

# 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"<sup>1</sup>. 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 mindestens 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 ( $48000 \text{ Hz} / 1024$ ) über eine Messdauer von 21.33 ms ( $1024 / 48000 \text{ s}$ ) 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.<sup>2</sup> Das führt bei der FFT zu unerwünschten Fehl-

---

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

<sup>2</sup>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 Fensters (windowing) angewandt. Dabei wird der Signalausschnitt am Anfang und am Ende mit null multipliziert. (z.B. Hamming).

### 3 drone.csd

```

43 <CsoundSynthesizer>
44 <CsInstruments>
45 instr 3
46
47 ishift      =      .00666667      ;shift it 8/1200.
48 ipch        =      cpspch(p5)      ;convert parameter 5 to cps.
49 ioct         =      octpch(p5)      ;convert parameter 5 to oct.
50 kadsr        linseg 0, p3/3, 1.0, p3/3, 1.0, p3/3, 0 ;ADSR envelope
51 kmodi         linseg 0, p3/3, 5, p3/3, 3, p3/3, 0 ;ADSR envelope for I
52 kmodr         linseg p6, p3, p7      ;r moves from p6->p7 in p3 sec.
53 a1           =      kmodi*(kmodr-1/kmodr)/2
54 a1ndx         =      abs(a1*2/20)      ;a1*2 is normalized from 0-1.
55 a2           =      kmodi*(kmodr+1/kmodr)/2
56 a3           tablei a1ndx, 23, 1      ;lookup tbl in f23, normal index
57 ao1           oscil a1, ipch, 22      ;cosine
58 a4           =      exp(-0.5*a3+ao1)
59 ao2           oscil a2*ipch, ipch, 22 ;cosine
60 aoutl         oscil 0.05*kadsr*a4, ao2+cpsoct(ioct+ishift), 21 ;fml outleft
61 aoutr         oscil 0.05*kadsr*a4, ao2+cpsoct(ioct-ishift), 21 ;fml outright
62              outs      Sigmoid(aoutl), Sigmoid(aoutr)-
63
64 endin
65 </CsInstruments>
66 </CsoundSynthesizer>

```

```

162 <CsoundSynthesizer>
163 <CsScore>
164 ;f tables for instr 3
165 f21 0 8192 10 1      ;hughres sine wave
166 f22 0 8192 11 1      ;hughres cosine wave

```

```

167 f23 0 8192 -12 20.0 ;unscaled ln(I(x)) from 0 to 20.0
168 </CsScore>
169 </CsoundSynthesizer>

```

```

175 <CsoundSynthesizer>
176 <CsScore>
177 ;instr3 start dur gain freq amp amp2
178 ;p1 p2 p3 p4 p5 p6 p7
179
180
181 ; p5 shift octaves
182 i3 0 30 1 4.06 1.0 0.2
183 i3 . . . 5.01 . .
184 i3 . . . 5.06 . .
185 i3 . . . 5.10 . .
186 i3 . . . 5.11 . .
187 i3 . . . 6.04 . .
188
189 i3 10 15 1 5.06 0.8 0.2
190 i3 . . . 6.01 . .
191 i3 . . . 6.06 . .
192 i3 . . . 6.10 . .
193 i3 . . . 6.11 . .
194 i3 . . . 7.04 . .
195
196 i3 20 80 1 4.06 1.0 0.2
197 i3 . . . 5.01 . .
198 i3 . . . 4.06 . .
199 i3 . . . 4.10 . .
200 i3 . . . 4.11 . .
201 i3 . . . 4.04 . .
202 </CsScore>
203 </CsoundSynthesizer>

```

```

167 <CsoundSynthesizer>
168 <CsOptions>

```

```

169 -o dac
170 ;-i dac
171 </CsOptions>
172 <CsInstruments>
173
174 sr = 44100
175 ksmps = 32
176 nchnls = 2
177 0dbfs = 1
178
179
180 gip init 0 ;global playtime
181
182 gkfreq init 0
183 gkrate init 0
184 gkbandw init 0
185
186 ;gkfreq invalue 'freq'
187 ;gkrate invalue 'rate'
188 ;gkbandw invalue 'bandw'
189
190 opcode Sigmoid, a, a
191     aX xin
192     aPow = exp(aX*3.0)
193     aOut = 2.0 * (1.0 / (1.0 + aPow))- 1.0
194     xout aOut
195 endop
196
197
198 instr 1
199
200 anoise pinkish 1
201 kf = p4
202 kbandw = 10
203 alp butbp anoise*p5, kf, kbandw
204
205 outs alp, alp
206
207 endin
208

