Lyapunov Space of Coupled FM Oscillators

Claude Heiland-Allen

Linux Audio Conference 2013

Lyapunov FM

Background

Recurrence Relations
Behaviour Of Recurrences
Mapping Behaviours

Recurrence Relations

Recurrence
Phase Space
Parameter Space
Example: Logistic Map

Recurrence

Next state depends on current state.

$z(t+1) = F(z(t))$

Phase Space

Internal state that varies over time.
z(0) determines all future z(t).

Parameter Space

Constant over time.
Family of recurrences.

Example: Logistic Map

$z(t+1) = r\, z(t)\, (1 - z(t))$

1D phase space: $z(t)$

1D parameter space: r

Behaviour Of Recurrences

Orbits
Attractors
Stability
Chaos
Lyapunov Exponent

Orbits

The set of points reached from $z(0)$.
$\{ z(0), z(1), z(2), \ldots \}$

Attractors

Orbits may become periodic.
Many points can reach one attractor.

Stability

Nearby points are pulled closer together.

Chaos

Nearby points are pushed further apart.
Butterfly effect.

Lyapunov Exponent

Quantifies butterfly effect.

< 0 : stable

> 0 : chaotic

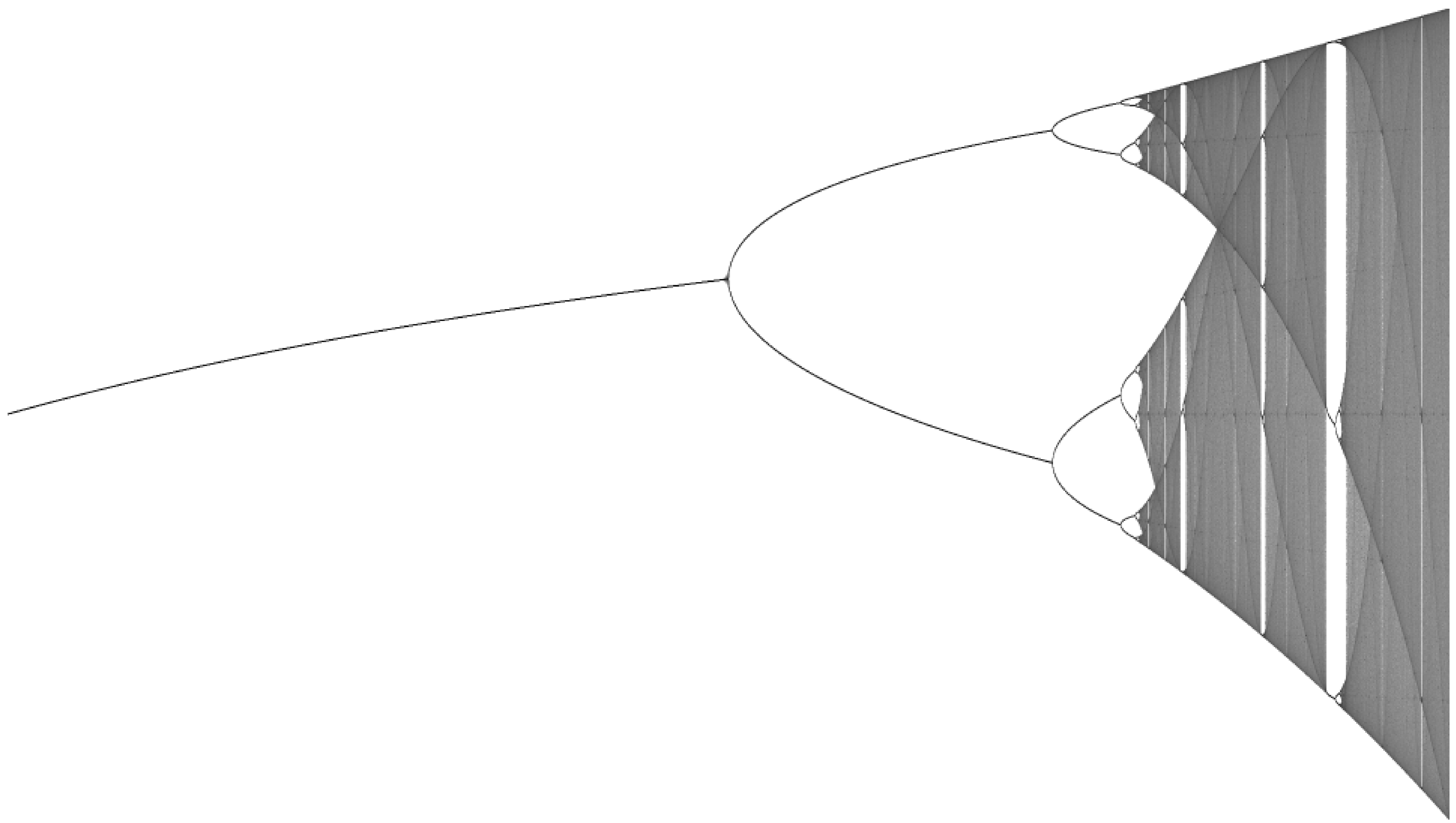
Mapping Behaviours

1D Logistic Map
1D Bifurcation Diagram
1D Lyapunov Space
2D Logistic Map
2D Lyapunov Space

