Spline Modeling of Audio Signals with Cycle Interpolation

Matt Klassen

DigiPen Institute of Technology, Redmond, Washington, USA

June 21, 2022

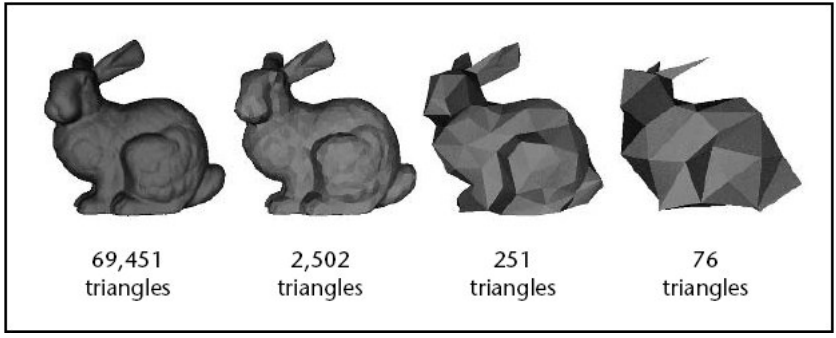
Thank you MCM 22

Thank you to the organizers of MCM 22!

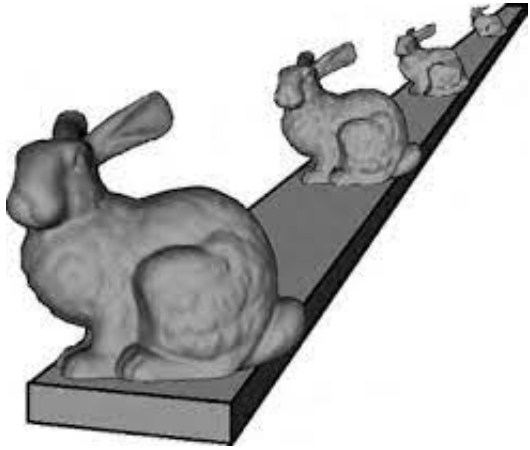
I appreciate the opportunity to participate in this conference.

My interest in this topic has grown out of years of teaching spline modeling for computer graphics, mathematics of music and sound, and digital signal processing, at DigiPen Institute of Technology in Redmond, WA.

Motivation: Level of Detail (LOD)



Motivation: Level of Detail (LOD)



Model Simplification

- Graphical models can be simplified by removing vertices, polygons, or textures
- What is the audio analogy?
- 1. Remove samples and replace with spline curve
- 2. Remove cycles and replace with interpolated cycles

Applications that could benefit

- Realtime interactive media (many audio assets plus effects)
- Music composition in a DAW (many instrument sample libraries)
- Others?

Splines and Signal Processing (Unser, 1999)

Splines

A Perfect Fit for Signal and Image Processing

Michael Unser

Finding a general mechanism for switching between the continuous and discrete signal domains is one of the fundamental issues in signal processing. It is a question that arises naturally during the acquisition process where an analog signal or image is to be converted into a sequence of numbers (discrete representation). Conversely, the need for a continuous signal representation comes up every time one wishes to implement numerically an operator that is initially defined in the continuous domain. Typical examples in image processing are the detection of edges through the computation of gradients (spatial derivatives), and geometric transformations such as rotations and scaling (interpolation).

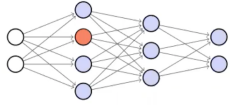
The textbook approach to those problems is provided by Shannon's sampling theory, which describes an equivalence between a band-limited function and its equidistant samples taken at a frequency that is superior or equal to the Nyquist rate [76]. Even though this theory has had an enormous impact on the field, it still has a number of problems. First, it relies on the use of ideal filters, which are devices not commonly found in nature. Second, the band-limited hypothesis is in contradiction with the idea of a finite (or finite duration) signal. Third, the bandlimiting operation tends to generate Gibbs oscillations, which can be visually disturbing in images. Finally, the underlying cardinal basis function [$\mathrm{sinc}(x)$] has a very slow decay, which makes computations in the signal do-

1053-5888/99/$10.00©1999IEEE

Splines and Signal Processing (Unser, 2021)

EPFL

Deep splines



Michael Unser
Biomedical Imaging Group
EPFL, Lausanne, Switzerland

Joint work with
Pakshal Bohra, Joaquim Campos, Harshit Gupta, Shayan Aziznejad

Keynote presentation, EUSIPCO'2020, Jan 18-22, 2021 Amsterdam, NL (virtual)

Splines and Synthesis (Collins 1999)

