

‘Continuous-Time’ Synthesis and the Context of Timelab’

March 15, 2014

1 Introduction

1.1 why timelab?

1.1.1 issues with existing systems

Music-N/CSound (Max Mathews and John ffitch, 1957-present) A classic programming environment, stemming from the pioneering work of Max Mathews, it is widely used, and highly developed.

- rigid structure
- difficult to interact with / compile time
- obtuse representation of scores and synths / hard to read: score instrument
- score/instrument layout insists on antique perception of music composition/creation – a notion which in many ways goes against the grain of analog electronic music practices

Max/Msp and Pd (David Zicarelli and Miller Puckette, 1987-present)

An almost ubiquitous and highly interactive paradigm, these programs the are industry standards (commercial and open source respectively) for experimental audio algorithm design.

- patch chords and guis!!!
- difficulty of iterating loops
- difficulty in doing things in mass
- flow diagram structure (borrowing from analog synthesizers) also insists on a way of thinking about music composition – denying the user a ‘code-ish’ mindset

SuperCollider (James McCartney, 1996-present) SC provides the inter-activeness of Pd with the ‘codiness’ of a text based programming environment such as CSound.

- server/client structure has benefits but designates a .2s default latency between server time and client time
- non-standard editor and gui environments

ChuckK (Ge Wang and Perry Cook) ChuckK is another text-based audio programming environment that emphasizes a live-coding ethos.

- admittedly emphasizes ease of programming over performance (ChuckK is slow)
- unintuitive approach to time by treating ‘now’ as a variable that we manipulate by hand: <http://en.wikipedia.org/wiki/ChuckK>

1.1.2 Fall 2012 206

1.1.3 Control timing

All of the above mentioned systems have specific solutions to problems of rectifying the timing between control input and precisely when control is enacted in the DSP engine. Cf. my poster from last year’s ICMC (TIMELAB: YET, YET ANOTHER REAL-TIME AUDIO PROGRAMMING SYSTEM – not yet available online) for a quick and dirty on this subject.

Also of relevance are these talks by Miller Puckette and James McCartney: <http://repmus.ircam.fr/mutant/rtmseminars>

- Puckette talk: 9:30 - 15:00 : DAGs and mutual exclusion, determinism
- McCartney talk: 29:00 - 37:45 : control rate issues

1.1.4 embedded audio programming

Just the other day (Friday, March 14) someone posted to the Pd-list asking if it was possible to crunch a Pd patch down into a guitar pedal (the answer was ‘no, not really’). Clearly, there is a market for throwing these kinds of experimental algorithms into a guitar pedal packages. One solution is to utilize micro computers such as the Raspberry Pi or UDOO. Another is to have an audio programming API in C with a build environment that can target truly embedable hardware, such as ST’s discovery board featuring the ARM Cortex processor (or any of the open source solutions built around these chips).

2 Continuous-Time Synthesis

What is ‘continuous-time synthesis’

- as distinct from traditional computer music DSP structure which consists of UGens and signal flow
- uses differential equations to model synthesis behaviors
- create complex synthesis algorithms by arranging equations in a network with interleaved numerical solver stages so that time does not pass between nodes in the network

This affords us advantages that are beyond the scope of Ugen/signal flow structures.

2.1 0-time delay lines

The object of allowing for 0-time delay lines in a network is to facilitate (among other things) accurate sync-ing. In the parlance of analog synthesizers, a master oscillator may slave another one to it by resetting the phase of the slave at every other zero crossing (period) of the master.

2.2 virtual analog

2.2.1 ngspice

2.2.2 wave-digital filters

2.2.3 moog filter example

2.3 new models for new sounds

2.3.1 noisy oscillators

2.3.2 non-linear filters

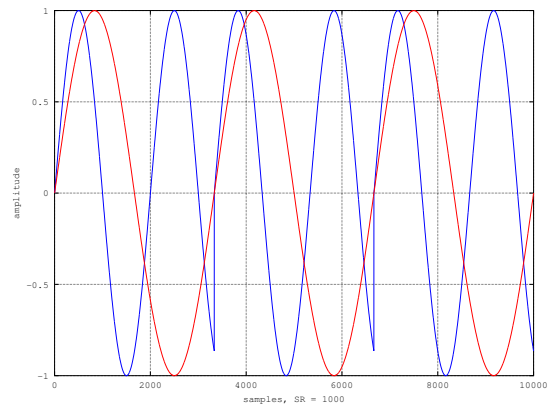
2.3.3 a new way to view synthesis

2.4 What’s Next?

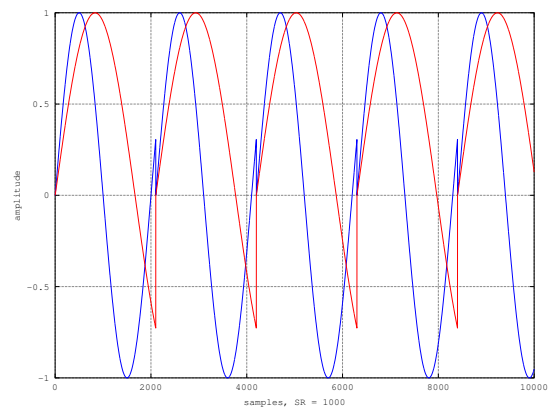
3 Timelab

3.1 Brief Overview

3.2 Examples



(a) red syncs blue



4
(b) misaligned sync

Figure 1: On the left, the blue oscillator is slaved to the red. On the right each is slaved to the other, but notice the ‘hiccups’ due to non-zero-time delay between the oscillators.