```

```

209 instr 3
210
211 ishift = .00666667 ;shift it 8/1200.
212 ipch = cpspch(p5) ;convert parameter 5 to cps.
213 ioct = octpch(p5) ;convert parameter 5 to oct.
214 kadsr linseg 0, p3/3, 1.0, p3/3, 1.0, p3/3, 0 ;ADSR envelope
215 kmodi linseg 0, p3/3, 5, p3/3, 3, p3/3, 0 ;ADSR envelope for I
216 kmodr linseg p6, p3, p7 ;r moves from p6->p7 in p3 sec.
217 a1 = kmodi*(kmodr-1/kmodr)/2
218 a1ndx = abs(a1*2/20) ;a1*2 is normalized from 0-1.
219 a2 = kmodi*(kmodr+1/kmodr)/2
220 a3 tablei a1ndx, 23, 1 ;lookup tbl in f23, normal index
221 ao1 oscil a1, ipch, 22 ;cosine
222 a4 = exp(-0.5*a3+ao1)
223 ao2 oscil a2*ipch, ipch, 22 ;cosine
224 aoutl oscil p4*0.05*kadsr*a4, ao2+cpsoct(ioct+ishift), 21 ;fml outleft
225 aoutr oscil p4*0.05*kadsr*a4, ao2+cpsoct(ioct-ishift), 21 ;fml outright
226 outs Sigmoid(aoutl), Sigmoid(aoutr)
227
228 endin
229
230 instr 2
231
232 ifn = p7 ;f table number
233
234 i1 = p6 ; INIT VALUES CORRESPOND TO FREQ.
235 i2 = 2*p6 ; OFFSETS FOR OSCILLATORS BASED ON ORIGINAL p6
236 i3 = 3*p6
237 i4 = 4*p6
238
239 ampenv linen p5,30,p3,30 ; ENVELOPE
240
241
242 a1 oscili ampenv,p4,ifn
243 a2 oscili ampenv,p4+i1,ifn ; NINE OSCILLATORS WITH THE SAME AMPENV
244 a3 oscili ampenv,p4+i2,ifn ; AND WAVEFORM, BUT SLIGHTLY DIFFERENT
245 a4 oscili ampenv,p4+i3,ifn ; FREQUENCIES TO CREATE THE BEATING EFFECT
246 a5 oscili ampenv,p4+i4,ifn
247 a6 oscili ampenv,p4-i1,ifn ; p4 = freq OF FUNDAMENTAL (Hz)
248 a7 oscili ampenv,p4-i2,ifn ; p5 = AMP

```

```

249 a8      oscili    ampenv,p4-i3,ifn      ; p6 = INITIAL OFFSET OF FREQ - .03 Hz
250 a9      oscili    ampenv,p4-i4,ifn
251
252 ;asnd     =        (a1+a2+a3+a4+a5+a6+a7+a8+a9)/9
253
254 asum     =        (a1+a3+a5+a7+a9+a2+a4+a6+a8)/p5
255
256 ;outs     a1+a2+a3+a4,a5+a6+a7+a8+a9
257
258
259 outs     asum, asum
260
261
262 ;garvbsig =        garvbsig+(asnd*.85)
263
264
265 endin
266
267 instr 4
268
269 ifn = 1      ;choose wave form table
270 ktrig metro 10
271 kGate      randomi 0, 1, 3
272 kdB        randomi 0, 12, 5
273
274
275 if kGate > 0.5 then      ;if kGate is larger than 0.5
276 kVol       =          kdB/12      ; open gate
277 else
278 kVol       =          0           ;otherwise close gate
279 endif
280 kVol       port      kVol, .02    ;smooth volume curve to avoid clicks
281
282 kamp linseg 0, 1, 0.2, 1, 0
283
284 aSig1 oscil3 kamp, 60, ifn
285 aSig2 oscil3 kamp, 120, ifn
286 aSig3 oscil3 kamp, 240, ifn
287 aSig4 oscil3 kamp, 320, ifn
288 aSig5 oscil3 kamp, 440, ifn