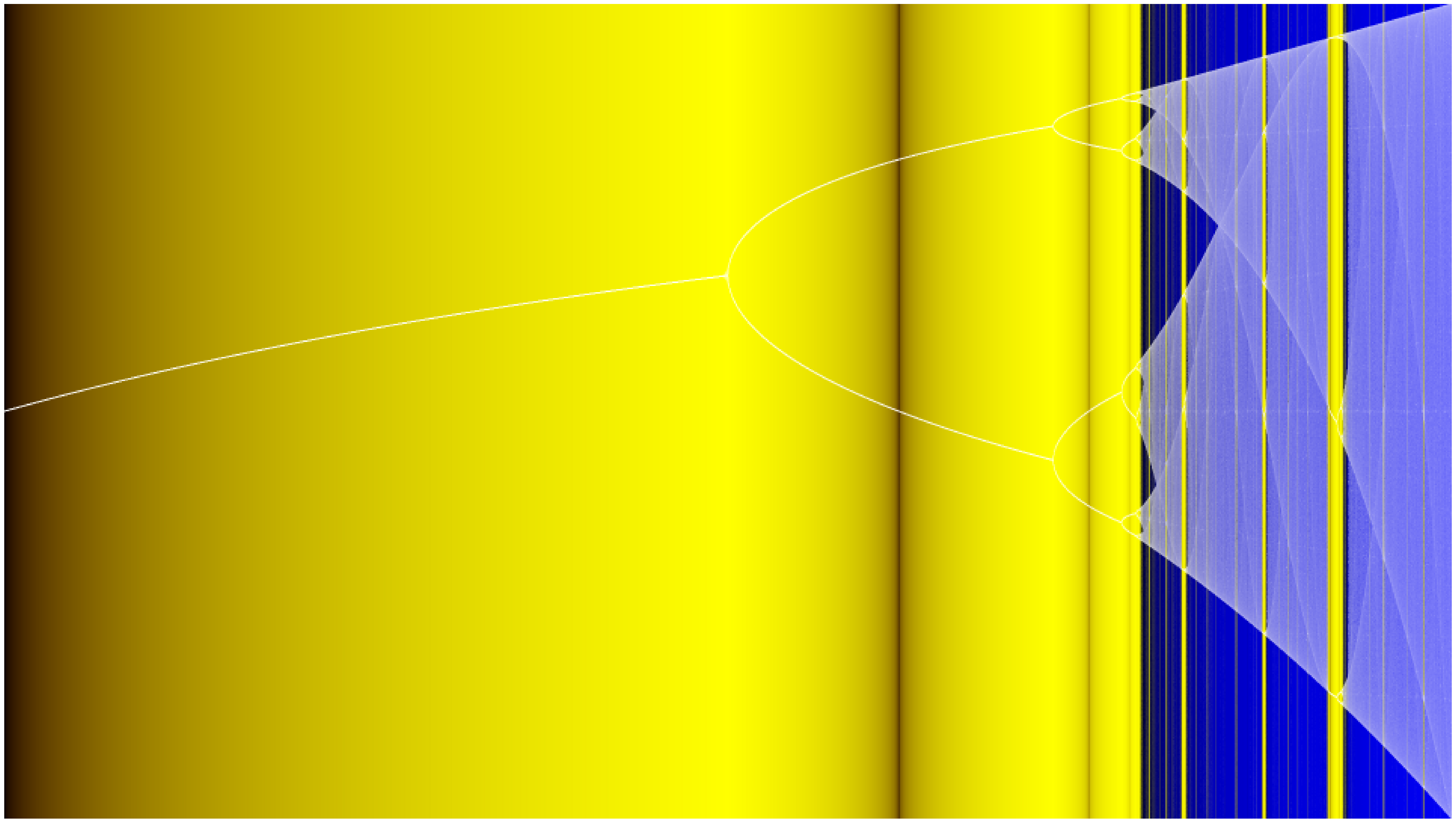
1D Logistic Map

z(t+1) = F(z(t))
F(x) = r x (1 - x)
1D phase space: z(t)
1D parameter space: r

1D Bifurcation Diagram



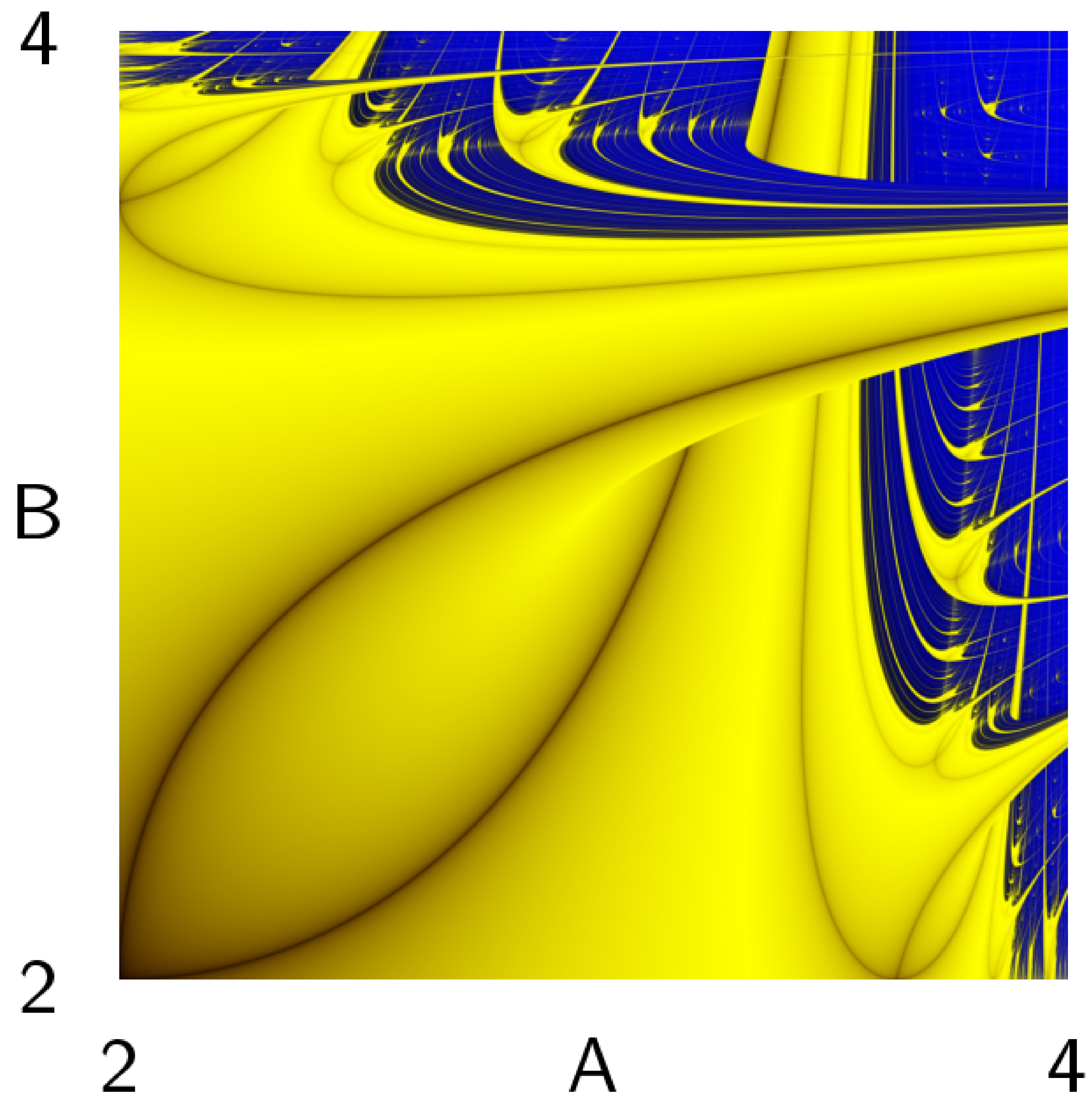
1D Lyapunov Space



2D Logistic Map

z(t+1) = G(F(z(t)))
F(x) = A x (1 - x)
G(x) = B x (1 - x)
1D phase space: z(t)
2D parameter space: (A,B)

2D Lyapunov Space



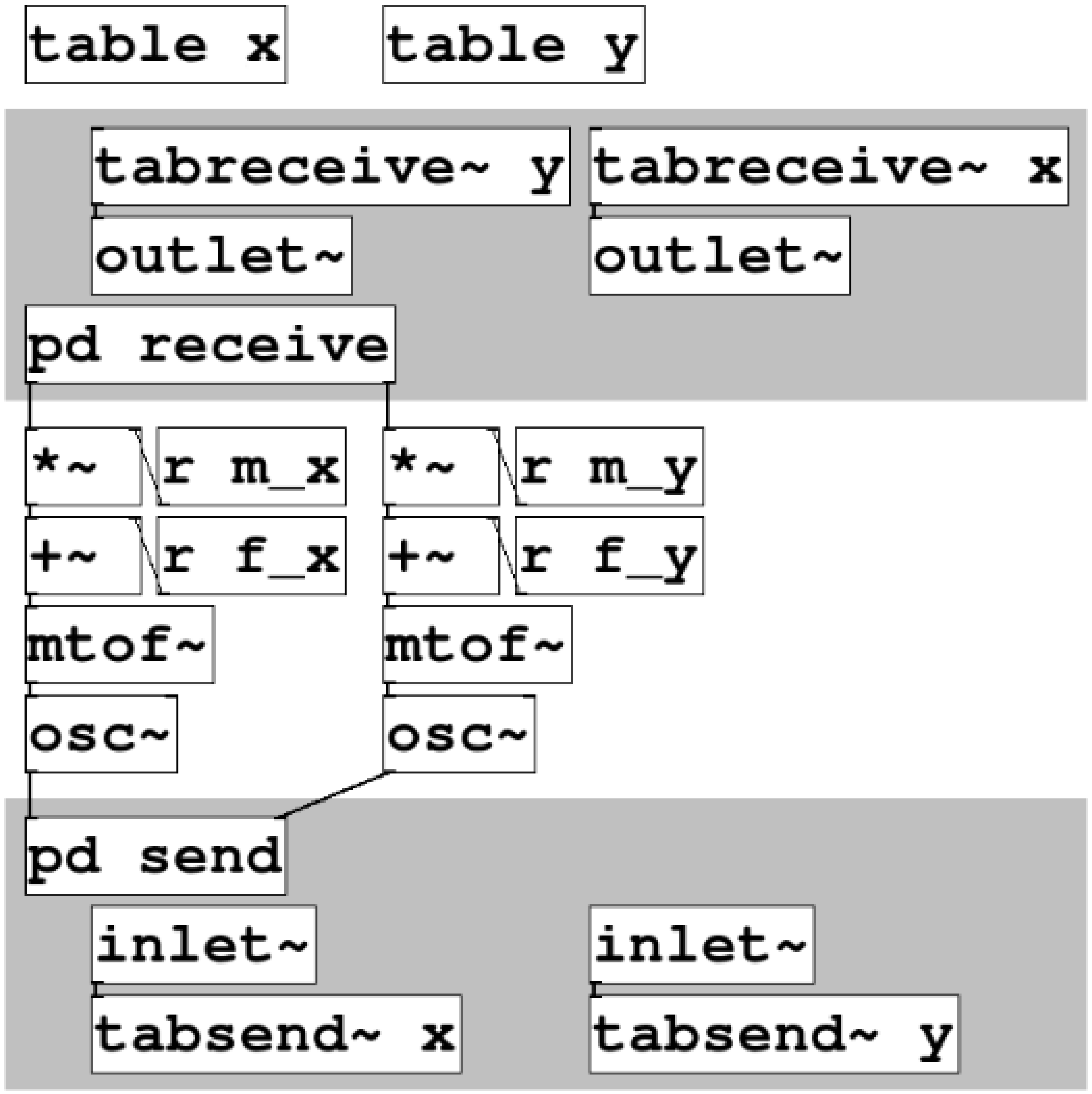
Implementation

Coupled FM Oscillators
GPU Overview
GPU Details
Lyapunov Exponents

Coupled FM Oscillators

Pure-data Implementation
Recurrence Relation
4D Parameter Space

Pure-data Implementation



Recurrence Relation

x(t+1) = F(fx, mx, x(t), y(t-d))
y(t+1) = F(fy, my, y(t), x(t-d))
F(f,m,u,v) = wrap(u + I(f + m cos(2 pi v)))
I(n) = 440/SR 2^((n - 69)/12)

