UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Vedran Mutić

ANALIZA IN IZVEDBA GLASBENIH UČINKOV S PROGRAMSKIM ORODJEM SIMULINK

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

Mentor: prof. dr. Dušan Kodek

Ljubljana, 2010

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01620/2009
Datum: 15.11.2009

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **VEDRAN MUTIĆ**

Naslov: **ANALIZA IN IZVEDBA GLASBENIH UČINKOV S PROGRAMSKIM ORODJEM SIMULINK**

ANALYSIS AND IMPLEMENTATION OF MUSICAL EFFECTS WITH THE SIMULINK PROGRAM TOOL

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Analizirajte glasbene učinke kot so oktaviranje, popačenja, zakasnitve, modulacije in filtriranje, ki se uporabljajo pri današnjih električnih glasbenih instrumentih. S pomočjo orodja Simulink in na osnovi teorije digitalnega procesiranja signalov izdelajte možne rešitve za digitalno izvedbo teh učinkov. Rezultate ovrednotite grafično in akustično ter jih primerjajte z obstoječimi analognimi in digitalnimi rešitvami, ki jih pozna današnja industrija pripomočkov za električne glasbene instrumente. Ocenite probleme, ki bi jih prinesla realizacija teh učinkov v realnem času.

Mentor:
prof. dr. Dušan Kodek

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:
prof. dr. Franc Solina

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Vedran Mutić,

z vpisno številko 63000228,

sem avtor diplomskega dela z naslovom:

Analiza in izvedba glasbenih učinkov s programskim orodjem Simulink.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Dušana Kodeka;
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela;
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 8. 1. 2010 Podpis avtorja: _____

Zahvala

Največja zahvala gre staršema, ki sta mi omogočila študij. Mentorju prof. dr. Dušanu Kodeku se zahvaljujem za pomoč pri izdelavi tega dela, asistentu dr. Robertu Rozmanu pa za koristne nasvete pri zapletih med delom. Na tem mestu se želim zahvaliti tudi kolegom s Fakultete za računalništvo in informatiko, brez katerih bi veliko težje prišel do konca študija. Bernardki Ravnikar se zahvaljujem za lektoriranje dela. Posebna zahvala gre Jani za potrpljenje in podporo.

Kazalo strani

Kazalo slik

Kazalo grafov

Povzetek

Diplomsko delo vključuje analizo glasbenih učinkov in njihovo izvedbo s programskim orodjem Simulink. Opisanih je več različnih družin glasbenih učinkov, pri vsaki pa je razdelanih nekaj učinkov in načinov njihove digitalne izvedbe. Opis je dopolnjen s slikami modelov v programskem orodju Simulink. Vsak model je nato preizkušen z zvočno datoteko in tudi slušno ocenjen. Podani so grafi signalov pred in po izvedeni simulaciji. Programsko orodje Simulink, v kateremu je delo izvedeno, omogoča modeliranje dinamičnih sistemov in prevedbo modela v programsko kodo. Omenjeno kodo lahko prenesemo na razvojne ploščice in modele preizkusimo v realnem času. Predstavljeni so način dela in težave, ki so nastopile pri delu s programskim orodjem Simulink. Opisani učinki so iz družin učinkov oktaviranja, popačenja, zakasnitev, moduliranja in filtrov. Pri učinkih oktaviranja so razloženi fenomen oktave in nekaj elektronskih rešitev, po katerih so načrtovane digitalne izvedbe. Učinki popačenja predvsem posnemajo preobremenjene ojačevalce zvoka, zato je tu poudarek na različnih načinih simuliranja takšnega zvoka. Pri družini učinkov zakasnitev so raziskane različne vrednosti parametra zakasnitve, ki vplivajo na dojemanje zvoka v časovni in frekvenčni domeni. Družina modulacij vključuje opis najpogostejših tovrstnih učinkov. Pri učinkih filtrov je predelana digitalna simulacija manj pogoste elektronske izvedbe filtriranja. Na koncu je naveden seznam možnih razširitev dela.

