FDN & Spectral Processing - eine Projektarbeit im Rahmen der Veranstaltung: Signalverarbeitung und Audiotechnik

Simon Johannes Braun
Matrikelnummer: 091603717

1 Feedback Network Delay

Im Seminar Signalverarbeitung im Wintersemester 19/20 haben wir in Gruppenarbeit und als gemeinsames Projekt ein Feedback Delay Network entwickelt. Nach gemeinsamen konzeptionellen Gesprächen, übernahmen wir, die Seminarteilnehmer, die eigenständige Entwicklung einzelner Komponenten, die anschließend zu dem Endergebnis zusammengeführt wurden.

Das Feedback Delay Network besteht aus mehreren eigenständigen Klangerzeugern und der Aufnahme und Verarbeitung von Umgebungsgeräuschen. Das aufgenommene Audiosignal wie auch die intern generierten Audiosignale werden mit Hilfe der "Fast Fourier Transformation" in einzelne Frequenzspektren aufgeteilt. Diese einzelnen analysierten Frequenzen werden danach wiederum an die internen Klangerzeuger übergeben, die mit diesen Frequenzen neue Audiosignale erzeugen. Dieser Prozess wiederholt sich in zufälligen Abständen in einer Endlosschleife. Dadurch wird stetig neuer Raumklang erzeugt, der gleichzeitig wieder aufgenommen, analysiert, re-synthetisiert und wiedergegeben wird.

2 Fast Fourier Transformation

Die Fast Fourier Transformation (FFT) ist ein optimierter Algorithmus zur Implementierung der "Diskreten Fourier Transformation"[1]. Dabei wird ein bestimmter und begrenzter Anteil eines Signals in seine Bestandteile zu einzelnen Sinusschwingungen zerlegt. Deren jeweilige Amplituden und Phasen können dann bestimmt werden. Um ein Signal auf werte in einem bestimmten Frequenzbereich zu untersuchen, ist für die FFT die geeignete Abtastrate und Abtastwerte (Samples) zu bestimmen. Der Wert der Abtastrate ist mindesten doppelt so groß zu wählen, wie die höchste zu analysierende Frequenz sein soll. Also legt der halbe Wert der Abtastrate die Obergrenze für das Frequenzspektrum fest, mit dem das Signal untersucht werden kann. Die Samples bestimmen die Auflösung der einzelnen Amplituden-Messwerte. Je mehr Samples, desto genauer aber auch aufwendiger die Berechnung einzelner Amplituden zu bestimmten Zeitpunkten. Bei einer Abtastrate von 48000 Hz und einer Samplerate von 1024 `(2^10)` können so Frequenzunterschiede in 46.88 Hz Schritten `(48000 Hz/1024 )` über eine Messdauer von 21.33 ms `(1024/48000 H)` ermittelt werden. Da ein bestimmter Bestandteil eines Signals analysiert werden kann, ist es wahrscheinlich, dass die Amplituden am Anfang und am Ende des Signalausschnitts nicht null sind.[2] Das führt bei der FFT zu unerwünschten Fehl-

[1] McGee, Kevin J., An Introduction to Signal Processing and Fast Fourier Transform, [online] http://www.fftguru.com/fftguru.com.tutorial.pdf [14.01.2020]

[2] National Instruments, Understanding FFTs and Windowing, [online] https://download.ni.com/evaluation/pxi/Understanding%20FFTs%20and%20Windowing.pdf [14.01.2020]

interpretationen, die sich in Frequenzen bemerkbar machen, die im Ursprungssignal gar nicht vorhanden sind. Um dem entgegenzuwirken, wird die Methode des Fensterns (windowing) angewandt. Dabei wird der Signalausschnitts am Anfang und am Ende mit null multipliziert (z.B. Hamming).

