„C-5 Nemlineáris multieffekt megvalósítása”

Zenei jelfeldolgozás

Budapesti Műszaki és Gazdaságtudományi Egyetem

Előszó

Ez a dokumentum a „C-5 Nemlineáris multieffekt megvalósítása” c. házi feladat dokumentációjaként készült a Budapesti Műszaki Egyetem 2013/2014/01 félévben induló Zenei jelfeldolgozás című kurzusára. A feladat pontos specifikációja a következő:

C-5 Nemlineáris multieffekt megvalósítása
Valósítson meg Compressor/Expandert és zajzárat. A változtatható paraméterek (logaritmikus skálán):
Compressor/Expander:
attack time for rms detection: 5 msec
release time for rms detection: 130 msec
attack time for C/E: 0,16-2600 msec
release time for C/E: 1-5000 msec
Zajzár:
Threshold: -10 - -60 dB
Attack: 10-100 ms
Release: 100-5000 msec
Ellenőrző pont: RMS-detektálás és kompresszor-karakterisztika megvalósítása.

Az elkészített feladat ezen dokumentáció mellett két általam implementált dinamikai multieffekt forráskódját tartalmazza. Tervezésük során próbáltam a követelményeknek maximálisan eleget tenni.

Szakállas Dávid
hallgató

Nemlineáris dinamikai effektek

1. Áttekintés

A dinamikai effektek manapság nélkülözhetetlenek a professzionális hangtechnikai stúdiókból. Még ha valaki amatőr szinten foglalkozik hangfeldolgozással, akkor is bizonyosan találkozott velük, hiszen a legtöbb hangmanipuláló szoftverben, kezdve az ingyenesen elérhető editoroktól, alapértelmezetten implementálva vannak. Ezen eszközök közös tulajdonsága az, hogy a bemenő hang dinamikai tartományát automatikusan módosítják. Tipikus feladatuk a jel hangerőtartományának csökkentése, vagy (ritkábban) növelése ami során egy bizonyos küszöbértéken túl az eszköz hangerőszabályzást hajt végre.

Habár funkciójuk a fenti mondattal meghatározható, az eltérő paraméterezésű eszközök a gyakorlatban más-más szerepet kapnak, ezek alapján kategóriákba soroljuk őket.

Kompresszor

A küszöbérték feletti hangokat csillapítja (downward compression), míg a halkabb hangokra nincs hatással. Egyértelmű, hogy ezáltal a hang dinamikai tartományát csökkenti. Rendkívül gyakran használt eszköz, egy tipikus keverési elrendezésben akár minden hangszersáv külön kompresszort kaphat.

Expander

Az expander a kompresszorral ellentétben a hang dinamikai tartományának növelésében játszik szerepet az által, hogy egy bizonyos küszöb alatti hangokon még inkább halkít. Használhatósága korlázott, amit nem találunk meglepőnek, ha figyelembe vesszük, hogy a stúdióban felvett hangok dinamikai tartománya általában eleve nagyobb, mint amit egy felvételen megkívánunk.

Limiter

Egyfajta komprosszor. A küszöbérték feletti hangok hangerejét a küszöbérték környékére csökkenti. Sok esetben hasznos, hiszen teljes mértékben kiszűri a túl erős hangokat.

Zajzár

Egyfajta expander. A küszöbérték alatti hangokat teljesen levágja. Nevéből sem nehéz kitalálni, hogy leginkább háttérzajok kiszűrésére használatos.

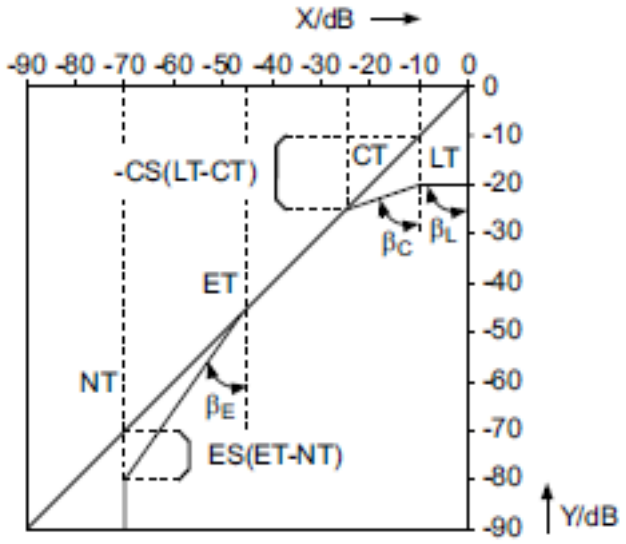
2. Egységes elméleti modell

Statikus görbe

A bemeneti jel mérése után a kimenetet egy súlyozú tényezővel befolyásoljuk:

$$y(n)=g(n)*x(n) \quad .$$

A kapcsolatot a bemeneti jelszint és a súlyozó függvény között egy statikus görbe határozza meg. A (z) 1. Ábra a kimenetet ábrázolja a bemenet függvényében.