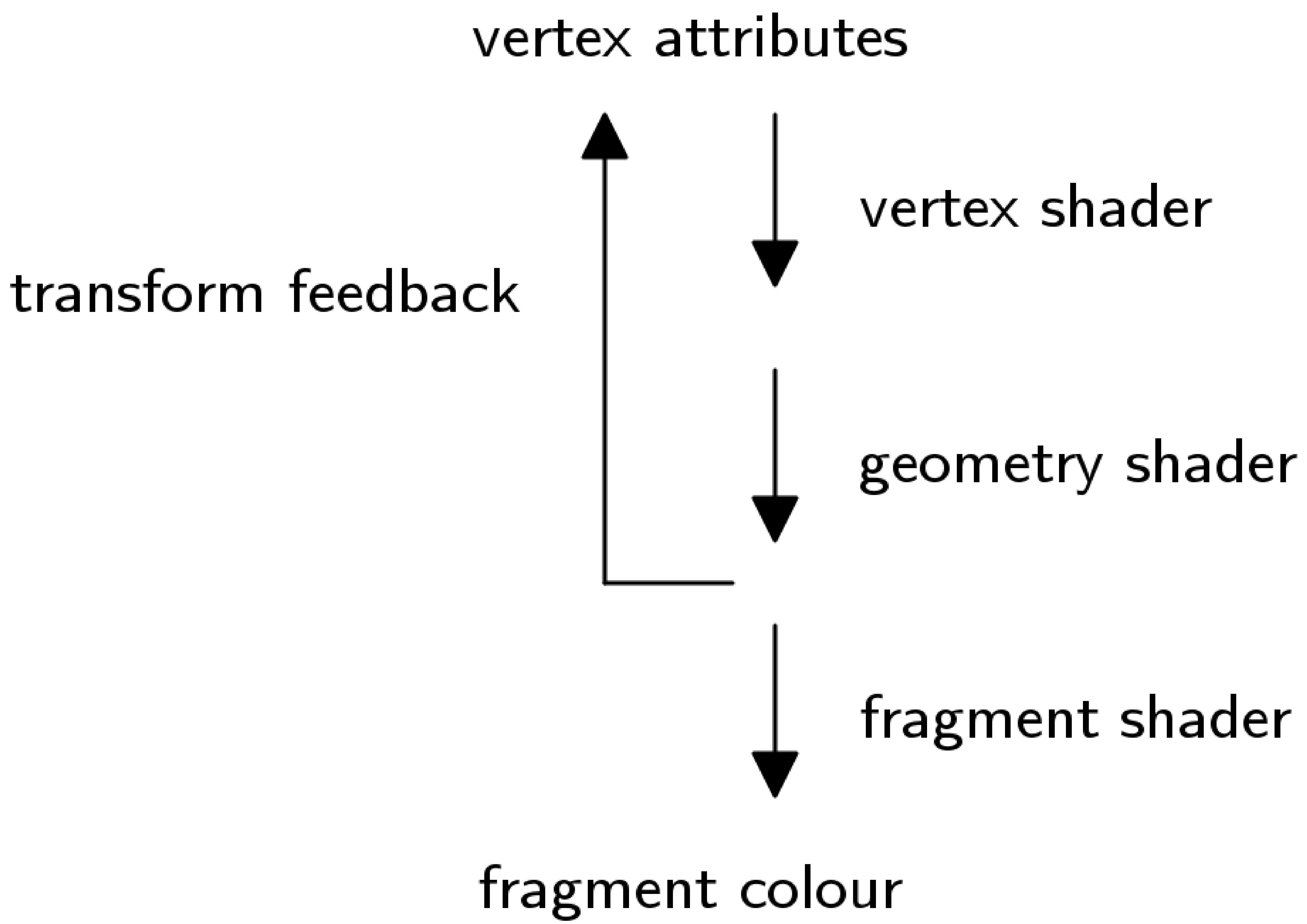
4D Parameter Space

Base frequency (oscillator X)
Base frequency (oscillator Y)
Modulation index (oscillator X)
Modulation index (oscillator Y)

GPU Overview

OpenGL Pipeline
OpenGL Glossary
Hardware Limits
Buffer Memory Layout
Vertex Attribute Pointers

OpenGL Pipeline



OpenGL Glossary

shader: stage in the pipeline
program: a linked set of shaders
uniform: parameters for a program
attribute: per-vertex shader input
varying: data passed between shaders
texture: 2D array stored in GPU memory
buffer: 1D array stored in GPU memory

Hardware Limits

16 attributes
4 components per attribute
2(d+1) components for phase space vector
5 components needed elsewhere
d <= 28 with optimal packing