3 drone.csd

Das Feedback Delay Network verarbeitet akustische Signale. Die nach der FFT berechnetet Frequenzen und Amplituden werden gespeichert und können durch programmierte Instrumente in CSound wiedergegeben werden. Ich hatte die Idee solch einen Synthesizer für das Feedback Delay Network zu entwickeln. Dieser Synthesizer soll anhand von einer zugewiesenen Frequenz eine gedehnte Klangfläche erzeugen, die sich über einen bestimmten Zeitraum auf- und abbaut. Der folgende Quellcode zeigt umsetzung des Instruments in CSound.

instr 1

```
<CsoundSynthesizer>
<CsInstruments>
instr 1

shift       =           .00666667                       ;shift it 8/1200.
ipitch      =           cpspch(p5)                      ;convert parameter 5 to cps.
ioct        =           octpch(p5)                      ;convert parameter 5 to oct.
kadsr       linseg      0, p3/3, 1.0, p3/3, 1.0, p3/6, 0    ;ADSR envelope for output
kmod        linseg      0, p3/3, 5, p3/3, 3, p3/6, 0        ;ADSR envelope for I
kenv        linseg      p6, p3, p7                      ;moves from p6->p7 in p3 sec.
;ares        oscil       1, ipitch, 21
ares2       oscil       1, ioct, 21
kamp1       =           kmod*((kenv-1)/kenv)/2
kamp2       =           kmod*((kenv+1)/kenv)/2

aindx1      =           abs(kamp1*1/20)                 ;kamp1 normalized between 0-1. for tbl 23

aamp3       tablei      aindx1, 23, 1                   ;lookup tbl in f23, normal index

aosc1       oscil       aamp3, ipitch, 22               ;cosine
aosc2       oscil       kamp2*ioct,ipitch, 22           ;cosine
```

```
64
65  aamp4       =           exp(-0.5*aamp3+aosc1)
66
67  aoutl       oscil       0.05*kadsr*aamp4, aosc2+cpsoct(ioct+ishift), 21 ;fnl outleft
68  aoutr       oscil       0.05*kadsr*aamp4, aosc2+cpsoct(ioct-ishift), 21 ;fnl outright
69
70              outs        Sigmoide(aoutl), Sigmoide(aoutr)
71
72  endin
73  </CsInstruments>
74  </CsoundSynthesizer>
```

Das Instrument `instr 1` greift auf den zugehörigen `i1` Werte aus dem `<CsScore>` zu. Es werden mehrere Instrument-Instanzen gleichzeitig und nacheinander gestartet. Dort sind unter dem Parameter `p5` die verschiedenen Ausgangsfrequenzen für das Instrument angegeben. Um vereinfachter und zugänglicher mehrtönige Klänge zu erzeugen werden diese Frequenzen unter `p5` als Kommazahlen angegeben, die harmonische Töne in Oktaven-Intervallen repräsentieren. Da für die Klangerzeugung in CSound Oszillatoren `oscil` die eine ganzahleige Frequenzangabe als eingabe Parameter erwarten, werden die Zahlen aus dem `i1` table mit `ioct = octpch(p5)` in Oktaven und mit `ipitch = cpspch(p5)` in Schwingungen pro Sekunde (Frequenzen) umgerechnet. Um ein langsames aufbauen und wieder abflachen des Klanges zu erreichen, wird eine Hüllkurve `kadsr` verwendet, die mit der Instanzdauer des aufgerufenen Instruments `p3` einen linearen Pegelverlauf von 0 über 1 bis 0 auf das Ausgangssignal anwendet. Zwei weitere Hüllkurven `kmod` und `kenv` werden für die letztendliche zusätzliche Modulation der Amplitude des Ausgabe Oszillatoren verwendet. Dafür werden weitere Zwischenschritte durchlaufen. Der Wert von `kamp1` wird durch `kmod*((kenv-1)/kenv)/2` berechnet und wird in einem weiteren Schritt für die Erzeugen eines Indexwertes verwendet `aindx1 = abs(kamp1*1.5/20)`. Dieser Indexwert ist auf einen Wertebereich zwischen 0 und 1 normalisiert und wird zum Abfragen eines Wertes aus einer Besselfunktion (Zylinderfunktion) `f23` verwendet, der wiederum die Amplitude eines weiteren Oszillator `aosc1` bestimmt. Dieser Oszillator verwendet die Frequenzen `ipitch` aus dem `<CsScore>`.

ftable

```
<CsoundSynthesizer>
<CsScore>
;f tables for instr 1
f21 0 8192 10 1                                         ;hughres sine wave
f22 0 8192 11 1                                         ;highres cosine wave
f23 0 8192 -12 20.0                                     ;unscaled ln(I(x)) from 0 to 20.0
</CsScore>
</CsoundSynthesizer>
```

Ein Weiterer Oszillator `aosc2` wird mit der Grundfrequenz `ipitch` erstellt, dessen Schwingungen die Frequenz des Ausgabe Oszillator modelliert `aosc2+cpsoct(ioct+ishift)`. `ishift` sorgt für eine zusätzliche unterschiedliche Tonverschiebung auf dem rechten und linken Ausgangssignal.