1. Ábra: Statikus görbe

A statikus görbe paraméterei tehát a következőek:

- Zajzár köszübértéke (NT) [dB]
- Expander küszöbértéke (ET) [dB]
- Expanziós meredekség (ES)
- Kompresszor küszöbértéke (CT) [dB]
- Kompressziós meredekség (CS)
- Limiter küszöbértéke (LT) [dB]

Egy másik lehetőség az egyenesek meghatározására a kompressziós és expanziós arány, melyek a következőképpen számítandók:

$$R_C = \tan\beta_C \quad \text{valamint} \quad R_E = \tan\beta_E \quad .$$

Az arányból a meredekség az

$$S = 1 - \frac{1}{R}$$

képlettel kifejezhető.

Jelszintmérés

A bemenő jelszint mérése többféleképpen történhet. Az egyik megoldás a PEAK mérés, ami során a bemenő jel amplitúdójának abszolútértékét hasonlítjuk össze a csúcsértékkel ($x_{PEAK}(n)$). Ha a jelünk abszolútértéke a csúcsértéknél nagyobb, akkor a következő interpolációt alkalmazzuk:

$$x_{PEAK}(n) = (1 - AT) * x_{PEAK}(n-1) + AT * |x(n)| \quad ,$$

melynek átviteli függvénye:

$$H(z) = \frac{AT}{1 - (1 - AT) * z^{-1}} \quad .$$

Ha az abszolútérték kisebb mint a csúcsérték, akkor az új csúcsérték a következő képlettel adható meg:

$$x_{PEAK}(n) = (1 - RT) * x_{PEAK}(n-1) \quad .$$

Az AT együttható meghatározható az alábbi képlettel:

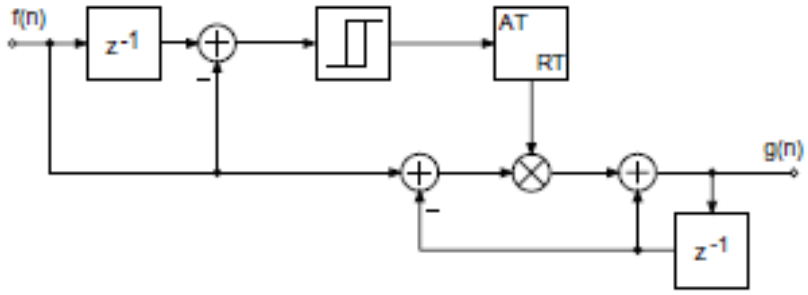
$$AT = 1 - \exp\left(\frac{-2.2\mathrm{T}_s}{t_a}\right) \quad , \text{ és az RT is, ezzel analóg módon.}$$

Másik módszer az RMS detektálás, ami a bemenő jel abszolútértéke helyett, annak négyzetével dolgozik.

Súlyozó függvény simítása

Az utolsó lépés az így keletkező függvény simítása (gain factor smoothing), amivel a felhasználó beállíthatja azt, hogy az eszköz milyen gyorsan reagáljon a jelszintváltásokra. Természetesen ezt az időt a jelszintmérésnél jelenlévő t_a és t_b paraméterek is befolyásolják, de szükségünk van egy ezeknél könnyebben állítható és jobban érthető reprezentációra. (Továbbá bizonyos implementációknál a jelszintmérő eszköz feketedoboznak tekinthető.)

Az ehhez a lépéshez tartozó blokkdiagram:



2. Ábra: Gain factor smoothing blokkvázlata

Képlettel:

$$g(n)=(1-k)*g(n-1)+k*f(n) \quad ,$$

ahol $k = AT$ vagy $k = RT$ attól függően, hogy le- vagy felfutó élet detektáltunk.

Implementáció

1. Audió plug-inok

Egy audió plug-in fejlesztése merőben eltér egy stand-alone alkalmazás fejlesztesétől, hiszen vannak olyan jellemzők amikre nagy hangsúlyt kell fektetni. Mindenféle programnál alapvető igény a használhatóság és a működőképesség, ezek természetesen itt is valósak. Ezeken felül az audió plug-inokkal szemben további speciális igényeket támasztunk:

- **valós-idejűnek kell lenniük**: a feldolgozási szálnak időben végeznie kell, különben a lejátszás szakadozni fog, vagy klikkek lesznek benne;
- **biztonság kritikusak:** ha az alkalmazás összeomlik, akkor a hoszt programot is magával rántja. Ez katasztrofális, és sajnos a valóságban sajnos túl sokszor megtörténik;
- **hibatűrőnek kell lenniük**: ha valami miatt az alkalmazás felülete nem válaszol, a feldolgozás akkor sem szakadhat meg.