Ključne besede:

zvočni učinki, oktava, popačenje, zakasnitev, modulacija, filter, Simulink.

Abstract

This thesis includes the analysis of musical effects and their realization with the Simulink program tool. Different groups of musical effects are described, along with the descriptions of effects within the group and their digital realization. Descriptions are supplemented with screenshots of models in Simulink program tool. Each model was tested with a sound file and evaluated afterwards. Graphs of signals before and after the simulation are given. Simulink program tool enables the user to simulate and design dynamic systems. It also generates programming code of those systems. This code can be transferred to embedded hardware systems and thus verified in real time. Methods of work and problems with the usage of Simulink program tool are presented. The described groups of effects are octaving, distortion, delay, modulation and filtering. The Chapter on octaving effects contains theoretical background of octave phenomenon, along with some electronic realizations which significantly influenced digital ones. Distortion effects primarily imitate the sound of overdriven audio amplifiers, so various simulations of that specific sound are emphasized. Different values of delay parameter were analyzed in the group of delay effects, with accentuation on the impact of that value on hearing the result in time and frequency domains. The group of modulation effects includes the descriptions of most commonly used effects, while the chapter on filtering contains the explanation of how some less used electronic realization of filtering is digitally simulated. At the end of this thesis a list of options for further analysis of this topic is added.

Keywords:

Musical effects, octave, distortion, delay, modulation, filtering, Simulink.

1 Uvod

To diplomsko delo je nastalo kot razširitev moje seminarske naloge z naslovom Harmonični množilnik pri prof. Dušanu Kodeku. V seminarski nalogi sem se ukvarjal z načrtovanjem in realizacijo vezja, ki vhodnemu signalu prišteje oktavo višji signal, vsoto dveh pa pošlje na izhod. Gre namreč za enega od učinkov pri ustvarjanju glasbe in je navadno v obliki pedala. Tega uporabnik priklopi med električni inštrument in ojačevalec, pedal pa sestavljajo še stikalo za vklop in izklop ter nekaj potenciometrov za uravnavanje npr. količine za oktavo višjega signala in glasnosti izhodnega signala v primerjavi z vhodnim. Prof. Kodek je predlagal, naj nalogo poskusim narediti še po digitalni poti, nato pa rezultat preizkusim na procesorjih DSP podjetja Texas Instruments (v nadaljevanju TI), ki so na voljo v laboratoriju za arhitekturo in procesiranje signalov.

Na predlog asistenta dr. Rozmana je prva simulacija sistema postavljena v grafično-dinamičnem okolju za modeliranje in simuliranje sistemov Scicos [8], ki deluje kot dodatek programa Scilab. V primeru uspešne realizacije bi lahko enako storil še s programom Simulink [9] in nato raziskal še možnost prenosa ustvarjenih modelov oz. programske kode na razvojne ploščice podjetja TI.

Ker sem analogno realizacijo seminarske naloge zelo hitro prevedel v digitalno obliko, sem v dogovoru s prof. Kodekom diplomsko nalogo razširil v raziskovanje in poskus realizacije več različnih zvočnih učinkov, ki so uporabni za procesiranje zvoka električnih inštrumentov oz. vokala. Na koncu sem se odločil za opis in postavitev modelov petih družin zvočnih učinkov, ki so urejeni glede na sorodstvo po načinu realizacije [11]. Tako so nekateri učinki v isti družini, ker gre v bistvu za enake učinke, do katerih pridemo na različne načine. Drugi učinki pa so slišno različni, čeprav jih ustvarjamo na enak način, spreminjamo le parametre.

Spodaj je naštetih teh pet družin.