Ausschnitt aus CsScore

```
<CsoundSynthesizer>
<CsScore>
;instr1    start    dur     gain    freq    amp    amp2
;p1        p2       p3      p4      p5      p6     p7


i1 0 80 1  4.01 1.0 0.1
i1 . . .  4.05 . .
i1 . . .  4.08 . .
i1 . . .  3.01 . .
i1 . . .  3.05 . .
i1 . . .  3.08 . .

i1 40 80 1  4.08 1.0 0.1
i1 . . .  4.12 . .
i1 . . .  5.03 . .
i1 . . .  3.08 . .
i1 . . .  3.12 . .
i1 . . .  4.03 . .

```

```
i1 80 100 1  5.12 1.0 0.1
i1 . . .  6.03 . .
i1 . . .  6.07 . .
i1 . . .  4.12 . .
i1 . . .  5.03 . .
i1 . . .  5.07 . .

i1 120 100 1 5.06 1.0 0.1
i1 . . .  5.10 . .
i1 . . .  6.01 . .
i1 . . .  4.06 . .
i1 . . .  4.10 . .
i1 . . .  5.01 . .
</CsScore>
</CsoundSynthesizer>
```

Nach dem Zusammenfügen der einzelnen Arbeiten, ist in Gruppenarbeit eine interessante, sich immer wieder neu generierende, Klanginstallation entstanden. Dies wurde bisher schon zweimal öffentlich aufgebaut. Zuhörer konnten Im E-Werk Freiburg und im Jos-Fritz Kaffee Freiburg den Klängen lauschen und durch aktives erzeugen von Geräuschen selbst teil der Klanginstallation werden.

---

drone1.csd

```
<CsoundSynthesizer>
<CsOptions>
-o dac
;-i dac
</CsOptions>
<CsInstruments>

sr = 44100
ksmps = 32
nchnls = 2
0dbfs = 1

opcode Sigmoide, a, a
    aX xin
    aPow = exp(aX*3.0)
    aOut = 2.0 * (1.0 / (1.0 + aPow))- 1.0
    xout aOut
endop

instr 1

ishift      =           .00666667                       ;shift it 8/1200.
ipitch      =           cpspch(p5)                      ;convert parameter 5 to cps.
ioct        =           octpch(p5)                      ;convert parameter 5 to oct.
kadsr       linseg      0, p3/3, 1.0, p3/3, 1.0, p3/6, 0    ;ADSR envelope for output
kmod        linseg      0, p3/3, 5, p3/3, 3, p3/6, 0        ;ADSR envelope for I
kenv        linseg      p6, p3, p7                      ;moves from p6->p7 in p3 sec.
;ares        oscil       1, ipitch, 21
ares2       oscil       1, ioct, 21
kamp1       =           kmod*((kenv-1)/kenv)/2
kamp2       =           kmod*((kenv+1)/kenv)/2

aindx1      =           abs(kamp1*1/20)                 ;kamp1 normalized between 0-1. for tbl 23

aamp3       tablei      aindx1, 23, 1                   ;lookup tbl in f23, normal index

aosc1       oscil       aamp3, ipitch, 22               ;cosine
aosc2       oscil       kamp2*ioct,ipitch, 22           ;cosine

```

```
aamp4       =           exp(-0.5*aamp3+aosc1)

aoutl       oscil       0.05*kadsr*aamp4, aosc2+cpsoct(ioct+ishift), 21 ;fnl outleft
aoutr       oscil       0.05*kadsr*aamp4, aosc2+cpsoct(ioct-ishift), 21 ;fnl outright

            outs        Sigmoide(aoutl), Sigmoide(aoutr)

endin


</CsInstruments>
<CsScore>

;f tables for instr
f1  0 512   9 1 1 0                                 ;sine lo-res
f2  0 512   5 4096 512 1                            ;exp env
f3  0 512   9 10 1 0 16 1.5 0 22 2 0 23 1.5 0       ;inharm wave
f4  0 512   9 1 1 0                                 ;sine
f8  0 512   5 256 512 1                             ;exp env
f9  0 512   5 1   512 1                             ;constant value of 1
f10 0 512   7 0 50 1 50 .5 300 .5 112 0             ;ADSR
f11 0 2048 10 1                                     ;SINE WAVE hi-res
f13 0 1024  7 0 256 1 256 0 256 -1 256 0            ;triangle
f14 0 512   7 1 17  1 0   0 495                     ;pulse for S&H clk osc
f15 0 512   7 0 512 1 0                             ;ramp up;;;left=>right
f16 0 512   7 1 512 0 0                             ;ramp down;;;right=>left
f17 0 1024  7 .5 256 1 256 .5 256 0 256 .5          ;triangle with offset
f18 0 512   5 1 512 256                             ;reverse exp env
f20 0 1024 10 1 0 0 0 .7 .7 .7 .7 .7 .7             ;approaching square

;f tables for instr 3
f21 0 8192 10 1                                     ;hughres sine wave
f22 0 8192 11 1                                     ;highres cosine wave
f23 0 8192 -12 20.0                                 ;unscaled ln(I(x)) from 0 to 20.0



;instr3    start    dur     gain    freq    amp     amp2
;p1        p2       p3      p4      p5      p6      p7