Általános felépítésük a következő:

- **előkészítő/lezáró blokk**: a hoszt meghívja az előkészítő függvényt, hogy közölje, a feldolgozás megkezdődik. Lefutása után az effektnek már készen kell állnia a feldolgozásra. A lezáró blokk ezzel ellentétes értelmű.
- **processz blokk** az effekt lelke, ez a callback hívódik meg újabb minták érkezésekor, azaz itt kell a hangfeldolgozást végezni. Blokkolásmentesnek kell lennie, nem várhat például az alsóbbrendű GUI szálakra.
- **Getter/setterek**: a paramétereket lekérik, módosítják. Nincs rá garancia, hogy ki, mikor hívja meg (például processzálás közben is meghívódhat), úgyhogy szálbiztossá kell tenni.

2. Felhasznált keretrendszerek

Audió plug-inok készítéséhez számos keretrendszer épült, amik lehetővé teszik, hogy az általuk megírt plug-in, bármely, a keretrendszert támogató gazdaprogramban futni tudjon. A legnépszerűbbek:

- VST, a Steinberg cég tulajdonában;
- AU, amit az Apple készített, és nagyban hasonlít a VST technológiára;
- AAX (az RTAS utódja), ami az Avid cég tulajdonában van.

A szűkös idő miatt egyik interfész natív programozását sem sajátítottam el. Ehelyett egy JUCE (Jules' Utility Class Extensions) nevű nyílt forráskódú keretrendszert alkalmaztam, ami elfedi ezen interfészek sajátosságait. Habár a keretrendszer lehetővé teszi, az így megírt forráskódból AU vagy AAX plug-int készítsek, én csak a VST-t választottam.

VST

A VST interfészt a Steinberg vállalat adta fejleszti. Az eslú erziója 1996-ra datálódik. Az akkor megjelent effektek az Espacial (egy zengető), a Choirus (egy chorus), Stereo Echo és Auto-Panner voltak. 1999-ben érkezett a 2.0-ás verzió, ami már MIDI üzenetek fogadására is alkalmas, így lehetővé téve hangszerek készítését (VSTi). A későbbi verziók újabb funkciókkla gazdagtották a rendszert. Ma a 3.5-ös verziónál tartunk.

Én a VST 2.4-es verzióját használtam fel.

Megjegyzés: A VST zárt szoftver, így a forráskódját nem mellékelhetem az alkalmazás mellé.

JUCE

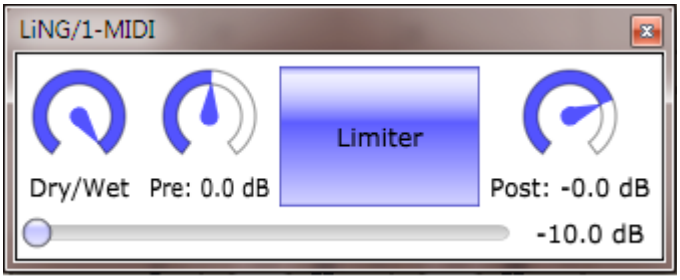
A JUCE egy osztálykönyvtár kifejezetten multimédia alkalmazások fejlesztésére. Legfőbb erénye, hogy multiplatform, és a mai napig fejlesztés alatt áll. Hihetetlen népszerűséget annak is köszönheti, hogy nem kereskedelmi célra felhasználható GPL licenc alatt. Weboldal: http://www.juce.com/

Rengeteg funkciója közül ezeket használtam fel:

- csomagolóosztályok audió plug-inok készítésére
- szálkezelés
- GUI könyvtárak
- OpenGL csomagolóosztályok

3. Az elkészült programok

LiNG



Minimalista limiter/zajzár. Alapvető funkciók megvalósítására képes. A limiter elég kemény, de gyors tranzienseket átengedhet, csak körültekintően használandó.

Paraméterek:

- *Dry/Wet*
- *Gain:* utóerősítés mértéke. Tartomány [dB]: *(-inf, +6)*
- *Threshold:* a limiter/zajzár küszöbértéke. Tartomány [dB]: *(-60, -10)*
- *Preamp:* utóerősítés mértéke. Tartomány [dB]: *(-inf, +12)*
- *Function:* limiter / zajzár között kapcsol.

