix until the modulation matrix is again changed.

As long as the modulation matrix is undergoing change, the entire matrix is processed as is, still utilizing efficient SIMD processing. Looking into ways of efficiently leveraging the still sparse nature of the matrix in this morphing state is an area of future improvement, should the current method prove too inefficient.

Figure 7. Graphical user interface for the Hadron Particle Synthesizer.

4. THE HADRON PARTICLE SYNTHESIZER

As a more developed example of the modulation matrix, we will show how it's been utilized in the Hadron Particle Synthesizer[3]. Hadron is a complex granular synthesis device

[2] Single Instruction Multiple Data. Matrix computations will in most cases benefit measurably from use of these facilities.

[3] Hadron is freely available as a Max for Live device (march - 11) and as a VST plugin (fall 2011). It can be downloaded from www.partikkelaudio.com

with approximately 200 synthesis parameters. The underlying synthesis engine was built with a primary focus on flexibility of sound processing. The high level of flexibility also led to high complexity in configuration and control of the device. A simplified control structure was developed to allow real-time performance with precise control over the large parameter set using just a few user interface controls (see). The modulation matrix is essential to link simplicity of control to the complexity of the parameter set in this instrument.

4.1 Hadron internals

The basic parameters of granular synthesis can be considered to be *grain rate*, *grain pitch* and *grain shape*, as well as the audio *waveform* inside each grain. Hadron allows mixing of 4 source waveforms inside each grain, with independent pitch and phase for each source. The source waveforms can be recorded sounds or live audio input. The grain rate and pitch can be varied at audio rate to allow for frequency modulation effects, and displacement of individual grains allows smooth transitions between synchronous and asynchronous granular techniques. To enable separate processing of individual grains, a grain masking system is incorporated, enabling "per grain" specification of output routing, amplitude, pitch glissandi and more. A set of internal audio effects (ring modulators, filters, delays) allow further processing of individual grains. The Hadron Particle Synthesizer also utilizes a set of modulators for automation of parameter values. The modulators are well known signal generators, e.g. low frequency oscillators, envelope and random generators. In addition, audio analysis data for the source waveforms are used as modulator signals. Within Hadron, any signal that can affect a parameter value is considered a modulator, so signals from midi note input and the 4 manual expression controls (Figure 7) also counts as modulators. There is also a set of programmable modulator transform functions to allow waveshaping, division, multiplication and modulo operations on modulator signals.

The full parameter set for Hadron currently counts 209 parameters and 51 modulators. The granular processing requires "only" about 40 of these parameters, and a similar amount of parameters are used for effects control. The largest chunk of parameters is actually the modulator controls (e.g. LFO amplitude, LFO frequency, Envelope attack etc.) with approximately 100 parameters. All parameters and modulators are treated in one single modulation matrix with size 209 x 51.

4.2 Hadron control

Hadron makes use of a preset interpolation system, in many ways similar to techniques explored by Momeni and Wessel [12], but with some modification. We use static positioning of the presets. A preset is placed in each corner of a 2D "joystick" control surface. Another difference is that a preset not only contains parameter values, but also modulator mapping coefficients. This means that the effect of e.g. an LFO can change gradually from one preset to another. Similarly, the manual expression controls will have different effects in different presets. For example, if *Expression 1* controls *grain pitch* in one preset, it may control *grain rate* in another. Moreover, since the modulation matrix allows flexible one-to-many mappings and also nonlinear mapping curves via the modulator transform functions; both the routing and the scaling of a modulator may change between presets. The presets are manually designed to meet specific needs using a custom design tool, but the parameter space could possibly be explored using techniques similar to those suggested by Dahlstedt [7].

5. CONCLUSION

The article describes a flexible mapping technique realized as a many-to-many dynamic mapping matrix. A generalization of all control signals to be used as modulators allows for full flexibility of routing and mapping. Computationally efficient implementation of the modulation matrix allows practical use of large parameter and modulator sets. Dynamic mapping is achieved by interpolating mapping coefficients in the modulation matrix. Dynamic mapping can be combined with geometric models for parameter vector interpolation to create an instrument with simple controls, complex mapping and modal behavior. The complex mapping is not a goal in itself, but rather a result of the complexity of the parameter vector one wishes to achieve detailed control over. This complexity may lead to a higher learning threshold for the digital instrument, since the expression controls do not have fixed labels (like pitch bend, filter cutoff etc.). The lack of cognitive labeling of the instrument controls can be confusing for an unskilled performer, but as is the case with all musical instruments, practice is needed to achieve familiarity and skill. In fact, the lack of cognitive labeling may force a more intuitive approach to the instrument with an enhanced focus on listening. Our experiments on performance [3] using this mapping technique in the Hadron Particle Synthesizer shows that it can be used as a means of effective and intuitive instrumental control.

6. REFERENCES

[1] Arfib, D., Couturier, J., Kessous, L. and Verfaille, V. Strategies of mapping between model parameters using perceptual spaces. *Organised Sound 7*, 2 (2002): 127-144

[2] Brandtsegg, Ø. and Saue, S. Particle synthesis, a unified model for granular synthesis. Accepted paper at Linux Audio Conference 2011

[3] Brandtsegg, Ø and Waadeland, C. H. Studio sessions, duo improvisation with percussion (CHW) and *Hadron* (ØB): http://soundcloud.com/brandtsegg/sets/little-soldier-joe

[4] Brandtsegg, Ø. Example available as a Csound csd file at http://oeyvind.teks.no/ftp/modmatrix-example/modmatrix-simple-example.csd

[5] CSound opcode *modmatrix*. See documentation at: http://www.csounds.com/manual/html/modmatrix.html

[6] CSound opcode *partikkel*. See documentation at: http://www.csounds.com/manual/html/partikkel.html

[7] Dahlstedt, P. Dynamic Mapping Strategies for Expressive Synthesis Performance and Improvisation, in *Proceedings of the Computer Music Modeling and Retrieval (CMMR) Conference*, Copenhagen 2008

[8] Electronic Music Studios (EMS). Homepage (no longer updated) at: http://www.ems-synthi.demon.co.uk/

[9] Grey, J.M. Multidimensional perceptual scaling of timbre. *Journal of Acoustical Society of America 61*, 5(1977): 1270-1277

[10] Hunt, A., Wanderley, M. and Paradis, M. The importance of parameter mapping in electronic instrument design. *Journal of New Music Research 32*, 4(2003): 429-440

[11] Image Line. Homepage at: http://www.image-line.com

[12] Momeni, A. and Wessel, D. Characterizing and controlling musical material intuitively with geometric models. In *Proceedings of the New Interfaces for Musical Expression Conference (NIME-03)* (Montreal, Canada, May 22-24, 2003). Available at http://www.nime.org/2003/onlineproceedings/home.html

[13] Momeni, A. *Composing instruments: Inventing and performing with generative computer-based instruments.* Ph.D. thesis, University of California, Berkeley, CA, 2005

[14] Native Instruments. Homepage at: http://www.native-instruments.com

[15] Roads, C. *Microsound*. MIT Press, Cambridge, MA, 2001

[16] Wessel, D. Timbre space as a musical control structure. *Computer Music Journal 3*, 2(1979): 45-52