Realisation

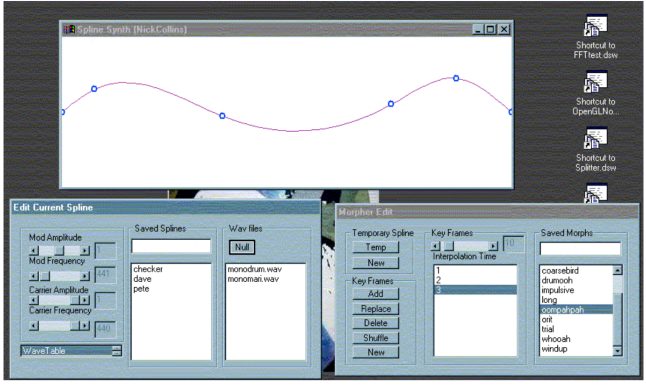


Figure 7 SplineSynth

Splines as Audio Generators

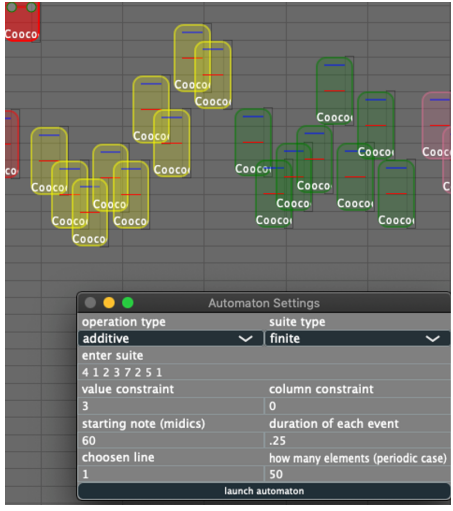
Square wave

Triangle wave

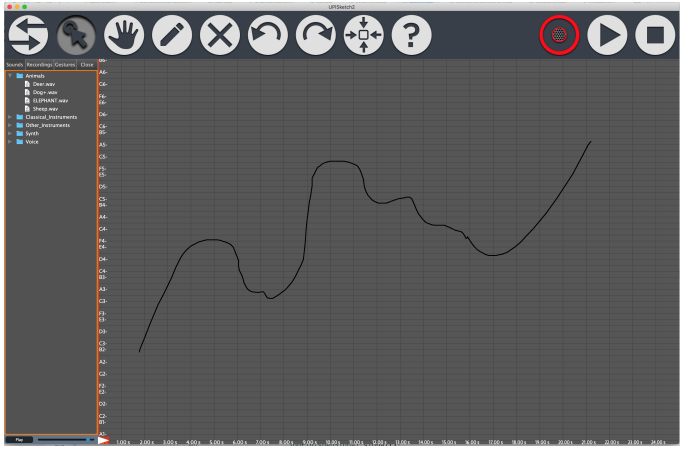
Parabolic sinusoid

Cubic sinusoid

UPISketch (Iannis Xenakis Center 2017)



UPISketch curves with splines for pitch



Properties of Spline Models of Audio

- low resolution
- time domain
- locally computable

Key-framing for Animation

- key frames are created (drawn, captured)
- in between frames are generated (interpolated, computed)
- discrete representation (wire frame, articulated skeleton)

Key-framing for Audio

- key cycles are created (designed, captured)
- in between cycles are generated (interpolated, computed)
- discrete representation (B-spline coefficients)

Three related papers (see https//azrael.digipen.edu/research)

- MCM (June 2022) *Spline modeling of audio signals and cycle interpolation*
  - introduces the Basic Model and cycle interpolation types
- SMC (June 2022) *Design of timbre with cellular automata and B-spline interpolation*
  - joint work with Paul Lanthier
- SIGMAP (July 2022) *Spline models and Level of Detail for Audio*
  - introduces the Delta Model

Spline Modeling Software Demo

- 1. Zero Crossings and Cycle Shading
- 2. B-spline Interpolation of One Cycle
- 3. The Basic Spline Model for an Audio Sample
- 4. Regular Cycle Interpolation and Singularities