- Učinki oktaviranja
  - Namen: doseči enak učinek na različne načine, poskus digitalne realizacije analognih rešitev.
- Učinki distorzije oz. popačenja
  - Namen: doseči enak učinek na različne načine, poskus digitalne realizacije analognih rešitev.
- Učinki zakasnitve
  - Namen: doseči različne učinke s spreminjanjem parametrov, raziskati njihov vpliv na končni izid.
- Učinki moduliranja
  - Namen: doseči različne učinke s spreminjanjem parametrov, raziskati njihov vpliv na končni izid.
- Učinki filtriranja
  - Namen: doseči enak učinek na različne načine, poskus digitalne realizacije analognih rešitev.

V laboratoriju sem si sposodil razvojno ploščico TMS320VC5505, emulator JTAG XDS100 podjetja Spectrum Digital in razvojno okolje Code Composer Studio (v nadaljevanju CCS) podjetja TI. CCS omogoča ustvarjanje in prenos programske kode za DSP procesorje, mikrokontrolerje in procesorje za posebne namene. Vključuje orodja za razvoj in odpravljanje napak, prevajalnike, simulatorje, urejevalnik izvorne kode in še več. Dodatek Embedded Link

znotraj delovnega okolja Matlab pa naj bi omogočal povezavo Simulinka s CCS-jem, kar v končni fazi pomeni povezavo med modelom in resničnim izdelkom.

Arvind Ananthan je projektni vodja podjetja MathWorks in vodi oddelek, ki povezuje TI orodja in procesorje z izdelki MathWorks. Žal je na internetnem forumu podjetja TI potrdil, da orodja MathWorks ne delujejo z različico CCS 4.0, temveč le do različice 3.3. Težava je še toliko večja, ker emulator XDS100 za razvojno ploščico TI C55xx deluje le na različici CCS 4.0 [10]. To pomeni, da čakamo na različico okolja CCS, ki deluje z Matlabom, oz. emulator XDS100, delujoč s starejšo različico CSS. Zato s strojno opremo, ki je na voljo v Laboratoriju za arhitekturo in procesiranje signalov na FRI, med izdelavo te diplomske naloge žal ni bilo mogoče preizkusiti narejenih modelov v realnem času. Toda vsak model je preizkušen s simulacijo z zvočno datoteko, zato si upam trditi, da bodo modeli, ki so uspešno prestali ta preizkus, dobro delovali tudi v realnem času. Kot izvor in ponor se bosta namreč namesto blokov, ki bereta in zapisujeta zvočne datoteke, pojavila bloka, ki predstavljata (že inicializirana) vhod in izhod razvojne ploščice.

2 Opis programske opreme in delovnega okolja

2.1 Programska oprema Scilab in Scicos

Okolje Scicos med drugim omogoča [8]:

