# [DRAFT#1] Lyapunov Space of Coupled FM Oscillators

**Claude Heiland-Allen**
claude@mathr.co.uk

## Abstract

I consider two coupled oscillators, each modulating the other's frequency. This system is governed by four parameters: the base frequency and modulation index for each oscillator. For some parameter values the system becomes unstable. I use the Lyapunov exponent to measure the instability. I generate images of the parameter space, implementing the number crunching on graphics hardware using OpenGL. I link the mouse position over the displayed image to realtime audio output, creating an audio-visual browser for the 4D parameter space.

## Keywords

chaos, DSP, GPU

## 1 Introduction

In my 2005 *Soft Rock EP* and 2006 *Soft Rock DVD* I explored the transitions between order and chaos in coupled FM oscillators, implemented in Pure-data using GridFlow for visualization of the output waveforms over time. More recently I developed the idea to map the parameter space and in performance choose parameters on the basis of desired sound output.

I used one Pure-data batch mode instance per CPU core each sending analysis data to a realtime Pure-data instance. The analysis used various methods (including FFT for spectral statistics and the sigmund external for pitch tracking) to classify points into pitched (ordered, stable) or unpitched (chaotic, unstable) with measures of distortion or noisiness. Sadly this approach proved impractical as it achieved only tens of pixels per second, even with a fast multicore CPU, and porting these signal analysis algorithms to massively-parallel programmable graphics hardware seemed to be too difficult.

Seeing a bifurcation diagram produced by an analogue Moog synthesizer [Slater, 1998] and Lyapunov fractals [Elert, 2007], I decided to apply the latter technique to coupled FM oscillators in the digital realm.

## 2 Formulation

### 2.1 Coupled FM Oscillators

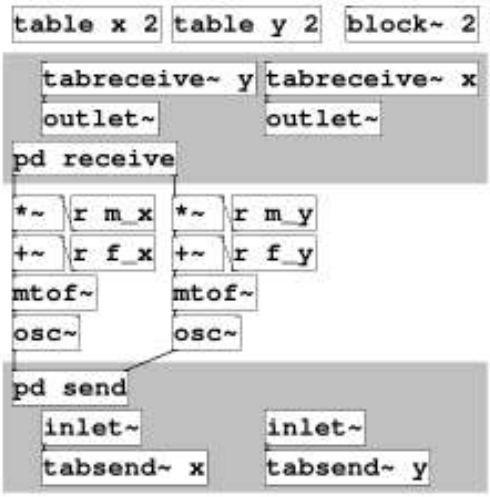



Figure 1: Coupled FM oscillators in Pure-data.

I formulate the two coupled oscillators in Figure 1 which each modulate the other's frequency by a mutual recurrence relation:

$$\begin{aligned} x_{n+1} &= \%(x_n + \mathrm{I}(f_x + m_x \cos(2\pi y_{n-d}))) \\ y_{n+1} &= \%(y_n + \mathrm{I}(f_y + m_y \cos(2\pi x_{n-d}))) \end{aligned} \quad (1)$$

where

$$\%(t) = t - \lfloor t \rfloor, \quad \mathrm{I}(t) = \frac{440}{\mathrm{SR}} 2^{\frac{t-69}{12}}$$

Here $x_n$, $y_n$ is the phase of each oscillator at time step $n$, $d$ is a delay measured in samples, $f_x$, $f_y$ is the base frequency of each oscillator as a MIDI note number, and $m_x$, $m_y$ is the modulation index of each oscillator as a MIDI note number.

I'll write the four-dimensional parameter space vector $a = (f_x, f_y, m_x, m_y)$, and the $(2d + 2)$-dimensional phase space vector $z = (x_n, y_n, x_{n-1}, y_{n-1}, \ldots, x_{n-d}, y_{n-d})$. I'll fix $d = 1$ and $\mathrm{SR} = 48000$.