Buffer Memory Layout

p_0	p_1	•••	p_N	l	q

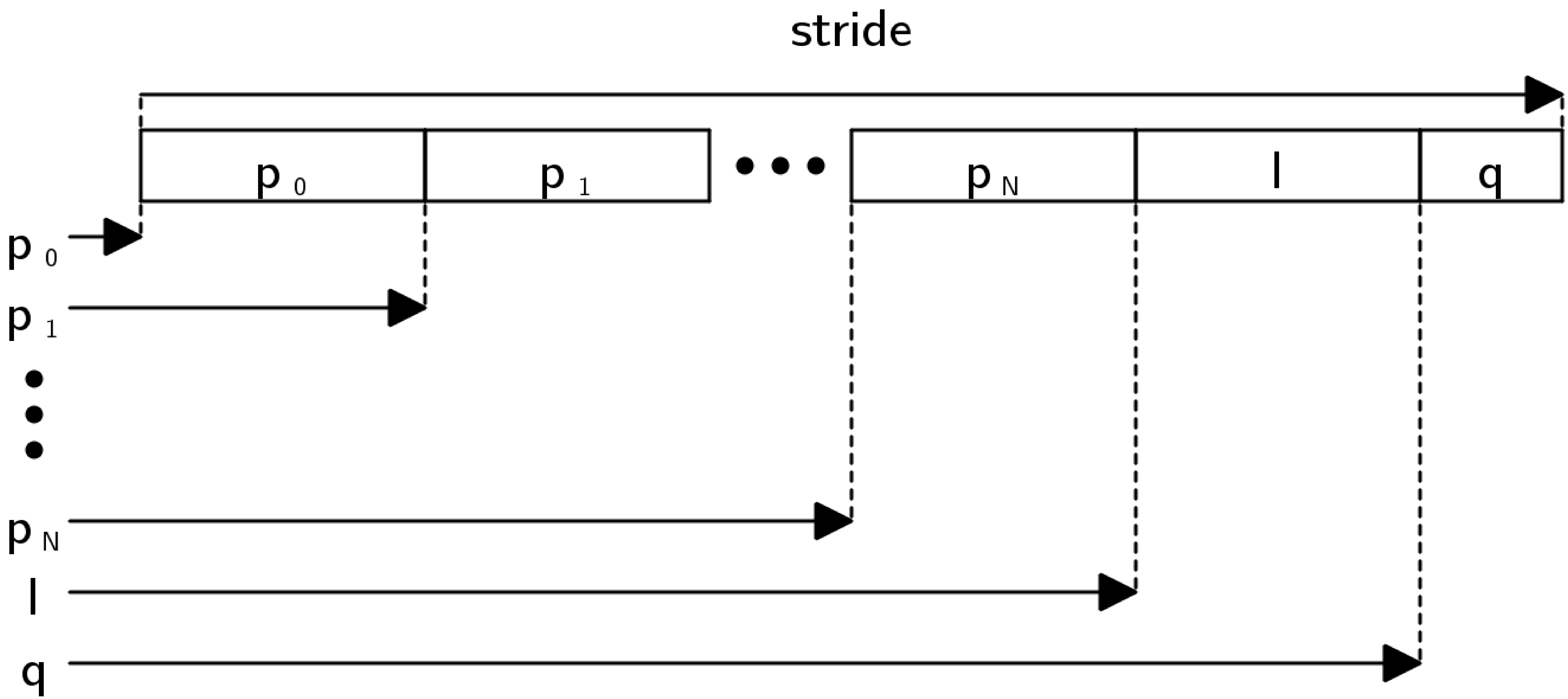
d odd

x_0	y_0	x_1	y_1	x_2	y_2	x_3	y_3	•••	x_{d-1}	y_{d-1}	x_d	y_d	1	L	LL		u	v

d even

x_0	y_0	x_1	y_1	x_2	y_2	x_3	y_3	•••	x_d	y_d			1	L	LL		u	v

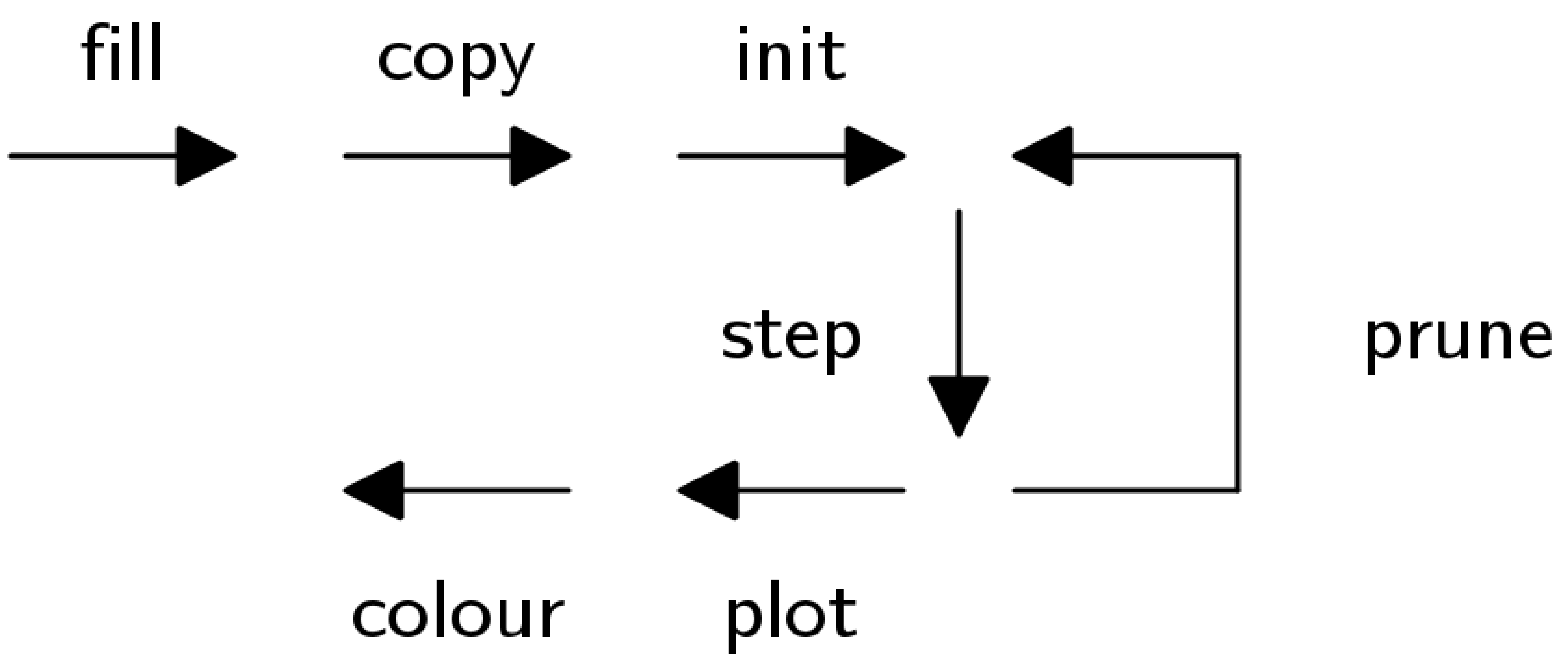
Vertex Attribute Pointers



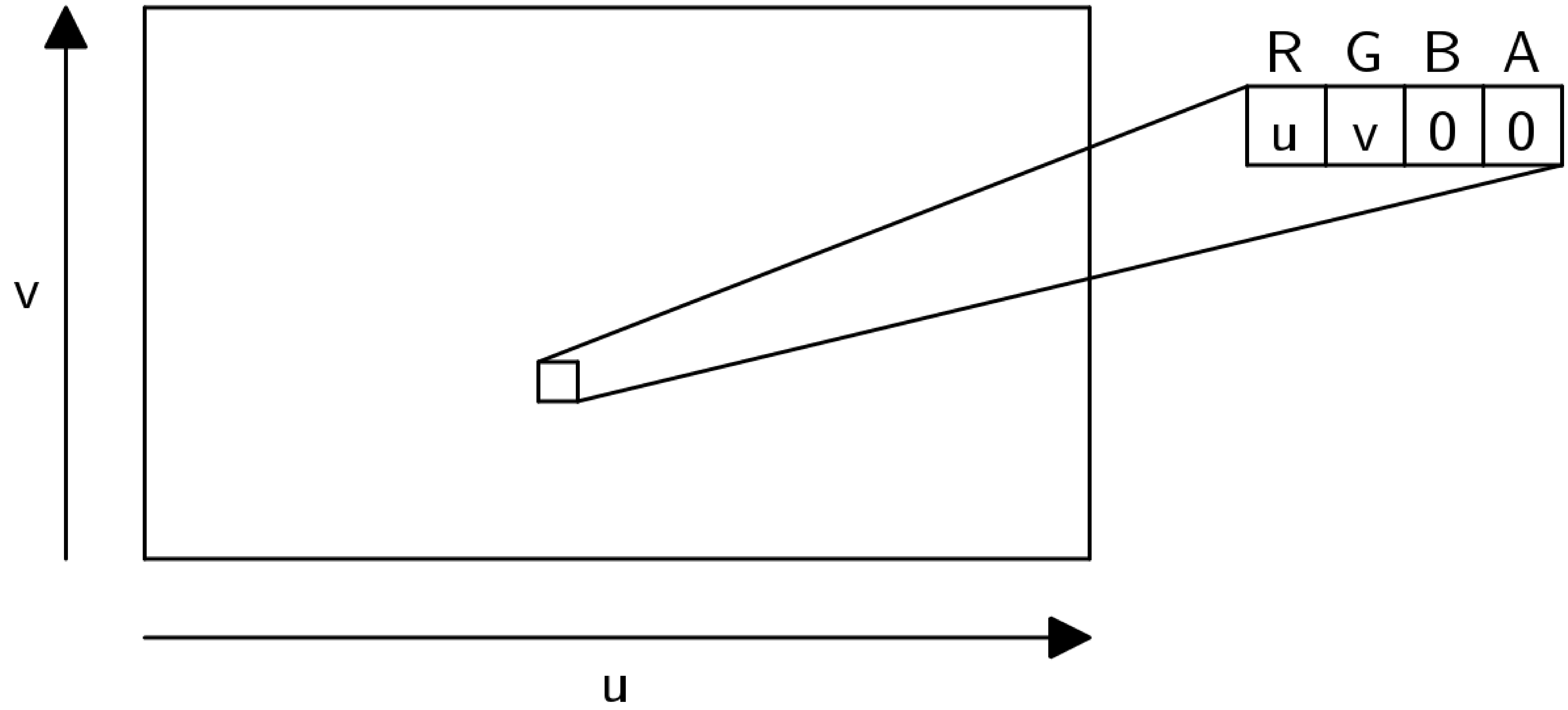
GPU Details

Computation Structure
Fill RGBA Texture with uv00
Copy RG Channels to Buffer
Initialize State
Step Recurrence
Prune Working Set
Plot Points
Colour Image