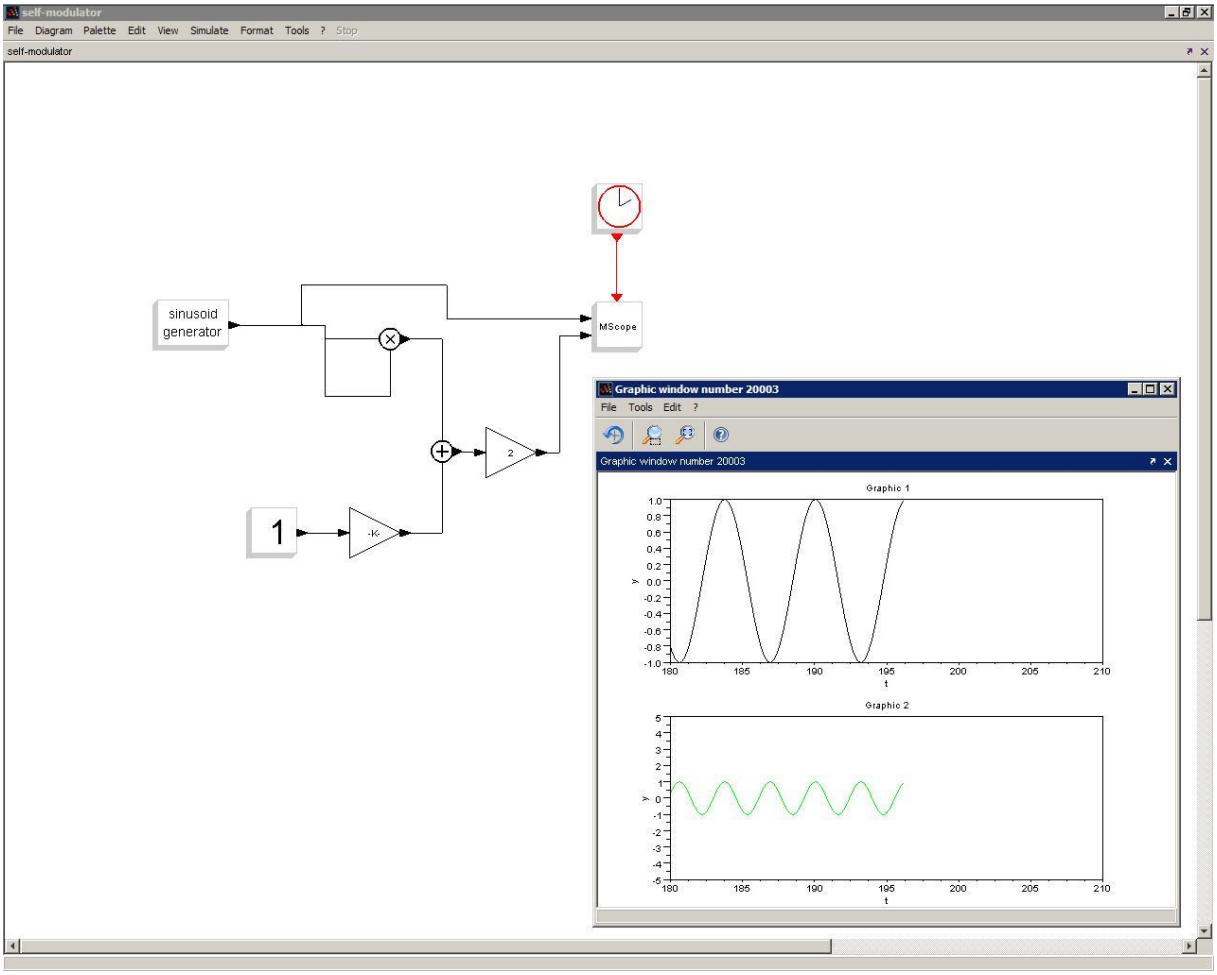
- modeliranje, prevajanje in simuliranje dinamičnih sistemov;
- združevanje zveznega in diskretnega delovanja znotraj istega modela;
- izbiro prednastavljenih standardnih blokov;
- izbiro prednastavljenih potekov inicializacije strojne opreme (natančen in posodobljen seznam podprte opreme je na [8]);
- programiranje novih blokov v programskih jezikih C, Fortran in Scilab;
- generiranje programske kode C z uporabo generatorja kode.

Delo v takem okolju poteka približno tako:

- ustvarimo nov model, ki je videti kot prazna stran;
- iz različnih palet blokov izbiramo želene bloke, te pa med seboj povezujemo;
- po potrebi iz standardnih blokov ustvarjamo nove, kar povečuje preglednost in lajša delo v prihodnje (podobno kot ustvarjanje knjižnic pri programiranju);
- standardne in lastne bloke lahko ponovno uporabimo pri izdelavi novih blokov (podobno kot gnezdenje zank pri programiranju);
- izmed različnih palet blokov želim posebno poudariti paleti izvorov in ponorov (*sources* in *sinks*), ki sta ključnega pomena za uspešno modeliranje in simulacijo – vsebujeta namreč različne prilagodljive generatorje signalov ter razširljive grafe, na katerih si lahko ogledamo potek signala (omenjena sta bila meni najbolj uporabna bloka);
- vsak blok ima različne lastnosti, ki jih lahko prilagajamo, posebno pozorni pa moramo biti pri povezovanju, saj ni mogoče povezati vsakega z vsakim;
- s spreminjanjem lastnosti blokov lahko med ostalim prilagajamo tudi vhode oz. izhode blokov, zato lahko povežemo tudi bloke, ki s privzetimi nastavitvami niso povezljivi. Vseeno pa je včasih potrebno uporabiti dodatne bloke, ki prevedejo obliko povezave oz. podatkov (npr. izvor daje t. i. *frame-based* podatke, ponor pa pričakuje *sample-based*).