### 2.2 Lyapunov Exponents

The Lyapunov exponent $\lambda$ measures divergence in phase space:

$$|z_1(t) - z_0(t)| \approx e^{\lambda t} |z_1(0) - z_0(0)|$$

$$\lambda = \lim_{t\to\infty} \lim_{z_1(0)\to z_0(0)} \frac{1}{t} \log \frac{|z_1(t) - z_0(t)|}{|z_1(0) - z_0(0)|} \quad (2)$$

An attracting orbit has $\lambda < 0$ and a divergent (chaotic) orbit has $\lambda > 0$.

The wrapping of phase into $[0, 1)$ requires a modified norm:

$$|z|_{\%} = \sqrt{\sum_i \min(\%(z_i), 1 - \%(z_i))^2}$$

I'll compute approximations to $\lambda$ in Equation 2 by iterations of Equation 1.

### 2.3 Viewing Planes

An image is 2D, which requires choosing a subset of the 4D parameter space to visualize. I chose two particular planes:

$$A_+(a_0, r_0) = a_0 + r_0 \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

$$A_-(a_0, r_0) = a_0 + r_0 \begin{pmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} \quad (3)$$

where $(u, v)$ is the coordinates of the pixel, $a_0$ is the centre of the view, and $r_0$ is the radius of the view.

## 3 GPU Implementation

I implemented imaging of the Lyapunov space of coupled FM oscillators on graphics hardware using OpenGL and OpenGL Shading Language.

### 3.1 Computation Overview

To render an image I first fill a texture with $(u, v)$ coordinates using a framebuffer object and a fragment shader. I copy this texture to a vertex buffer object, interleaving the initial phase space vector $z = (0, 0, 0, 0)$ and Lyapunov exponent statistics vector $l = (0, 0, 0, 0)$.

Using a vertex shader, I repeatedly compute rough estimates of the Lyapunov exponent using Equation 2 by perturbing $z_1(0) = z_0(0) + \delta$ with $\delta$ small and iterating Equation 1 $t = 256$ times, having calculated $a$ from $(u, v)$ using Equation 3. I discard the first few repetitions and those resulting in $-\infty$, and accumulate the results (sum, count) in $l$.

Between each repetition I compact the working set using a geometry shader, plotting points whose mean Lyapunov exponent estimate changed very little during the previous step. I keep only the others to refine further. This directs the computational effort on the points that need it most: those that are slow to converge.

### 3.2 Noise Increases Stability

At the end of each repetition I keep $z_1$ instead of $z_0$. This effectively adds a small amount of noise, counter-intuitively increasing stability. Noise allows more of the phase space to be explored, and makes it more likely for the perturbed orbit to reach an attracting part of the phase space.

### 3.3 Dither Increases Quality

To reduce grid sampling artifacts, I perturb $(u, v)$ within the bounds of its corresponding pixel before calculating the $a$ parameter vector for each repetition.