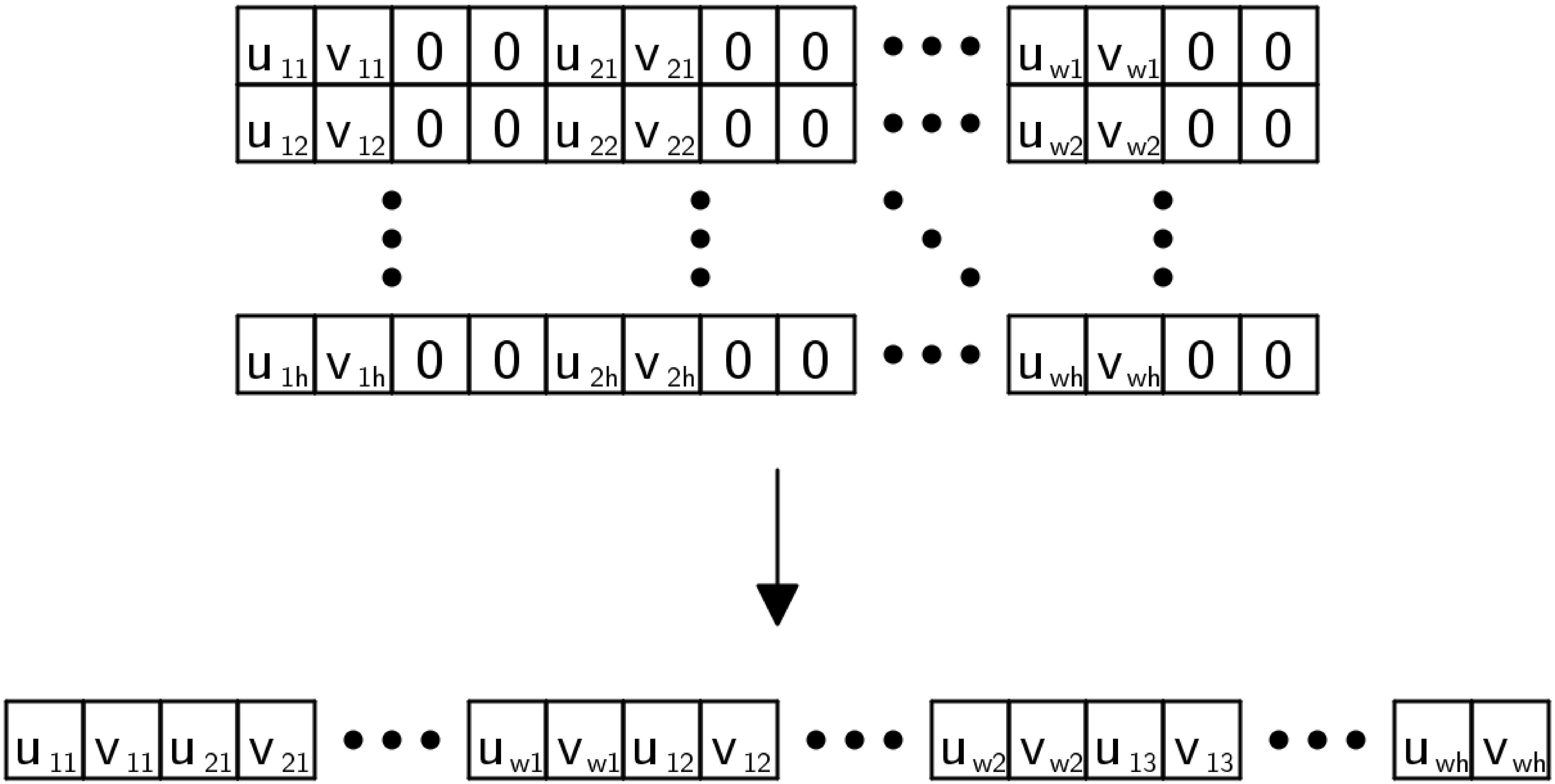
Computation Structure



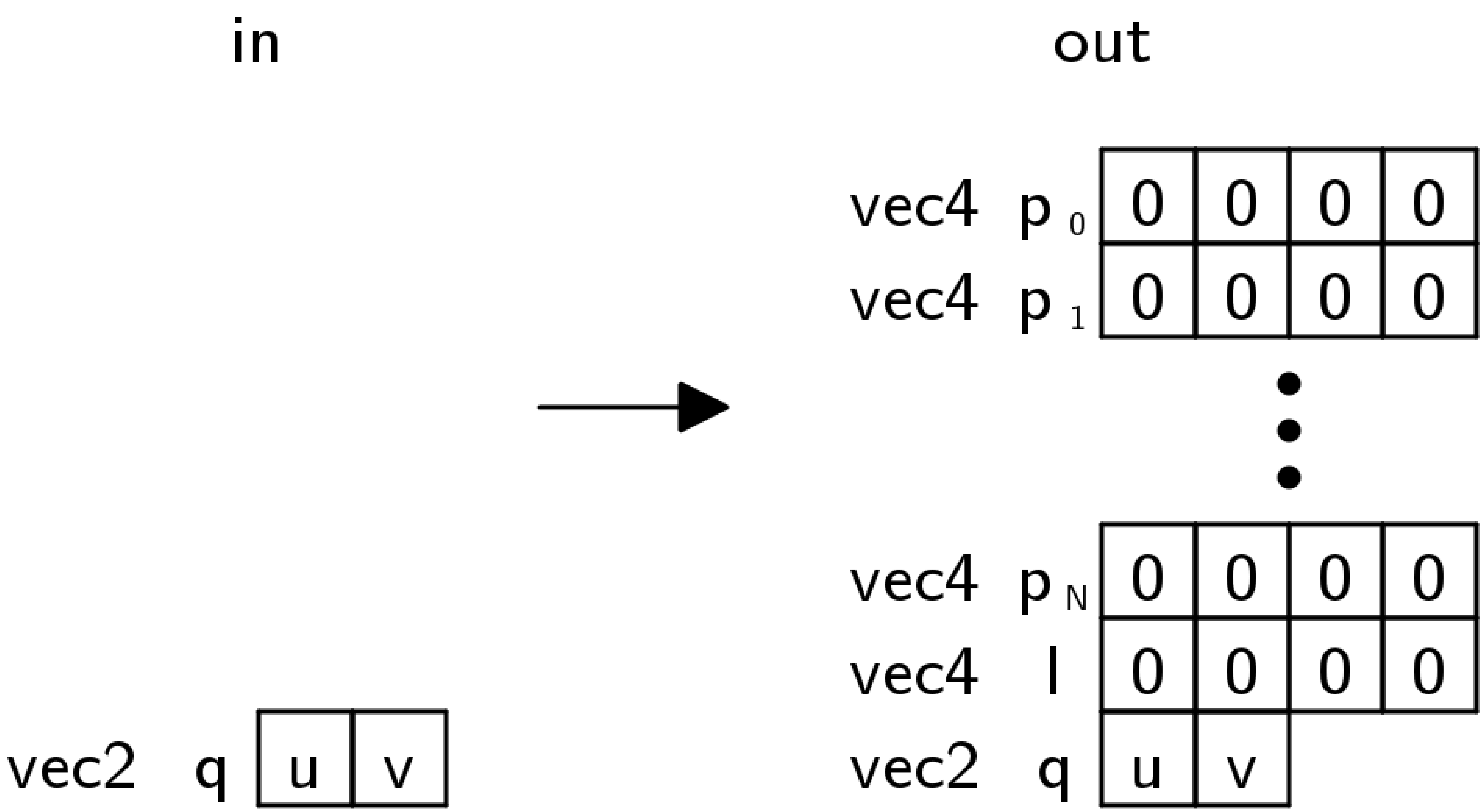
Fill RGBA Texture with uv00



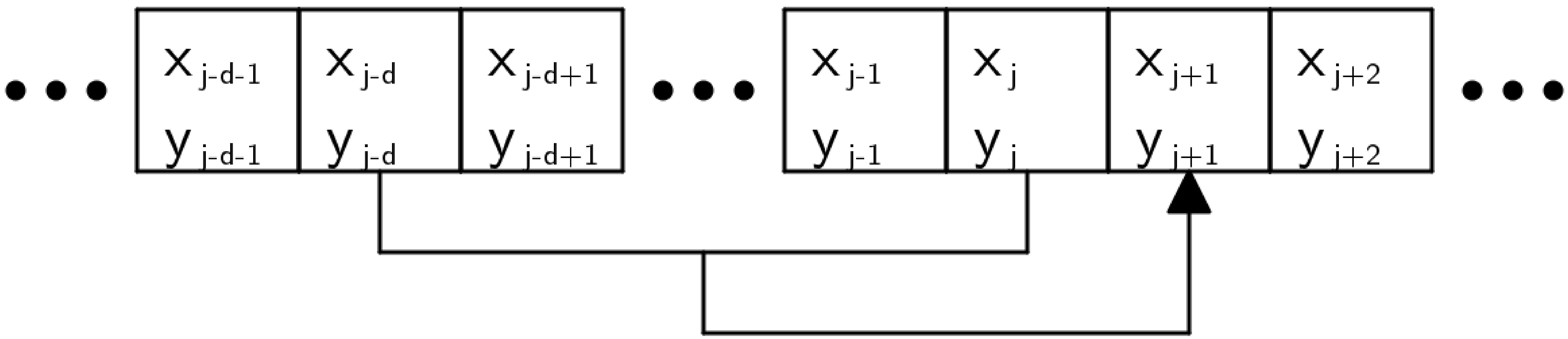
Copy RG Channels to Buffer



Initialize State



Step Recurrence



$$\begin{pmatrix} x_{j+1} \\ y_{j+1} \end{pmatrix} = \operatorname{wrap}\left(\begin{pmatrix} x_j \\ y_j \end{pmatrix} + \operatorname{inc}\left(\begin{pmatrix} f_x \\ f_y \end{pmatrix} + \begin{pmatrix} m_x \\ m_y \end{pmatrix} \cos\left(2\pi \begin{pmatrix} y_{j-d} \\ x_{j-d} \end{pmatrix} \right) \right) \right)$$

$$\operatorname{inc}(n) = \frac{440}{\mathrm{SR}} 2^{\frac{n-69}{12}}$$

Prune Working Set

Memory bandwidth limits speed
Some points converge faster
Geometry shader culls converged points
Working set becomes leaner

Plot Points

Render to framebuffer texture
gl_FragColor = raw L statistics

Colour Image

Read raw L statistics from texture
gl_FragColor = F(l)
stable $\rightarrow$ yellow
chaotic $\rightarrow$ blue

Lyapunov Exponents

Informal Definition
Formal Definition
Calculation in 1D
Calculation in >1D
Numerical Calculation

Informal Definition

$|\Delta z(t)| = \exp(L\ t)\ |\Delta z(0)|$

$L < 0$ stable; nearby values pulled closer

$L > 0$ chaotic; nearby values pushed apart

Formal Definition

$L = 1/t \log |\Delta z(t)|/|\Delta z(0)|$

Take limits as $t \to$ infinity and $\Delta z(0) \to 0$

Calculation in 1D

Inner limit become derivative
Recurrence derivative is product of step derivatives
Log of product is sum of logs

Calculation in $>$1D

Inner limit becomes Jacobian
Outer limit defines a matrix
Log of matrix eigenvalues are Lyapunov spectrum
Maximal Lyapunov exponent measures stability

Numerical Calculation

Pick a point on the attractor z(0)
Pick a nearby point w(0)
Run recurrence on both until t large
Compute $L = 1/t \log |z(t)-w(t)|/|z(0)-w(0)|$
Repeat, averaging all the results