Program je (tako kot Scilab) na voljo brezplačno, poleg izvornih ustvarjalcev ga posodabljajo še navdušenci iz vseh koncev sveta. Dobra stran je vsekakor cena in dostopnost programske opreme, slaba stran pa težave pri uporabi – pri trenutni različici Scilaba (5.1.1) se pojavljajo različne hibe v zvezi s stabilnostjo in odzivnostjo delovnega okolja. Zaradi prostovoljnih dodatkov in posodobitev gre večinoma za strogo specializirane potrebe posameznikov, zato z izjemo zelo splošne uporabe na kaj več ne moremo upati, razen, če sami ne ustvarimo potrebno. Toda v tem primeru nismo več osredotočeni na reševanje naloge (kaj), temveč ustvarjamo oz. izpopolnjujemo orodje (kako).

Vseeno se je Scicos po začetnem uvajanju izkazal kot zelo uporaben, saj so prvi modeli in simulacije nastali prav v njem. Tudi pri delu z njim je kmalu postalo jasno, da je potrebno začetno nalogo (digitalno realizacijo oktaviranja) razširiti, saj je bila realizirana v zelo kratkem času: skupaj z vsemi prilagajanji blokov in preverjanji nastalega modela je izdelava naloge trajala dobri dve uri, zahtevala pa je manj kot deset gradnikov, med katere sta šteta tudi generator signala ter prikaz izvornega in obdelanega signala v grafu. Generirana programska koda je znašala nekaj več kot tisoč tristo vrstic programskega jezika C.



Slika 1: Model in graf simulacije v programu Scicos

Na sliki vidimo generator sinusnega signala kot izvor, časovno prožen blok, ki izrisuje graf, bloka za množenje in seštevanje, konstanto ter dva bloka za ojačenje. Levi je uporabljen za spremembo predznaka in vrednosti konstante (–0.5), desni pa rezultat vsote pomnoži z dve. Zagon simulacije prikaže okno grafa, v katerem se v resničnem času izrisujeta dva diagrama: prvi je izvorni signal, drugi pa obdelani. Oba signala imata amplitudi numeričnih vrednosti med –1 in 1.

Na tem mestu želim poudariti, da takšen način ustvarjanja zelo pospeši in olajša delo (ustvarjanje in razhroščevanje). Med učnim materialom podjetja TI je zaslediti podatek, da je skupina strokovnjakov pri izdelavi programske opreme za nadzorni stolp letališča namesto ocenjenih šestih tednov »ročnega« programiranja porabila tri dni dela s Simulinkom (pri oceni je vštet povprečen čas iskanja in odpravljanja napak). Po pravici povedano, bi veliko hitreje »na roke« napisal program, ki izvede operacijo, enakovredno operaciji na zgornji sliki – od omenjenih tisoč tristo vrstic jih dober del odpade na ustvarjanje slike, prostorsko razmestitev elementov, njihove lastnosti in povezovalne zmogljivosti. Vendar pa se pri večjih projektih in povezljivostjo s strojno opremo to razmerje kmalu obrne.

2.2 Programska oprema Matlab in Simulink

Plačljiva vzporednica Scilabu je Matlab, znotraj katerega deluje Simulink, po katerem so se zgledovali pri ustvarjanju Scicosa. Simulink seveda vsebuje nemerljivo več elementov,

uporabniški vmesnik in datoteke za pomoč uporabniku so neprimerljivo bolj dodelane in uporabne, o stabilnosti sistema in podprtosti različne strojne opreme pa tudi ne gre izgubljati besed. Seveda to delovno okolje ni uporabno za vse izzive v računalništvu, menim pa, da je za specifično uporabo zelo dobro načrtovano in izdelano, saj je veliko pozornosti namenjene povratnim informacijam uporabnikov.