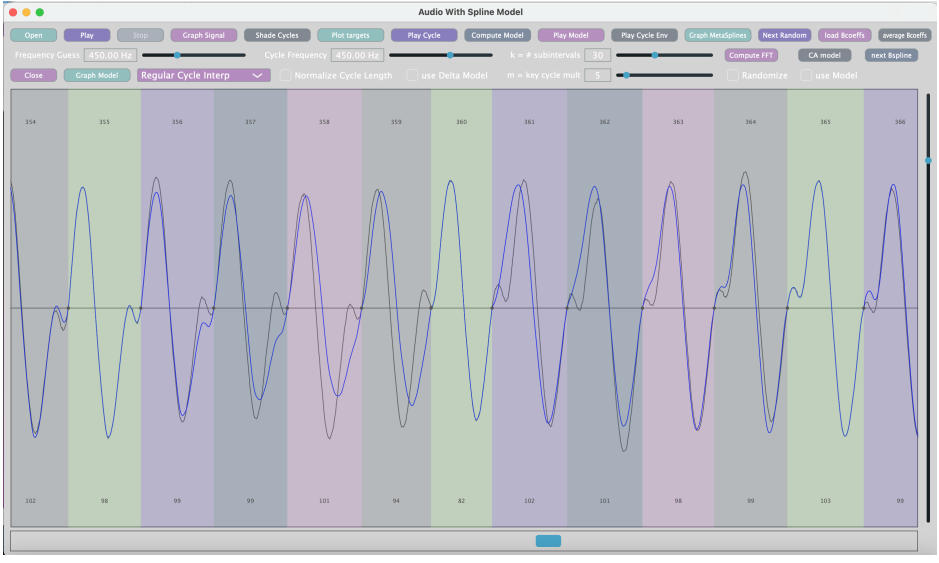
Attempts to Patch the Singularity

- add some more key cycles
  - on either side of the singularity
  - smooth the transition of cycles
- remove some key cycles
  - near the singularity
  - allows for longer interpolation sequence

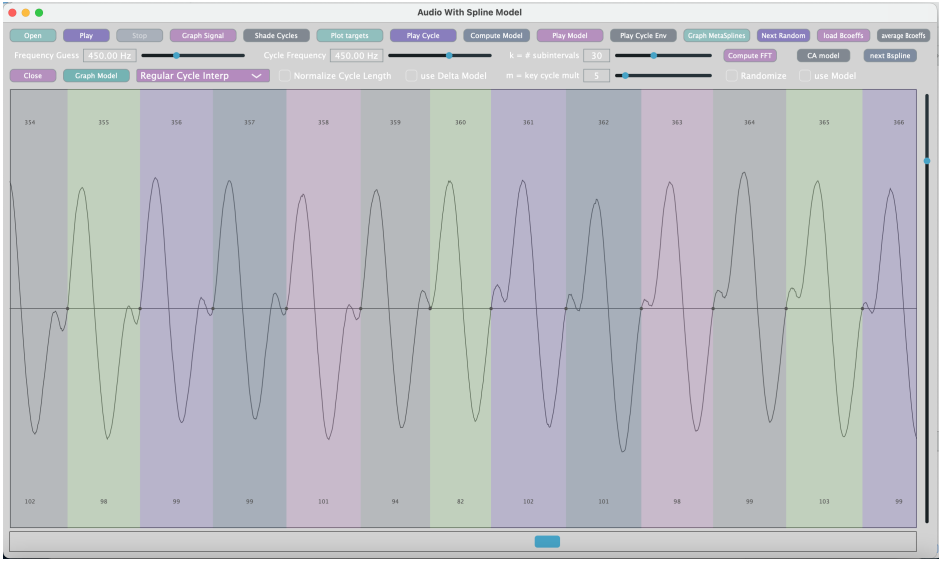
Types of Cycle Interpolation

- Regular
  - every m^{th} cycle is key
  - default for continuant type sounds
  - good for filtering, mixing
- Left-biased
  - key cycles are more dense at the beginning
  - indices can follow seqeuences such as exponential or Fibonacci
  - default for attack-decay type sounds

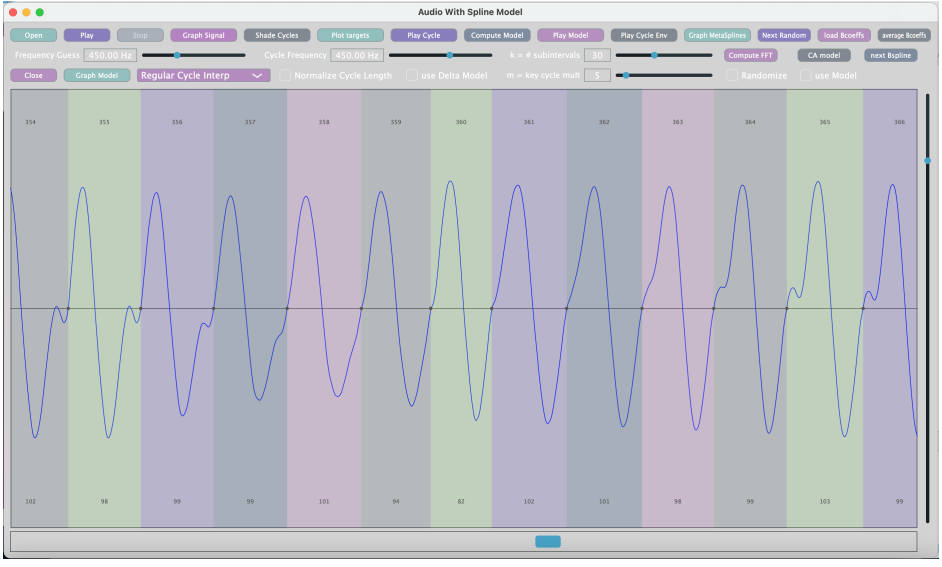
Singularity at cycle 360: signal (gray) model (blue)