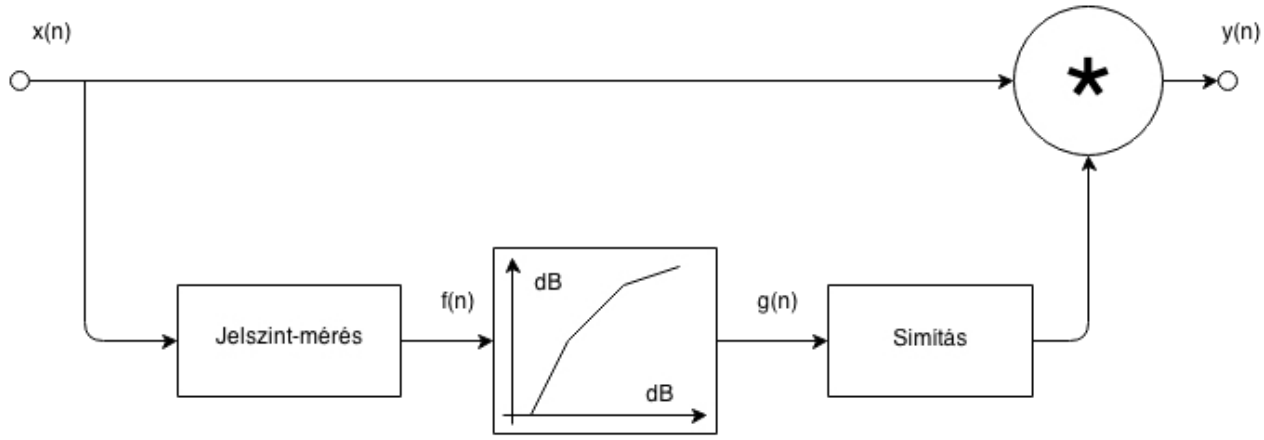
Kex

Specifikáció

Grafikai felülettel rendelkező kompresszor/expander plug-in. Az effektnek 11 kivezetett paramétere van, amelyek mindegyike automatizálható a hoszt program által.

Paraméterek:

- *Dry/Wet*
- *Gain:* utóerősítés mértéke. Tartomány [dB]: *(-inf, +6)*
- *Attack:* a simítógörbe paramétere. Ennyi idő telik el amíg küszöbértéket áthaladó jel maximálás feldolgozáson esik át. Tartomány [ms]: (*0.16, 2600)*
- *Release:* a simítógörbe pramétere. Ennyi idö telik el, amíg a küszöbérték alá csökkenő jelen a feldolgozás teljesen mgszűnik. Tartomány [ms]: *(1, 5000)*
- *Comp. Thres.:* a kompresszor küszöbértéke. Tartomány [dB]: *(-40, 0)*
- *Comp. Rat.:* a kompresszor arányténye zője *(ld. Egységes elméleti modell)*. Tartomány: *(1, 100)*
- *Exp. Thres.:* az expander küszöbértéke. Tartomány [dB]: *(-50, -14)*
- *Exp. Rat.:* a kompresszor arányténye zője *(ld. Egységes elméleti modell)*. Tartomány: *(0.05, 1)*
- *Level Meas.:* szintmérés módja. *Peak* vagy *RMS*.
- *Envelope:* a burkológörbe típusa. *LR* esetben a csatornák külön-külön szenvedik el a feldolgozást, így a sztereókép torzulhat. *Stereo* esetben, ha bármelyik csatorna feldolgozást igényel, az mindkét csatornát érinti.
- *Preamp:* utóerősítés mértéke. Tartomány [dB]: *(-inf, +12)*



3. Ábra: Csatornánként egy ilyen rendszerre van szükség (LR feldolgozásnál)

Pszeudokód (LR):