Ravno zaradi boljše podpore uporabnikom, stabilnega delovanja, večjega razpona standardnih blokov in večje prilagodljivosti ter možnosti, da bi ustvarjene modele prenesel na enega od procesorjev za obdelavo digitalnih signalov podjetja TI in tako preizkusil delovanje v resničnem času, sem z ustvarjanjem modelov in simulacij nadaljeval znotraj delovnega okolja Matlab. Simulink je okolje za simuliranje in snovanje modelov dinamičnih sistemov, ponuja pa grafični vmesnik in prilagodljivo množico knjižnic blokov, s katerimi lahko načrtujemo, simuliramo, implementiramo in preizkušamo različne časovno spremenljive sisteme, vključno s komunikacijami, nadzorom, procesiranjem signalov in videa ter slikovnega gradiva [9].

Delo v Simulinku poteka enako kot pri Scicosu, le da je delovno okolje bolj odzivno in stabilno, knjižnice blokov večje, datoteke pomoči pa obširnejše. Temu primerno so tudi sistemske zahteve večje. Že sama namestitev Matlaba z vsemi dodatki zahteva 4,22 GB prostora na disku (v primerjavi s Scilabom in njegovimi pribl. 100 MB), traja pa približno eno uro. Podobno je s CCS, kjer se namestitev ustvarja in prevaja na kraju samem, zato je računalnik dobre pol ure neuporaben za kaj drugega. Med delovanjem Matlaba pa ni nobenih težav, v operacijskem sistemu Windows XP (SP3 in popravki do decembra 2009) in osebnem računalniku s procesorjem Intel P4 2,6 GHz in 1 GB delovnega pomnilnika brez zastojev lahko hkrati uporabljamo Matlab, Simulink, internetni brskalnik Firefox, urejevalnik besedila MS Word, predvajalnik Winamp ter še kaj.

Pri vseh ustvarjenih modelih je privzeto, da imajo signali numerično vrednost med –1 in 1. Vsi modeli so najprej preizkušeni s sinusnim generatorjem kot izvorom signala ter grafom za prikaz kot ponorom signala. Tudi to mi je bilo v večini primerov izjemno v pomoč, saj je za nekatere učinke znano, kakšni so »videti«. Dosti napak v načrtovanju sem odpravil ravno na ta način. Vseeno pa končnih nastavljanj parametrov ne moremo izvesti brez zvočnih datotek, saj je zvok končno merilo rezultata.

2.3 Programska in strojna oprema za potrebe zajemanja in predvajanja zvoka

Zvočne vzorce sem posnel na lastnem osebnem računalniku, opremljenem z zvočno kartico Creative Audigy Platinum, ki ima vhod, posebej namenjen inštrumentom. Snemal sem s programsko opremo Cubase SX3, vsaki datoteki pa sem odstranil enosmerno komponento in jo normaliziral, tj. ji maksimiziral amplitudo. Zvočni zapis je t. i. CD-kvalitete, torej 44,1 kHz vzorčenje in 16-bitni zapis vzorca.

Inštrumenti, ki sem jih uporabil pri snemanju so:

- električna kitara Yamaha RGX 121D,
- električni bas Ernie Ball MusicMan StingRay5,
- električni pianino Casio Privia PX-310.