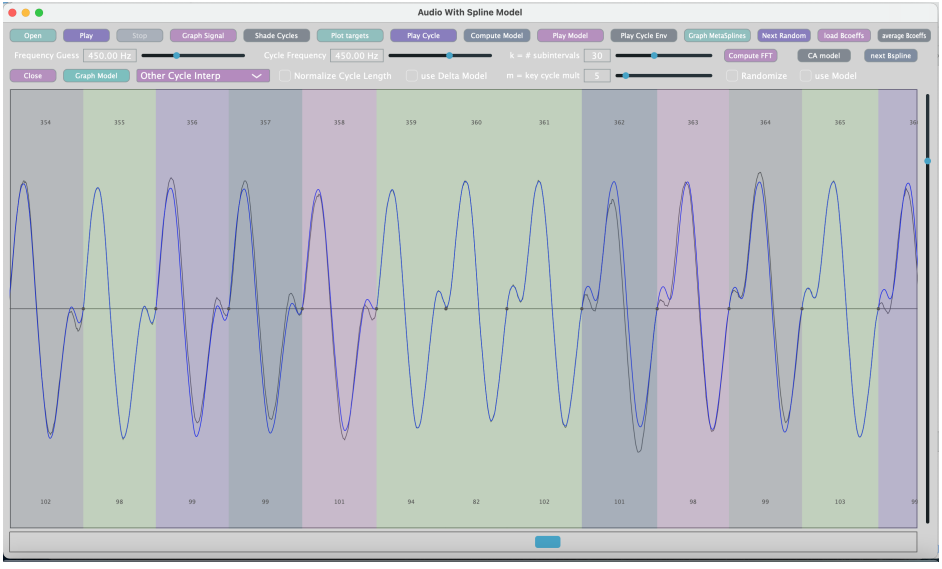
Singularity at cycle 360: signal



Singularity at cycle 360: model



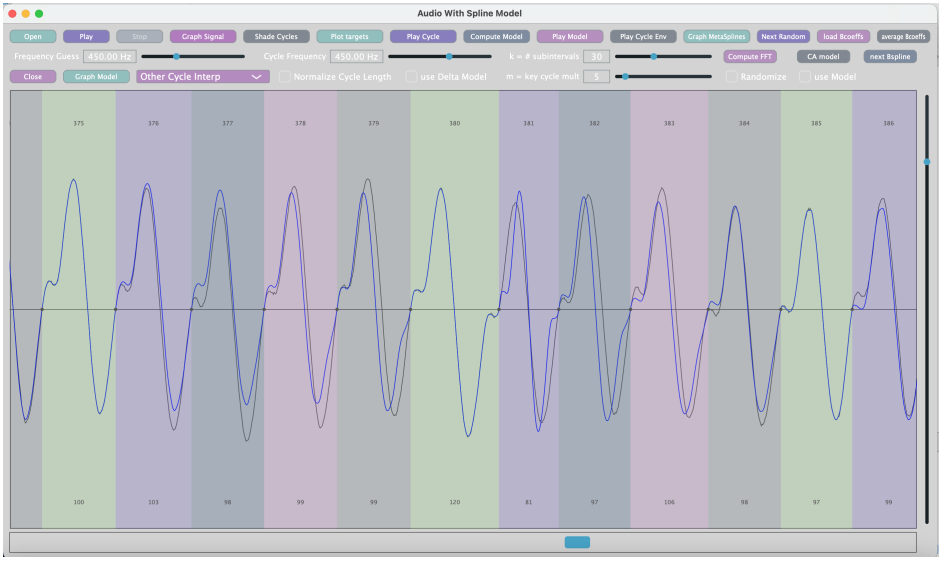
Patched Singularity at cycle 360



Patched Singularity at cycle 360: signal

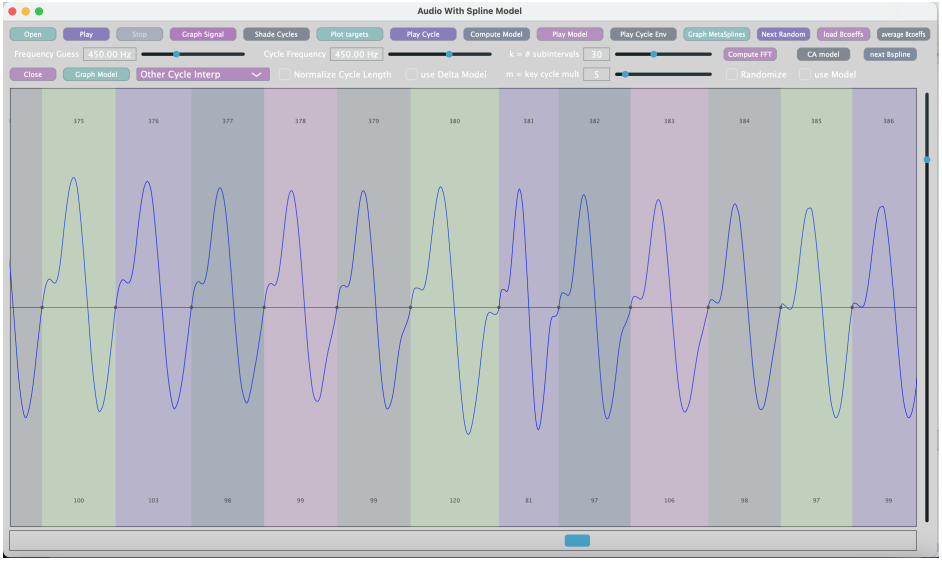
Patched Singularity at cycle 360: model

Singularity at cycle 380

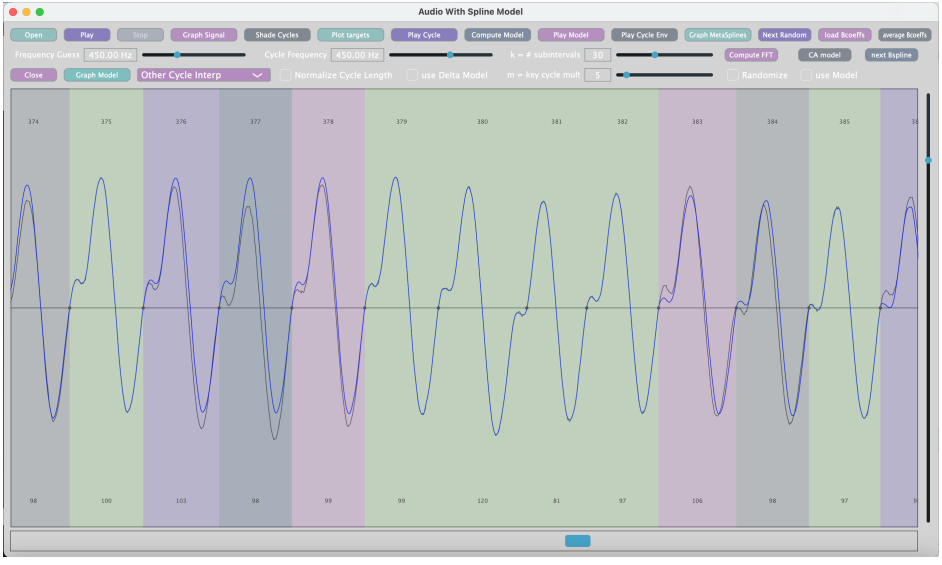


Singularity at cycle 380: signal

Singularity at cycle 380: model



Patched Singularity at cycle 380



Patched Singularity at cycle 380: signal

Patched Singularity at cycle 380: model

Defects of Cycle Interpolation

- cycles restricted to zero crossings can cause singularities
- Cycles should oscillate in a natural way
- Linear interpolation of B-spline coefficients doesn't capture this
- Missing subharmonics

All of these issues point toward the need for cycles defined without zero crossings.

Delta Model

Characteristics:

- allow cycle endpoints to be at any point on the signal graph
- express cycle as a sum of one cubic polynomial plus spline
- decouples B-spline model from endpoint values
- allows for more similarity between cycles with different heights
- goal: preserve continuity of cycle shape

Delta Model

Process:

- Begin with small number of cycles based on zero crossings
- Project endpoint b_i for cycle C_i based on frequency guess
- Project cycle C_{i-1} onto cycle C_i based on B-spline coefficients
- Compute error functional with these projected values and y_0 and y_1 on C_i
- Repeat this comparison for various b_i nearby the projected value
- Choose b_i to minimize error functional

Delta Model

The "Delta Cubic" on interval $[0, 1]$ is:

$$p(t) = y_0 + \Delta * (3t^2 - 2t^3)$$

where the endpoints are $(0, y_0)$ and $(1, y_1)$, and $\Delta = y_1 - y_0$.