Conclusions

Examples
Evaluation
Ends

Examples

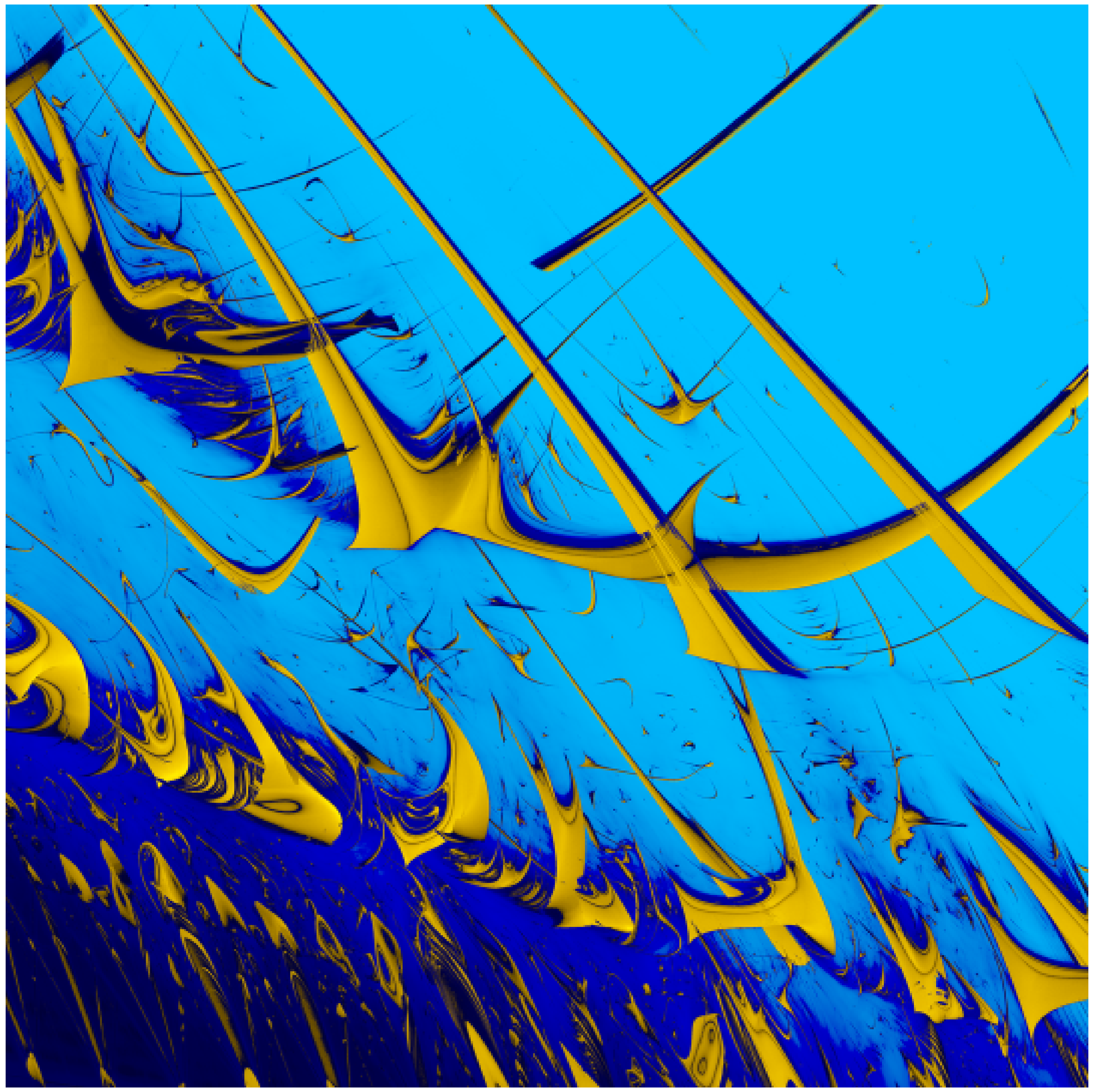
$d = 0$

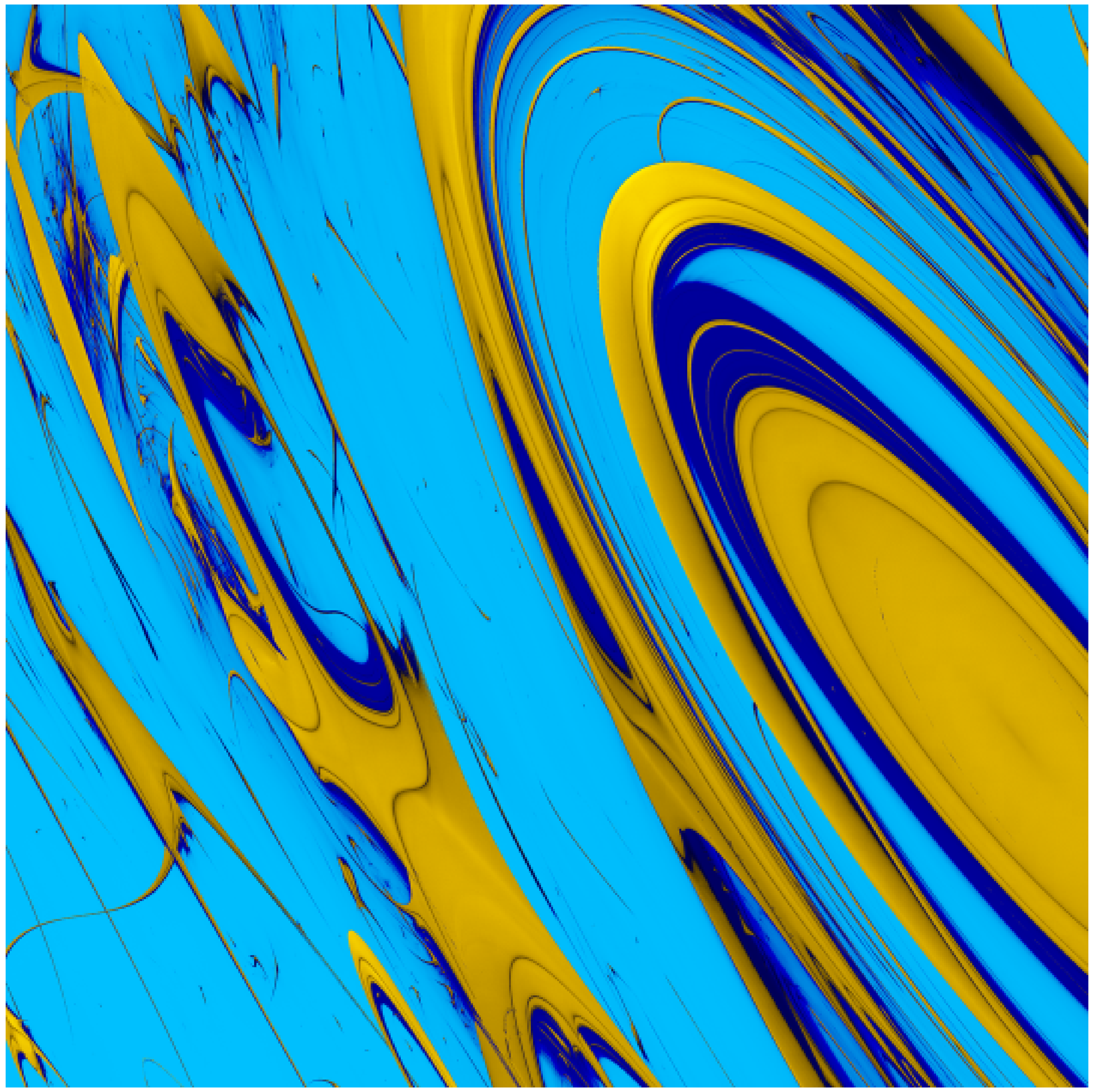
$d = 1$

$d = 4$