```

```

289
290 aSig6 oscil3 kVol, 60, ifn
291 aSig7 oscil3 kVol, 120, ifn
292
293 ;aPhas phasor 300
294
295 ;aSig2 vco2 0.5, 300
296
297 ;aPhas poscil kVol,60,2
298
299 aSigSum = aSig1 + aSig2 + aSig3 + aSig4 + aSig5 + aSig6 + aSig7
300
301 arev reverb aSigSum, 2
302
303 outs arev, arev
304 endin
305
306
307 </CsInstruments>
308
309 <CsScore>
310
311 ;f tables for instr 2
312 f1 0 512 9 1 1 0 ;sine lo-res
313 f2 0 512 5 4096 512 1 ;exp env
314 f3 0 512 9 10 1 0 16 1.5 0 22 2 0 23 1.5 0 ;inharm wave
315 f4 0 512 9 1 1 0 ;sine
316 f8 0 512 5 256 512 1 ;exp env
317 f9 0 512 5 1 512 1 ;constant value of 1
318 f10 0 512 7 0 50 1 50 .5 300 .5 112 0 ;ADSR
319 f11 0 2048 10 1 ;SINE WAVE hi-res
320 f13 0 1024 7 0 256 1 256 0 256 -1 256 0 ;triangle
321 f14 0 512 7 1 17 1 0 0 495 ;pulse for S&H clk osc
322 f15 0 512 7 0 512 1 0 ;ramp up;;;left=>right
323 f16 0 512 7 1 512 0 0 ;ramp down;;;right=>left
324 f17 0 1024 7 .5 256 1 256 .5 256 0 256 .5 ;triangle with offset
325 f18 0 512 5 1 512 256 ;reverse exp env
326 f20 0 1024 10 1 0 0 0 .7 .7 .7 .7 .7 .7 ;approaching square
327
328 ;f tables for instr 3

```



```

329 f21 0 8192 10 1 ;hughres sine wave
330 f22 0 8192 11 1 ;highres cosine wave
331 f23 0 8192 -12 20.0 ;unscaled ln(I(x)) from 0 to 20.0
332
333
334
335 ;instr      start      dur      freq      amp      offset      table number
336 ;p1         p2         p3         p4         p5         p6         p7
337 ;i1         0          100        120        10
338 ;i2         40         40        33        10        .03        20
339 ;i4         0          100
340
341 ;instr3      start      dur      gain      freq      amp      amp2
342 ;p1         p2         p3         p4         p5         p6         p7
343
344
345 ; p5 shift octaves
346 i3 0 30 1 4.06 1.0 0.2
347 i3 . . . 5.01 . .
348 i3 . . . 5.06 . .
349 i3 . . . 5.10 . .
350 i3 . . . 5.11 . .
351 i3 . . . 6.04 . .
352
353 i3 10 15 1 5.06 0.8 0.2
354 i3 . . . 6.01 . .
355 i3 . . . 6.06 . .
356 i3 . . . 6.10 . .
357 i3 . . . 6.11 . .
358 i3 . . . 7.04 . .
359
360 i3 20 80 1 4.06 1.0 0.2
361 i3 . . . 5.01 . .
362 i3 . . . 4.06 . .
363 i3 . . . 4.10 . .
364 i3 . . . 4.11 . .
365 i3 . . . 4.04 . .
366
367
368

```

```

369 i3 5 30 1 4.16 1.0 0.2
370 i3 . . . 5.11 . .
371 i3 . . . 5.16 . .
372 i3 . . . 5.20 . .
373 i3 . . . 5.21 . .
374 i3 . . . 6.14 . .
375
376 i3 15 15 1 5.16 0.8 0.2
377 i3 . . . 6.11 . .
378 i3 . . . 6.16 . .
379 i3 . . . 6.20 . .
380 i3 . . . 6.21 . .
381 i3 . . . 7.14 . .
382
383 i3 35 15 1 4.16 1.0 0.2
384 i3 . . . 5.16 . .
385 i3 . . . 5.11 . .
386 i3 . . . 5.20 . .
387 i3 . . . 5.21 . .
388 i3 . . . 5.14 . .
389
390
391
392 i3 15 30 1 4.03 1.0 0.2
393 i3 . . . 4.01 . .
394 i3 . . . 4.06 . .
395 i3 . . . 4.00 . .
396 i3 . . . 4.01 . .
397 i3 . . . 6.04 . .
398
399 i3 25 15 1 4.06 0.8 0.2
400 i3 . . . 4.11 . .
401 i3 . . . 4.16 . .
402 i3 . . . 4.20 . .
403 i3 . . . 4.21 . .
404 i3 . . . 4.14 . .
405
406 i3 45 15 1 4.06 1.0 0.2
407 i3 . . . 5.11 . .
408 i3 . . . 4.16 . .

```

```
409 i3 . . . 4.20 . .
410 i3 . . . 4.21 . .
411 i3 . . . 4.14 . .
412 </CsScore>
413
414 </CsoundSynthesizer>
```