## 4 Interactive Browsing

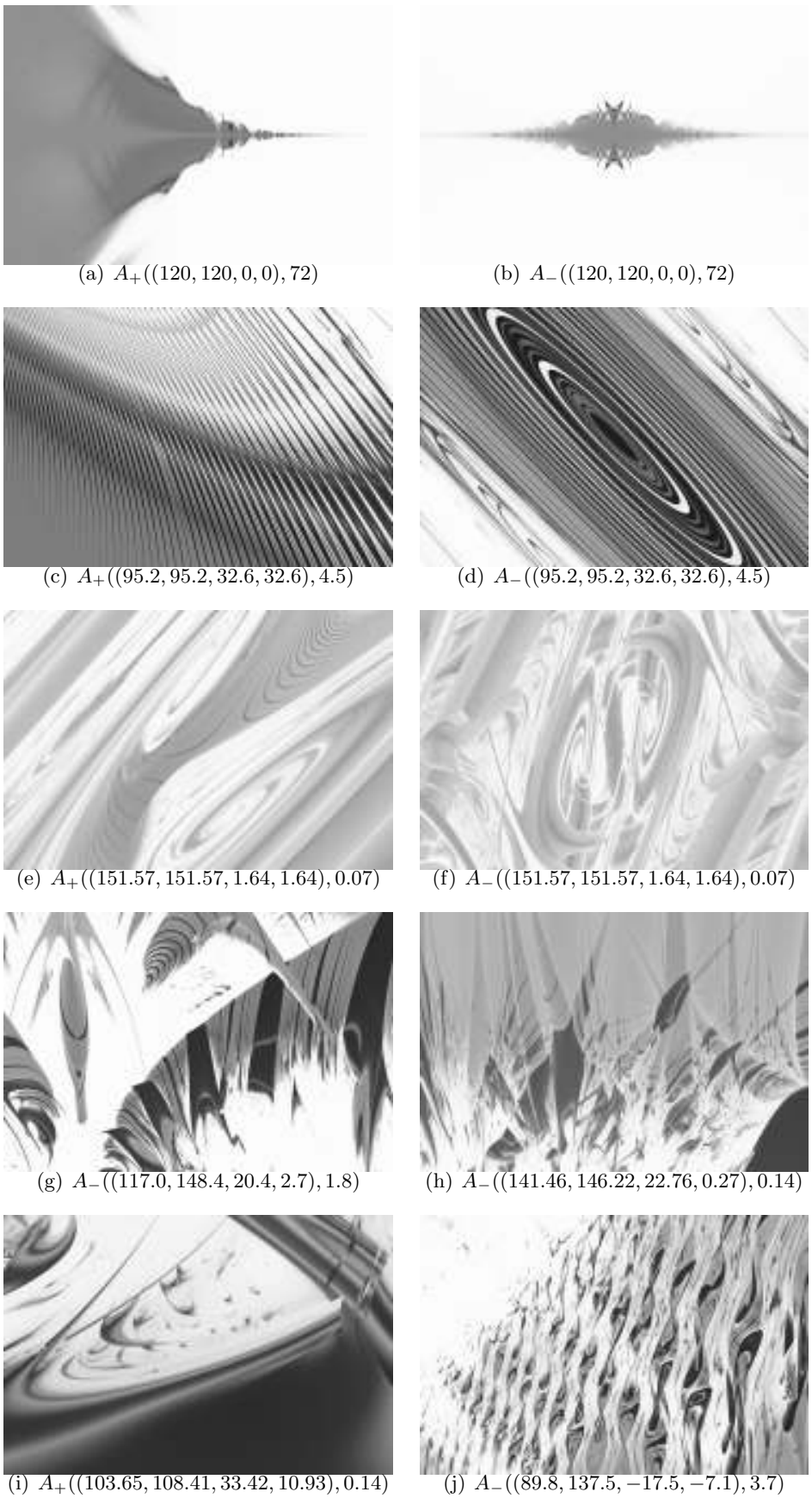
### 4.1 Zoomable Visual Interface

I implemented the graphical user interface using GLUT. Clicking with the mouse zooms the view about the point clicked on. The left button (or scroll up) zooms in, the right button (or scroll down) zooms out, with the middle button centering the view on the target point. Pressing the TAB key toggles between the $A_+$ and $A_-$ planes in Equation 3, and F11 toggles full screen operation.

To ensure user interface responsiveness, the computation is amortized over several frames. I divide the target frame period by the measured time for one repetition to compute how many repetitions to perform each frame. The repetitions-per-frame increases as the working set becomes smaller.

### 4.2 Realtime Audio Output

While the GPU simulates and analyses one oscillator pair per pixel, the CPU simulates one oscillator pair with $a$ determined from the pixel under the mouse pointer. I move the mouse pointer to hear what different parts of the image sound like. The image acts as a map, a reference frame for chosing parameters to audition. I implemented the audio using JACK.



(a) $A_+((120, 120, 0, 0), 72)$ (b) $A_-((120, 120, 0, 0), 72)$

(c) $A_+((95.2, 95.2, 32.6, 32.6), 4.5)$ (d) $A_-((95.2, 95.2, 32.6, 32.6), 4.5)$

(e) $A_+((151.57, 151.57, 1.64, 1.64), 0.07)$ (f) $A_-((151.57, 151.57, 1.64, 1.64), 0.07)$

(g) $A_-((117.0, 148.4, 20.4, 2.7), 1.8)$ (h) $A_-((141.46, 146.22, 22.76, 0.27), 0.14)$

(i) $A_+((103.65, 108.41, 33.42, 10.93), 0.14)$ (j) $A_-((89.8, 137.5, -17.5, -7.1), 3.7)$

Figure 2: Example images. Darker shades are stable, lighter shades chaotic.