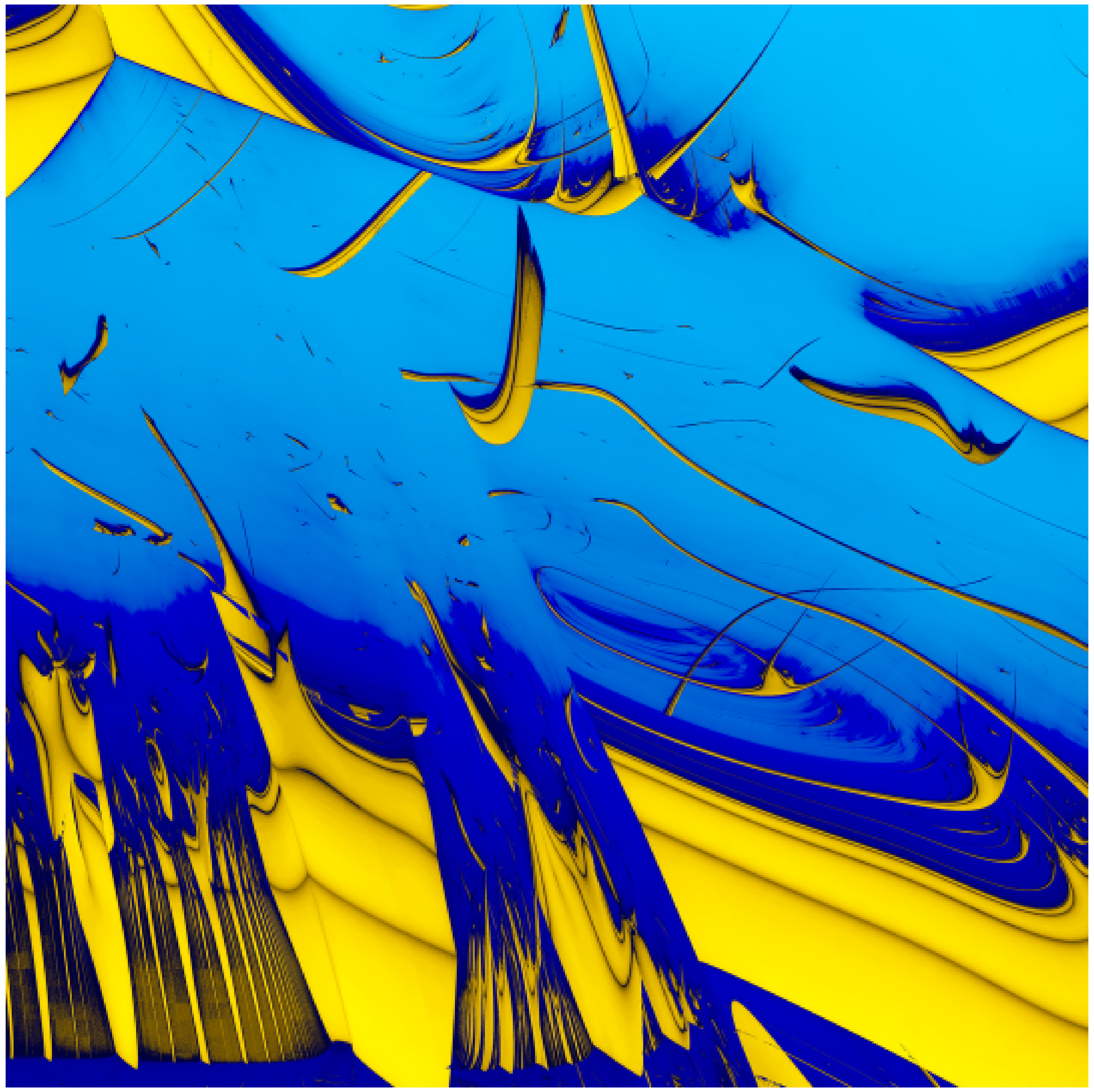
$d = 16$

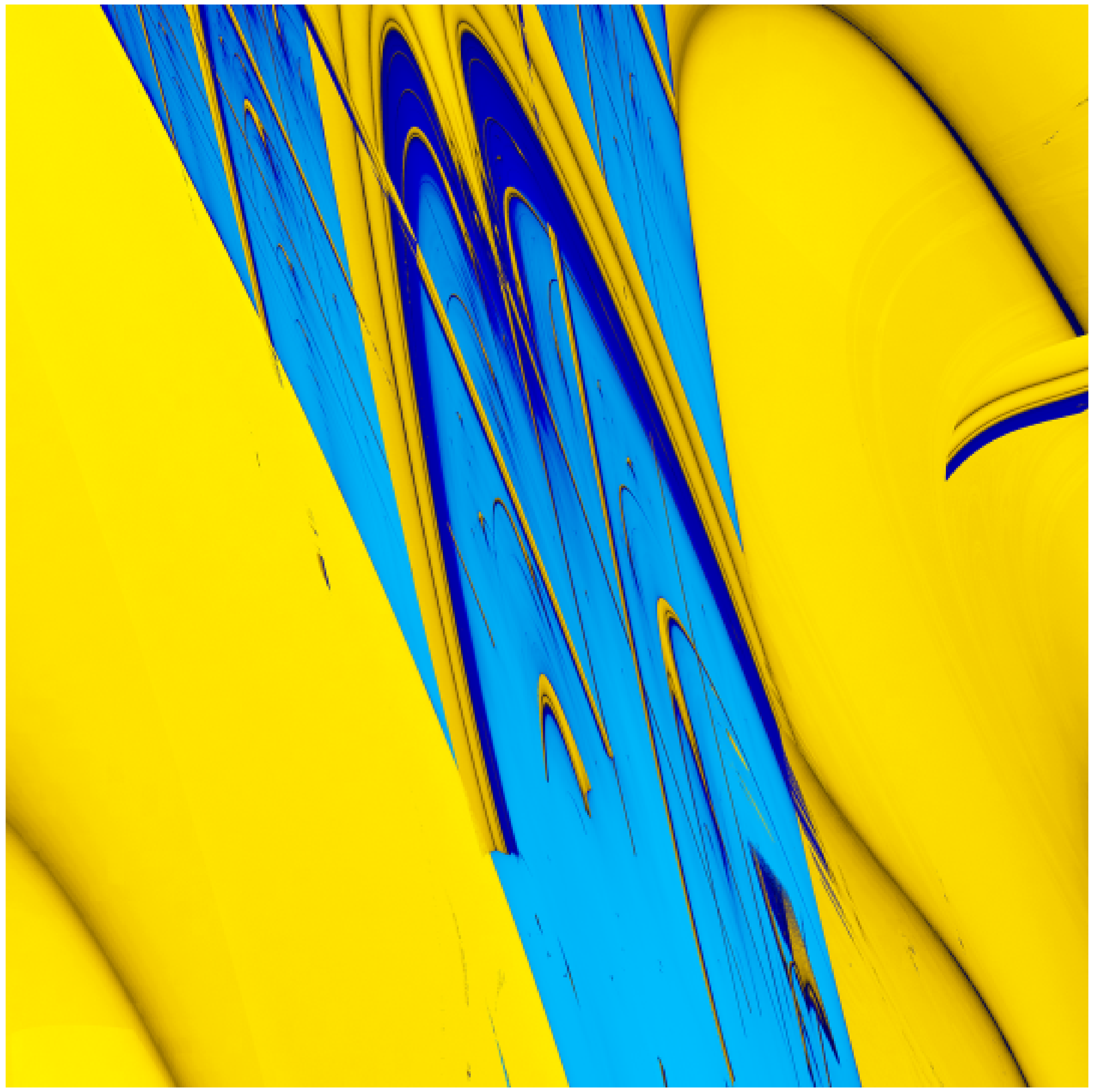
d = 0



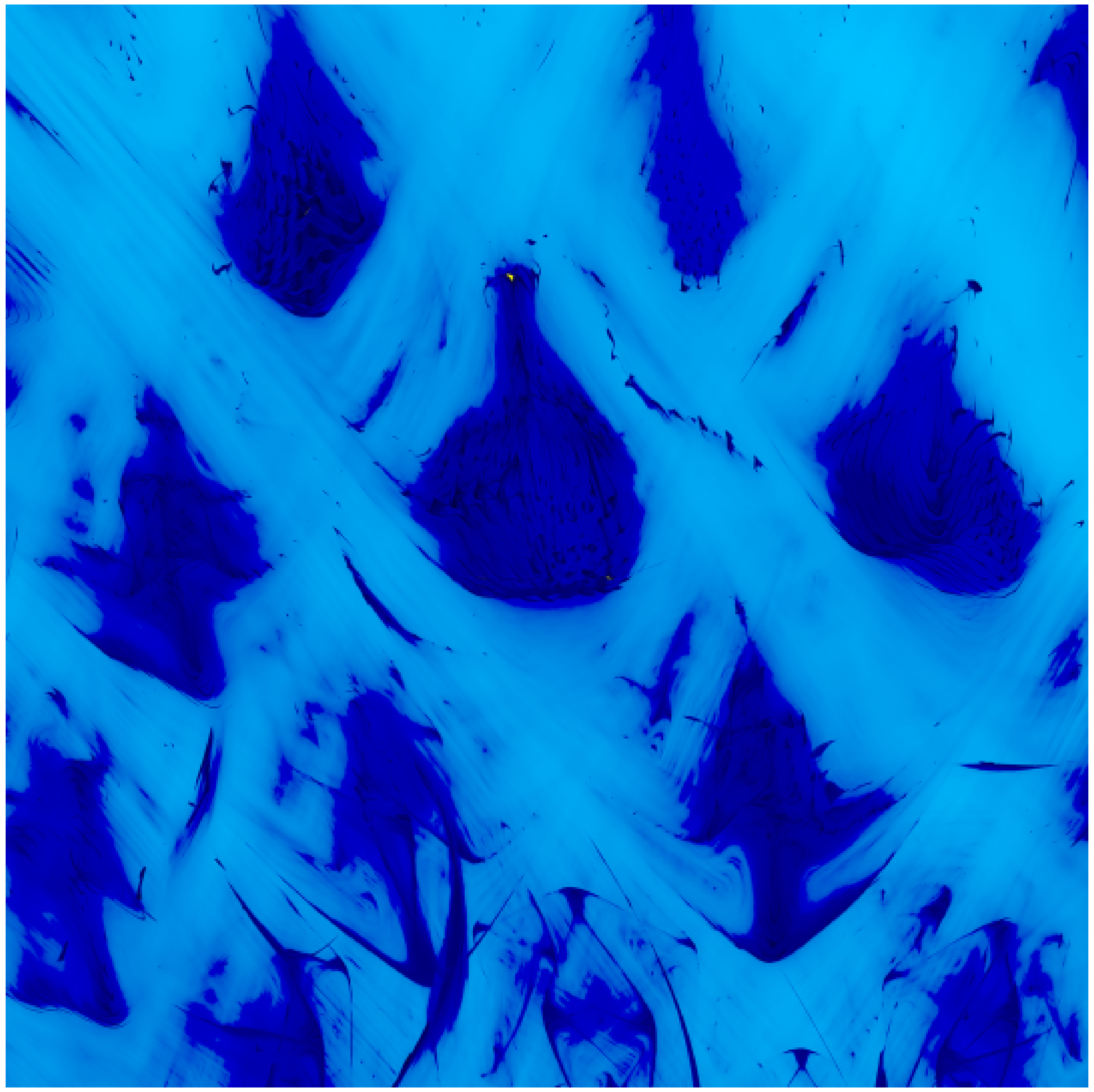
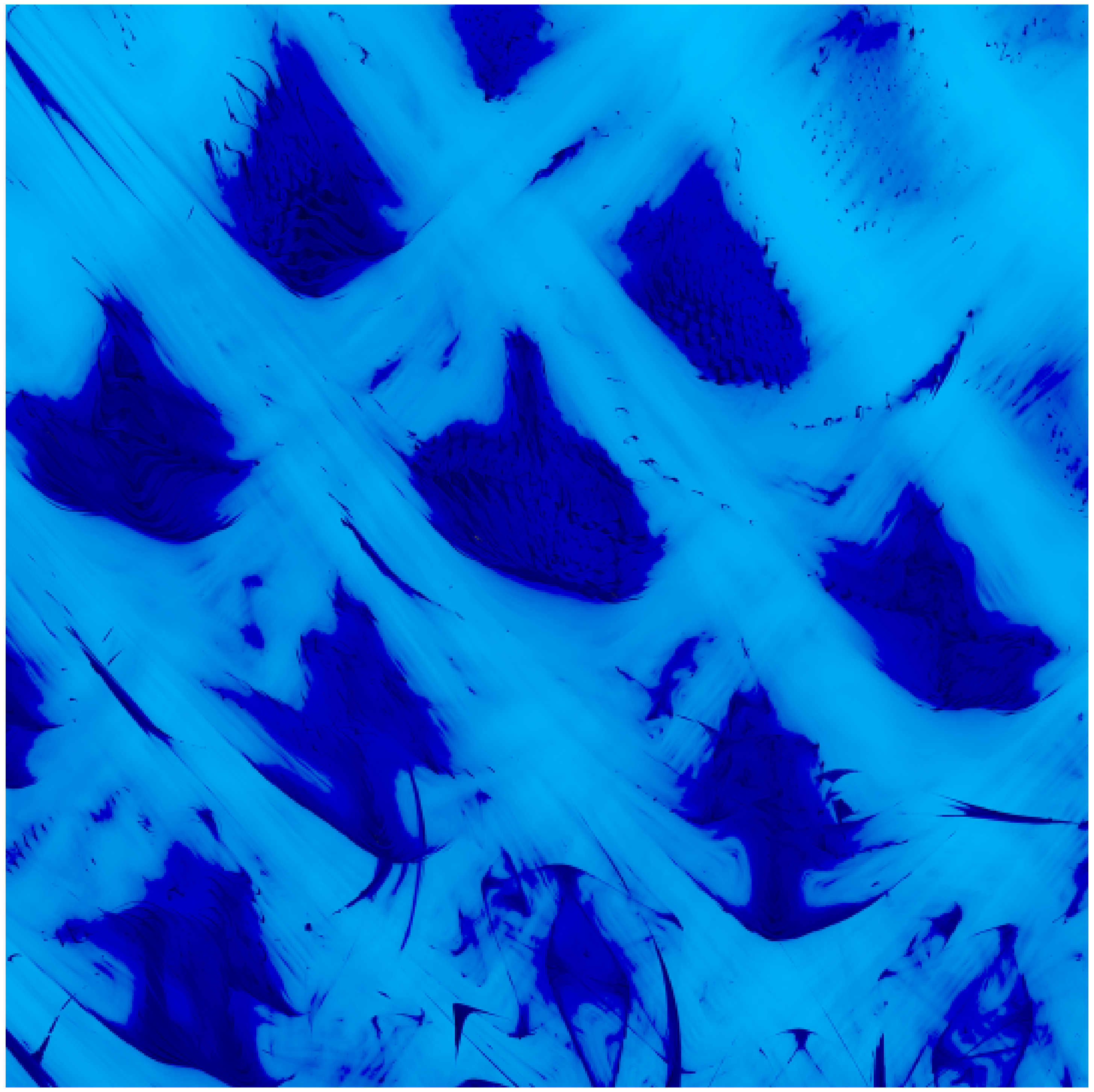