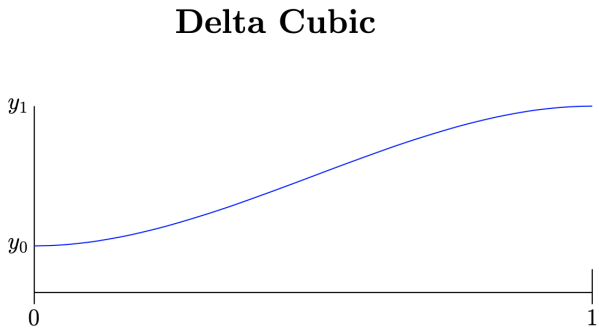
$p(t)$ can be easily converted to B-spline form using its polar form

$$F[x, y, z] = xy + xz + yz - 2xyz$$

with B-spline coefficients computed as:

$$c_i = F[t_{i+1}, t_{i+2}, t_{i+3}], i = 0, \ldots, n - 1.$$

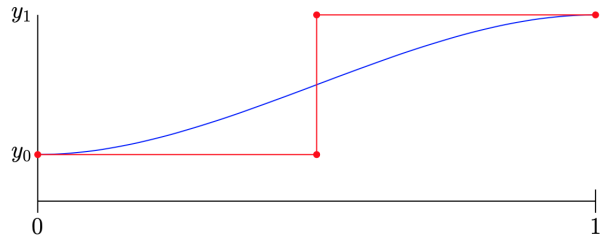
Delta Cubic



$$p(t) = y_0 + \Delta \cdot (3t^2 - 2t^3)$$

Delta Cubic

Delta Cubic Control Points



$$
\begin{aligned}
p(t) &= y_0\left[(1-t)^3 + 3(1-t)^2 t\right] \\
&+ y_1\left[3(1-t)t^2 + t^3\right]
\end{aligned}
$$

Delta Model

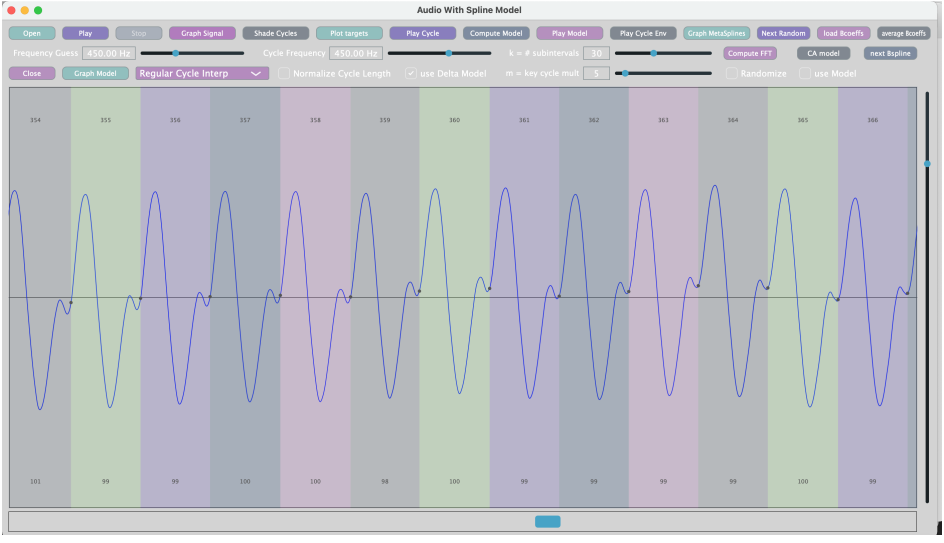
Cycle comparison

- Let S_i be the set of B-spline coefficients in the model for cycle C_i
- Let $C(S_i)$ be the B-spline curve generated by S_i
- So the cycle C_i is represented in the delta model as S_i, y_0 and y_1
- B-spline coefficients are computed after first subtracting $p(t)$ on cycle
- Error functional compares S_i to S_{i-1} (independent of y_0 and y_1)