```

```
246
247 ; p5 shift octaves
248 i1 0 5 1  4.01 1.0 0.1
249 i1 . . .  4.05 . .
250 i1 . . .  4.08 . .
251 i1 . . .  3.01 . .
252 i1 . . .  3.05 . .
253 i1 . . .  3.08 . .
254 i1 . . .  5.01 . .
255 i1 . . .  5.05 . .
256 i1 . . .  5.08 . .
257 i1 . . .  6.01 . .
258 i1 . . .  6.05 . .
259 i1 . . .  6.08 . .
260
261 i1 5 5 1  4.08 1.0 0.1
262 i1 . . .  4.12 . .
263 i1 . . .  5.03 . .
264 i1 . . .  3.08 . .
265 i1 . . .  3.12 . .
266 i1 . . .  4.03 . .
267 i1 . . .  5.08 . .
268 i1 . . .  5.12 . .
269 i1 . . .  6.03 . .
270 i1 . . .  6.08 . .
271 i1 . . .  6.12 . .
272 i1 . . .  7.03 . .
273
274 i1 10 5 1  4.12 1.0 0.1
275 i1 . . .  5.03 . .
276 i1 . . .  5.07 . .
277 i1 . . .  3.12 . .
278 i1 . . .  4.03 . .
279 i1 . . .  4.07 . .
280 i1 . . .  5.12 . .
281 i1 . . .  6.03 . .
282 i1 . . .  6.07 . .
283 i1 . . .  6.12 . .
284 i1 . . .  7.03 . .
285 i1 . . .  7.07 . .
```

```
i1 15 5 1 5.06 1.0 0.1
i1 . . .  5.10 . .
i1 . . .  6.01 . .
i1 . . .  4.06 . .
i1 . . .  4.10 . .
i1 . . .  5.01 . .
i1 . . .  6.06 . .
i1 . . .  6.10 . .
i1 . . .  7.01 . .
i1 . . .  7.06 . .
i1 . . .  7.10 . .
i1 . . .  8.01 . .

i1 0 60 1  4.01 1.0 0.1
i1 . . .  4.05 . .
i1 . . .  4.08 . .
i1 . . .  3.01 . .
i1 . . .  3.05 . .
i1 . . .  3.08 . .

i1 20 60 1  4.08 1.0 0.1
i1 . . .  4.12 . .
i1 . . .  5.03 . .
i1 . . .  3.08 . .
i1 . . .  3.12 . .
i1 . . .  4.03 . .

i1 60 60 1  5.12 1.0 0.1
i1 . . .  6.03 . .
i1 . . .  6.07 . .
i1 . . .  4.12 . .
i1 . . .  5.03 . .
i1 . . .  5.07 . .

i1 80 60 1 5.06 1.0 0.1
i1 . . .  5.10 . .
i1 . . .  6.01 . .
i1 . . .  4.06 . .
i1 . . .  4.10 . .
```

```
i1 . . .  5.01 . .

,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,

i1 100 40 1  4.01 1.0 0.1
i1 . . .  4.05 . .
i1 . . .  4.08 . .
i1 . . .  3.01 . .
i1 . . .  3.05 . .
i1 . . .  3.08 . .
i1 . . .  5.01 . .
i1 . . .  5.05 . .
i1 . . .  5.08 . .

i1 120 40 1  4.08 1.0 0.1
i1 . . .  4.12 . .
i1 . . .  5.03 . .
i1 . . .  3.08 . .
i1 . . .  3.12 . .
i1 . . .  4.03 . .
i1 . . .  5.08 . .
i1 . . .  5.12 . .
i1 . . .  6.03 . .

i1 140 40 1  5.12 1.0 0.1
i1 . . .  6.03 . .
i1 . . .  6.07 . .
i1 . . .  4.12 . .
i1 . . .  5.03 . .
i1 . . .  5.07 . .
i1 . . .  6.12 . .
i1 . . .  7.03 . .
i1 . . .  7.07 . .

i1 160 60 1 5.06 1.0 0.1