d = 1

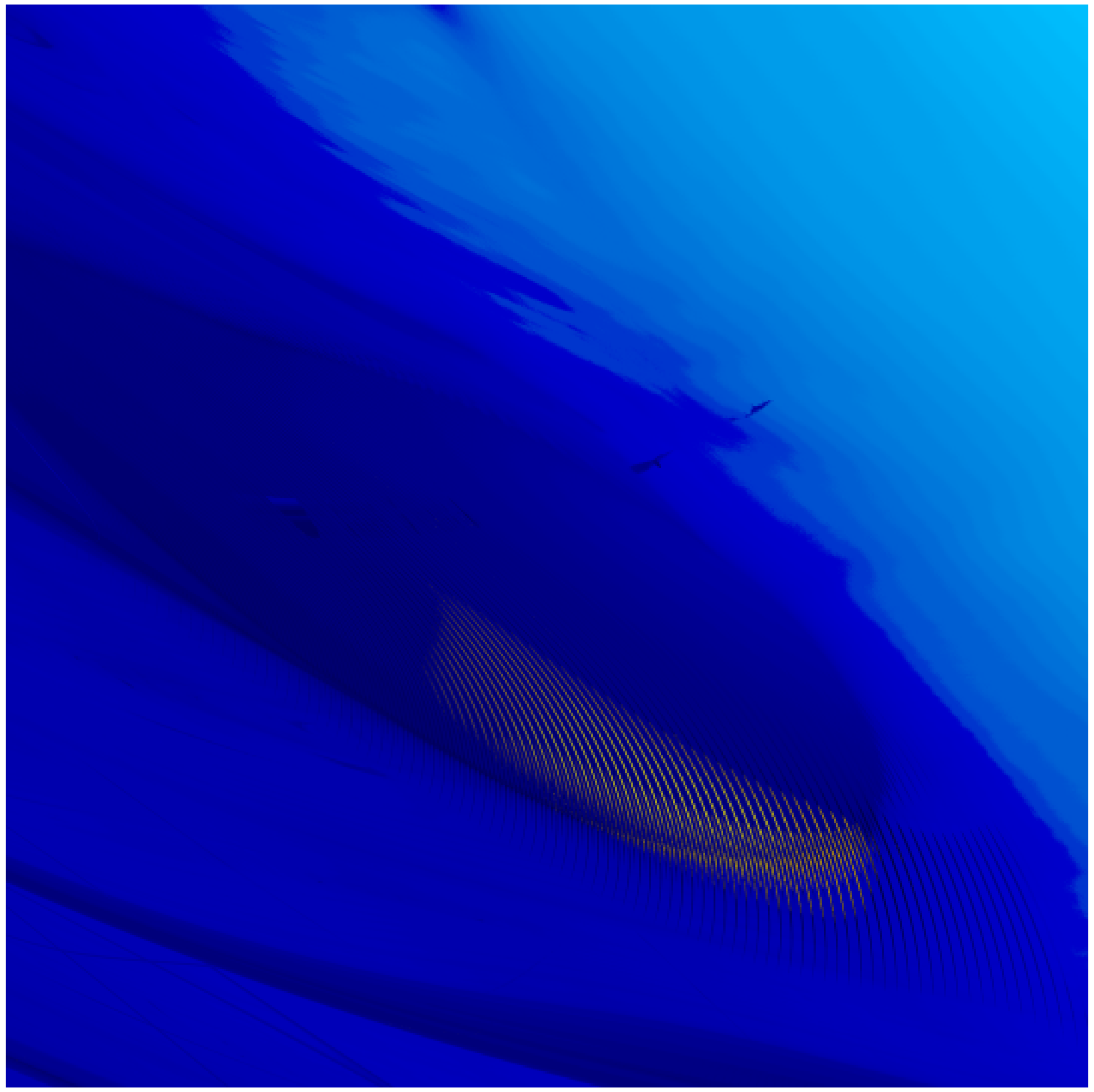


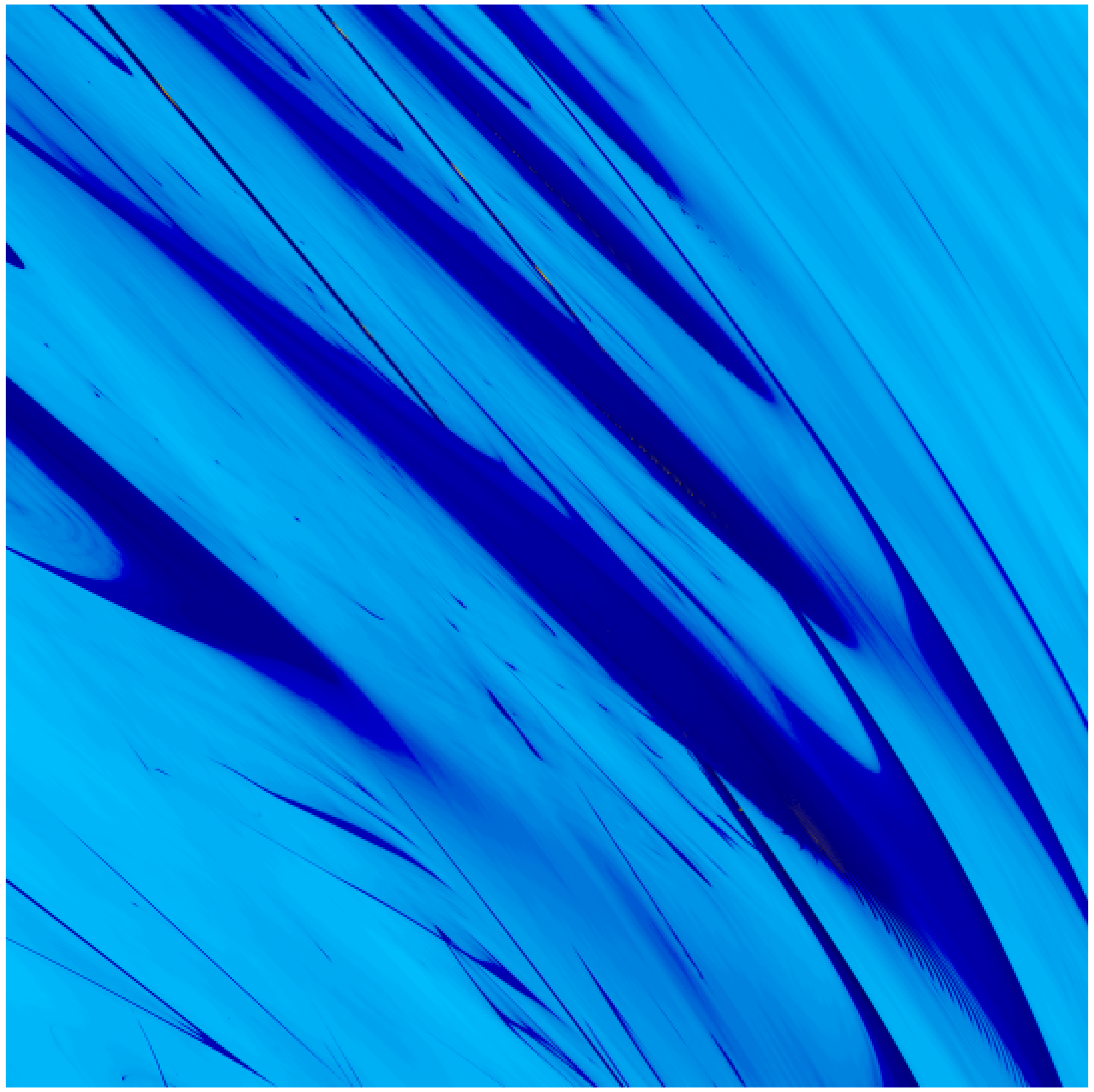


d = 4

d = 16





Evaluation

Good Points
Bad Points

Good Points

Operation at 0 <= d <= 27
Experimental fractional d (interpolation)
Batch mode for offline rendering
Tiled rendering of huge images

Bad Points

JACK xruns with heavy GPU load
Free drivers might be better than proprietary?
Perceptual (ir)relevance of Lyapunov exponent
Tiled rendering random seed glitches
Long video render times even with fast hardware

Ends

Links

Links

http://mathr.co.uk/lyapunov-fm

claude@mathr.co.uk