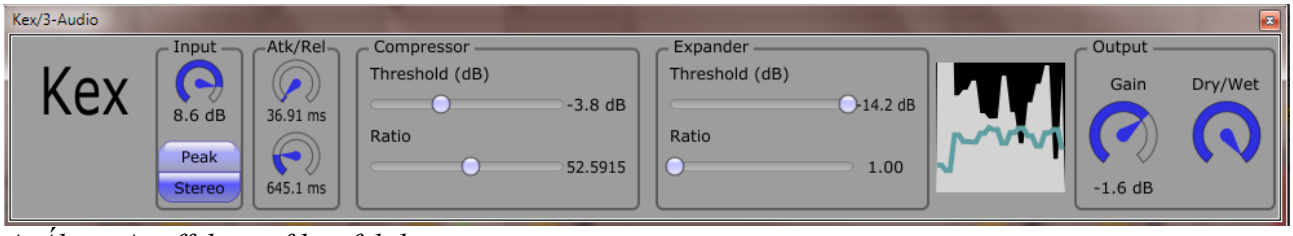
```
//Bufferek (késleltetések)
env[channelcount];
lvl[channelcount];
g[channelcount];

//Paraméterek
att, rel, matt, mrel, ct, et, cs, es, wet, preamp, gain;

function process_lr(
  buffer[][],
  samplerate)
begin
  foreach channel in buffer
  begin
        foreach sample in channel
        begin
              w, q, z, k;
              //Jelszint-mérés
              if rms then
                    w := sample * sample;
              else
                    w := abs(sample);

              if w > lvl[channel.index] then
                    lvl[channel.index] :=(1 - matt) * lvl[channel.index] +
                    matt * w;
              else
                    lvl[channel.index] := (1 - mrel) * lvl[channel.index];
              //Statikus görbe
              //Előbb a kompresszor...
              if lvl[channel.index] > ct then
                    q := exp10( -cs * (log10(lvl[channel.index]) – log10(ct)));
              else
                    q := 1;
              //..majd a módosított burkolón az expander
              if lvl[channel.index] < et then
                    q:= q * exp10( -es * (log10(lvl[channel.index]) –
                    log10(et)));
              //Simítás
              z:= env[channel.index – q;
              if z > .04 then k:=att;
              else begin
                    if z < -.04 then k:=rel;
                    else k:= (rel + att) * .5;

              g[channel.index]:= (1 - k) * g[channel.index] + k * q;
              env[channel.index]:= q;

              sample := wet * g[channel.index] * sample * preamp * gain + (1 -
              wet) * sample;
        end
  end
end
```

A program két fő komponensből áll, az effektprocesszorból, illetve a hozzá tartozó grafikus felületből. A grafikus felületen implementáltam egy megjelenítőt („szkópot”), ami OpenGL renderelést hajt végre. Ezen komponensek között a szálbiztosság garantálása – az effekt implementálása mellett – jelentette az elsődleges kihívást.

A kölcsönös kizárási szakaszokat úgy implementáltam, hogy azok a feldolgozó szálat a lehető legkisebb mértékben akadályozzák. Például, ha a feldolgozó szálnak frissíteni kéne egy változót, ami a GUI megjelenítéséhez tartozik, de a változó már zárolva van, a feldolgozó szál lemond a módosításról, ahelyett hogy várakozna. Másik esetben a feldolgozási szál a paramétereket beolvasná, hogy frissítse a belső állapotváltozóit. Ez esetben, ha a paraméterek zárolva vannak, a feldolgozó szintén lemond a frissítésről, és a régi állapotváltozóival dolgozik tovább.

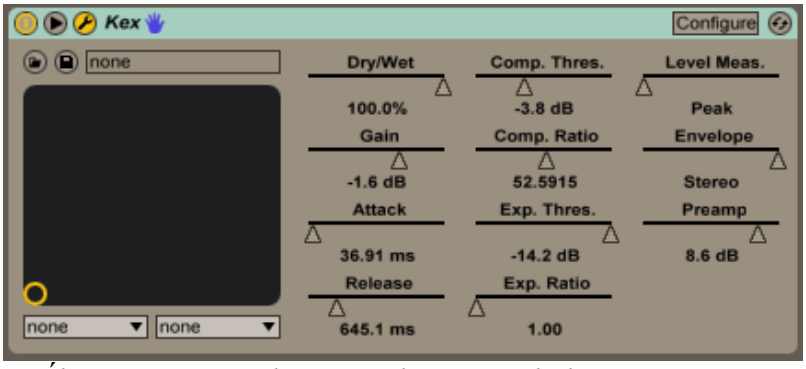
Használati útmutató

Az audió plug-in felhasználói felülettel rendelkezik, ezért a paraméterei könnyen állíthatók. A vezérlők mellé tettem egy kijelzőt, amin a feldolgozást nyomon követhetjük. A fehér görbe jelöli a bemeneti jelet (lineáris skálán), a kék pedig a súlyozófüggvény értékét (szintén lineáris skálán). A kettő szorzata jelentené a kimeneti jelszintet (ha a *Gain* ill. *D/W* paramétert nem tekintjük), de ennek ábrázolásától eltekintettem, mert a grafikon túl sűrű lenne vele.



4. Ábra: Az effekt grafikus felülete

A paraméterek ki vannak vezetve a gazdaprogram felületére, így automatizálhatóak is, habár erre általában (az effekt jellégéből fakadóan) nincs szükség.



5. Ábra: Paraméterek egy gazdaprogramból