i1 . . .  5.10 . .
i1 . . .  6.01 . .
i1 . . .  4.06 . .
i1 . . .  4.10 . .
```

```
i1 . . .  5.01 . .
i1 . . .  6.06 . .
i1 . . .  6.10 . .
i1 . . .  7.01 . .

,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,

i1 180 40 1  4.01 1.0 0.1
i1 . . .  4.05 . .
i1 . . .  4.08 . .
i1 . . .  3.01 . .
i1 . . .  3.05 . .
i1 . . .  3.08 . .
i1 . . .  5.01 . .
i1 . . .  5.05 . .
i1 . . .  5.08 . .
i1 . . .  6.01 . .
i1 . . .  6.05 . .
i1 . . .  6.08 . .


i1 200 40 1  4.08 1.0 0.1
i1 . . .  4.12 . .
i1 . . .  5.03 . .
i1 . . .  3.08 . .
i1 . . .  3.12 . .
i1 . . .  4.03 . .
i1 . . .  5.08 . .
i1 . . .  5.12 . .
i1 . . .  6.03 . .
i1 . . .  6.08 . .
i1 . . .  6.12 . .
i1 . . .  7.03 . .


i1 220 40 1  4.12 1.0 0.1
i1 . . .  5.03 . .
i1 . . .  5.07 . .
i1 . . .  3.12 . .
```

```
i1 . . .  4.03 . .
i1 . . .  4.07 . .
i1 . . .  5.12 . .
i1 . . .  6.03 . .
i1 . . .  6.07 . .
i1 . . .  6.12 . .
i1 . . .  7.03 . .
i1 . . .  7.07 . .

i1 240 40 1 5.06 1.0 0.1
i1 . . .  5.10 . .
i1 . . .  6.01 . .
i1 . . .  4.06 . .
i1 . . .  4.10 . .
i1 . . .  5.01 . .
i1 . . .  6.06 . .
i1 . . .  6.10 . .
i1 . . .  7.01 . .
i1 . . .  7.06 . .
i1 . . .  7.10 . .
i1 . . .  8.01 . .

,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,

i1 260 40 1  4.01 1.0 0.1
i1 . . .  4.05 . .
i1 . . .  4.08 . .
i1 . . .  3.01 . .
i1 . . .  3.05 . .
i1 . . .  3.08 . .
i1 . . .  5.01 . .
i1 . . .  5.05 . .
i1 . . .  5.08 . .

i1 280 40 1  4.08 1.0 0.1
i1 . . .  4.12 . .
i1 . . .  5.03 . .
i1 . . .  3.08 . .
i1 . . .  3.12 . .
```

```
i1 . . .  4.03 . .
i1 . . .  5.08 . .
i1 . . .  5.12 . .
i1 . . .  6.03 . .

i1 300 40 1  5.12 1.0 0.1
i1 . . .  6.03 . .
i1 . . .  6.07 . .
i1 . . .  4.12 . .
i1 . . .  5.03 . .
i1 . . .  5.07 . .
i1 . . .  6.12 . .
i1 . . .  7.03 . .
i1 . . .  7.07 . .

i1 320 40 1 5.06 1.0 0.1
i1 . . .  5.10 . .
i1 . . .  6.01 . .
i1 . . .  4.06 . .
i1 . . .  4.10 . .
i1 . . .  5.01 . .
i1 . . .  6.06 . .
i1 . . .  6.10 . .
i1 . . .  7.01 . .

,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,

i1 340 60 1  4.01 1.0 0.1
i1 . . .  4.05 . .
i1 . . .  4.08 . .
i1 . . .  3.01 . .
i1 . . .  3.05 . .
i1 . . .  3.08 . .
i1 . . .  5.01 . .
i1 . . .  5.05 . .
i1 . . .  5.08 . .

</CsScore>
</CsoundSynthesizer>
```