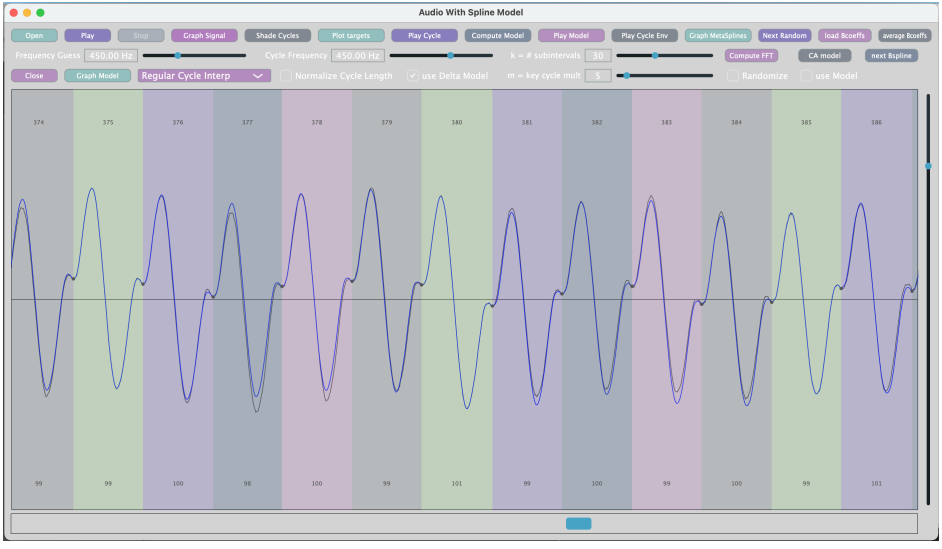
Delta Model Smooths the Singularity

Delta Model Smooths the Singularity

Delta Model Smooths the Singularity



Delta Model Smooths the Singularity



Delta Model Smooths the Singularity

Delta Model Smooths the Singularity

Comparison of Models of Guitar Pluck

Guitar pluck at 450Hz, Frequency Guess: 450, Length: 600 cycles

Comparison of Models of Guitar Pluck

Guitar pluck at 450Hz, Frequency Guess: 450, Length: 600 cycles

- Original recorded sound

Comparison of Models of Guitar Pluck

Guitar pluck at 450Hz, Frequency Guess: 450, Length: 600 cycles

- Original recorded sound
- Basic Model with $k = 30$ (about 34% of data)

Comparison of Models of Guitar Pluck

Guitar pluck at 450Hz, Frequency Guess: 450, Length: 600 cycles

- Original recorded sound
- Basic Model with $k = 30$ (about 34% of data)
- Regular Cycle Interpolation (CI) with $k = 30$, $m = 5$ (about 7% of data)

Comparison of Models of Guitar Pluck

Guitar pluck at 450Hz, Frequency Guess: 450, Length: 600 cycles

- Original recorded sound
- Basic Model with $k = 30$ (about 34% of data)
- Regular Cycle Interpolation (CI) with $k = 30$, $m = 5$ (about 7% of data)
- Add patches around cycles 360 and 380 (about 8% of data)

Comparison of Models of Guitar Pluck

Guitar pluck at 450Hz, Frequency Guess: 450, Length: 600 cycles

- Original recorded sound
- Basic Model with $k = 30$ (about 34% of data)
- Regular Cycle Interpolation (CI) with $k = 30$, $m = 5$ (about 7% of data)
- Add patches around cycles 360 and 380 (about 8% of data)
- Fibonacci CI $k = 30$, 15 key cycles (about 1.8% of data)

Comparison of Models of Guitar Pluck

Guitar pluck at 450Hz, Frequency Guess: 450, Length: 600 cycles

- Original recorded sound
- Basic Model with $k = 30$ (about 34% of data)
- Regular Cycle Interpolation (CI) with $k = 30$, $m = 5$ (about 7% of data)
- Add patches around cycles 360 and 380 (about 8% of data)
- Fibonacci CI $k = 30$, 15 key cycles (about 1.8% of data)
- Delta Model with Fibonacci CI $k = 30$, 15 key cycles (about 2.8% of data)

Comparison of Models of Guitar Pluck

Audio Demo

Small Models of Various Instruments

Model Parameters:

- Length 1 second
- Sample Rate $44,100$
- $k = 30$ subintervals per cycle
- $n = 33$ B-spline coefficients per cycle
- cycle lengths normalized
- $y_0 = y_1 = 0$
- about 1.3% of data

Timbre blending examples

- modeled 33 instruments
- each based on one recorded sample
- extracted 18 key cycles from a delta model
- $0, 5, 10, 15, 20, 25, 30, 40, 50, 60, 70, 80, 100, 120, 150, 180, 220$
- blending is based on key cycles only

Timbre blending examples

Timbre Blending Demo

Timbre blending examples

Timbre Blending Demo

- French horn evolving into Flute

Timbre blending examples

Timbre Blending Demo

- French horn evolving into Flute
- Guitar evolving into Flute

Timbre blending examples

Timbre Blending Demo

- French horn evolving into Flute
- Guitar evolving into Flute
- Single B-spline evolving into Guitar

Truncated power function

Truncated Power Function:

$$(t-c)_+^k = \begin{cases} 0, & t < c \\ (t-c)^k, & t \geq c \end{cases}$$

Important Fact: If two cubic polynomials $p_1(t)$ and $p_2(t)$ match at $t = c$ so that:

$$p_1(c) = p_2(c) \quad p_1'(c) = p_2'(c) \quad \text{and} \quad p_1''(c) = p_2''(c)$$

then $p_2(t) - p_1(t) = a \cdot (t-c)^3$ for some constant a.