## 5 Examples

Figure 2(a) is the initial view on starting the interactive browser. Low frequencies to the left are stable even at high modulation index away from the central axis. High frequencies to the right become chaotic at progressively lower modulation index. (b) shows the $A_-$ plane at the same location. When $f_x = f_y$ and $m_x = m_y$ the $A_+$ plane has mirror symmetry about its horizontal axis, and the $A_-$ plane has two-fold rotational symmetry about its centre.

(c) shows bands alternating between stability and chaos. The bands become distorted and collapse as the modulation index and frequency increase. (d) shows its $A_-$ plane, bands become rings.

When the frequency is greatly increased, the shapes become more intricate. (e) exhibits spirals of stability, with similar spirals in the $A_-$ plane in (f).

Breaking the symmetry and setting $f_x \neq f_y$ or $m_x \neq m_y$ leads to diverse forms. In particular (i) has shapes that resemble those of Lyapunov space images of the logistic map.

## 6 Conclusions

### 6.1 OpenGL Issues

The current implementation is hardcoded with delay $d = 1$ and would be very awkward to generalize. OpenGL architecture limits each vertex attribute to four components with the maximum number of attributes typically limited to sixteen. This totals 64 floats per vertex, 6 of which are needed for the pixel coordinates and Lyapunov exponent statistics accumulation. Therefore using OpenGL imposes a limit $d < 28$. For comparison my original experiments in *Soft Rock EP* used Pure-data's default block size of 64, with $d = 32$. Moreover, increasing $d$ increases video memory consumption. With the maximum $d = 27$, browsing at $1920 \times 1080$ resolution would require over 1GB.

My future work on this project will look into using OpenCL, which provides a heterogenous CPU and GPU computation framework. I hope it will avoid the inherent awkwardness of abusing OpenGL shaders to perform calculations.

### 6.2 Audio Issues

While the implementation works as intended, unfortunately with $d = 1$ the sound is nowhere near as rich and varied as with $d = 32$. With small $d$ there is much more very high frequency content in interesting-looking regions. I've not found any regions of the parameter space with both interesting appearance and palatable audio frequencies at $d = 1$, while with high $d$ there are plenty of parameters that generate sounds that fluctuate intermittently between smooth tones and chaotic noise. But as I haven't been able to visualize with high $d$ I can't be confident that their neighbourhoods will look as interesting as they sound.

Unfortunately, heavy use of the GPU as in the interactive browser can block the operating system for too long and cause audible JACK xruns. This somewhat dampens my hopes of using the browser in a live situation.

### 6.3 Pretty Pictures

Despite these shortcomings, I think the images at least are beautiful. I plan to render a selection at high resolution and print postcards and posters. I'll divide the image plane into tiles and compute each tile in succession, finally combining the pieces into one large picture to makes it computationally feasible.

There is also scope for video work, moving and rotating the viewing plane through the 4D parameter space, with different shapes forming and collapsing over time. My rough benchmarks take 5-10 seconds per frame at $1920 \times 1080$, so for now I'll wait until faster cheaper graphics cards become available.

## 7 Acknowledgements

## 8 FIXME

Needs links to Soft Rock EP and DVD. Needs references for OpenGL and GLSL.

Needs more examples of Lyapunov exponents with orbit diagrams etc. Needs an example of the more direct computation of $\lambda$ for the logistic equation and explanation of why that approach isn't available (phase space dimension $> 1$ means derivatives are not simple).

Needs an overview of the OpenGL pipeline for those not familiar with it (diagram of shaders with inputs and outputs, vertex geometry fragment, transform feedback, what is a vbo fbo texture, etc).

Should the Computation Overview section use pseudocode instead of prose?

Needs links to code + example audio / video.

## References


Glenn Elert. 2007. *The Chaos Hypertextbook*.

Dan Slater. 1998. Chaotic sound synthesis. *Computer Music Journal*, 22(2):12–19.