Spline bases from truncated powers

This leads leads to the straight forward construction of spline bases.

For example, to represent any sequence of 3 cubic polynomials:

$$p_1(t), \quad p_2(t), \quad \text{and} \quad p_3(t)$$

on the sequence of intervals $[0, 1, 2, 3]$, with the requirement that:

the resulting piecewise function is C^2, we can write:

$$f(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 (t-1)_+^3 + a_5 (t-2)_+^3$$

Spline bases from truncated powers

Equivalently, we can write:

$$f(t) = a_0(t-0)_+^0 + a_1(t-0)_+^1 + a_2(t-0)_+^2 + a_3(t-0)_+^3 + a_4(t-1)_+^3 + a_5(t-2)_+^3$$

which gives rise to the basis:

$$\{(t-0)_+^0, (t-0)_+^1, (t-0)_+^2, (t-0)_+^3, (t-1)_+^3, (t-2)_+^3\}$$

and the "short hand" notation (knot sequence):

$$\{0, 0, 0, 0, 1, 2\}$$

Conversion to B-splines

Each cubic B-spline is a linear combination of 5 consecutive truncated powers, corresponding to 5 consecutive knot values. For example:

$$\mathcal{B}_0^3(t) = b_0(t-0)_+^0 + b_1(t-0)_+^1 + b_2(t-0)_+^2 + b_3(t-0)_+^3 + b_4(t-1)_+^3$$

To represent any $f(t)$ as above, we extend the knot sequence to:

$$\{0, 0, 0, 0, 1, 2, 3, 3, 3, 3\}$$

B-spline formulas

knot sequence:

$$\mathbf{t} = \{t_0, \ldots, t_N\} = \{0, 0, 0, 0, \frac{1}{k}, \frac{2}{k}, \ldots, \frac{k-1}{k}, 1, 1, 1, 1\}.$$

B-splines associated to $\mathbf{t}$:

$$\mathcal{B}_i^d(t) = (-1)^{d+1}(t_{i+d+1} - t_i)[t_i, t_{i+1}, \ldots, t_{i+d+1}](t - x)_+^d$$

where the bracket operator is a divided difference with dummy variable x and t constant.

B-spline recursion

(de Boor-Cox) recursion formula:

$$\mathcal{B}_i^d(t) = \frac{t - t_i}{t_{i+d} - t_i}\mathcal{B}_i^{d-1}(t) + \frac{t_{i+d+1} - t}{t_{i+d+1} - t_{i+1}}\mathcal{B}_{i+1}^{d-1}(t)$$

base case:

$$\mathcal{B}_i^0(t) = \begin{cases} 1, & t_i \leq t < t_{i+1} \\ 0, & \text{elsewhere} \end{cases}$$

B-spline evaluation with de Boor Algorithm

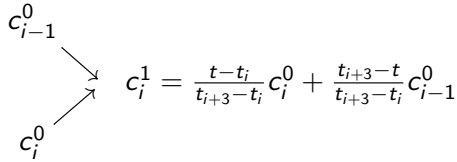
To compute $f(t) = \sum_{i=0}^{n-1} c_i \mathcal{B}_i^d(t)$

- Initialize: $c_i^0 = c_i$ for $i = 0, \ldots, N - d - 1$.
- For $t \in [t_d, t_{N-d})$ set J to be the index so that $t \in [t_J, t_{J+1})$.
- Recursion: For $p = 1, \ldots, d$, for $i = J - d + p, \ldots, J$: set

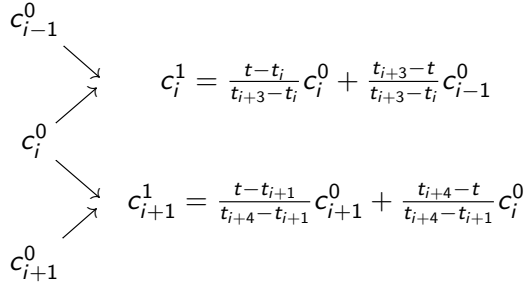
$$c_i^p = \frac{t - t_i}{t_{i+d-(p-1)} - t_i} c_i^{p-1} + \frac{t_{i+d-(p-1)} - t}{t_{i+d-(p-1)} - t_i} c_{i-1}^{p-1}$$

- $f(t) = c_J^d$.

de Boor Algorithm Diagram



de Boor Algorithm Diagram



de Boor Algorithm Diagram