2.4 Frekvenčni razpon električnega basa in kitare

Čeprav končni izdelek (torej tisto, kar slišimo preko medijev) velikokrat ne kaže tako, ima omenjeni inštrument v bistvu poln frekvenčni razpon. Privzemimo, da je sistem za zajem, prenos in reprodukcijo tona idealen: v takem primeru lahko z različnimi tehnikami igranja dosežemo od najniže uglašene strune B0 pri 30,87 Hz (za 5 oz. 6-strunski bas; pri standardnih 4-strunskih je to E1 pri 41,20 Hz), prek umetnih alikvotov, do udarcev kovine ob kovino v področju slišne meje pri 20 kHz. Če se omejimo le na razpon temeljnih frekvenc, znaša ta pri električnem basu od B0 pri 30,87 Hz do G4 pri 392,00 Hz – toliko namreč omogoča ubiralka z dvema oktavama oz. štiriindvajsetimi polji. Za električno kitaro sta ta dva podatka od E2 pri 82,41 Hz do E6 pri 1318,52 Hz [3].

3 Učinki oktaviranja

3.1 Dodajanje harmonikov in subharmonikov

Ta učinek se izvaja z enostavnimi analognimi oz. digitalnimi napravami, katerih naloga je ustvarjanje tona eno ali dve oktavi nad in/ali pod določenim tonom. Uporablja se večinoma na posameznih notah (torej ne na akordih, tj. več sočasnih notah), napravi pa pravimo *octaver*. Pogosto se uporablja pri solističnih oz. basovskih inštrumentih, ker jim ojači temelj in/ali prezenco med ostalimi inštrumenti (kitara, bas, tenorski saksofon, trobenta). Delovanje tega učinka omogoča odnos 1 proti 2 med frekvencami glasbenih not [5].

3.2 Definicija oktave in teoretično ozadje

V glasbi je oktava interval med enim tonom in drugim tonom, ki ima dvojno oz. polovično frekvenco prvega. Oktavni odnos je naravni fenomen, ki mu pravijo tudi *osnovni čudež glasbe*, njihova uporaba pa je vsakdanja v večini glasbenih sistemov. Izpeljemo ga lahko iz vrste harmonikov (tj. večkratnikov temeljne frekvence), kot interval med prvim in drugim harmonikom. Oktavi včasih pravimo tudi obseg, *diapazon* (iz grščine: čez vse) [5].

- Primer: če ima nota frekvenco 400 Hz, je za oktavo višja nota pri 800 Hz, za oktavo nižja pa 200 Hz. Razmerje med frekvencama dveh tonov, ki sta oktavo narazen, je torej 1:2 oz. 2:1. Naslednje oktave se pojavijo pri 2^n, pomnoženo s frekvenco note (pri čemer je n celo število). 50 Hz in 400 Hz sta eno oz. dve oktavi stran od 100 Hz, ker sta enaki ½ (oz. 2^{-1}) in 4 (oz. 2^2) pomnoženo s frekvenco note. Toda 300 Hz ni celo število oktav nad 100 Hz, čeprav je večkratnik frekvence 100 Hz [5].

Interval opisuje razdaljo med dvema notama. Za primo oz. unisom (zaigrani oz. zapeti noti sta isti) je interval oktava najenostavnejši interval v glasbi. Zaigrani oz. zapeti noti sta torej oktavo narazen. Kljub temu ljudsko uho sliši obe noti kot isto noto, zaradi strogo povezanih harmonikov, ki sestavljajo posamezno noto. To je tudi razlog, da v zahodnjaškem sistemu glasbene notacije enako poimenujemo note, ki so oktavo narazen – ton A, ki je oktavo višji, je tudi A [5].

- Za razlikovanje med njimi so oktave oštevilčene, primer pa je standardni klavir, katerega razpon se razteza od A0 do C8.

Oktavna ekvivalenca je privzetek, da so toni eno ali več oktav narazen glasbeno ekvivalentni na več načinov in da so lestvice enovito definirane z določanjem intervalov znotraj oktave. To pomeni, da bo neka lestvica prepoznavna po svoji kvaliteti, ne glede na to, v kateri oktavi oz. kje v razponu inštrumenta jo zaigramo [5].

Ta privzetek je del večine "naprednih glasbenih kultur", ne pa tudi "primitivnih" ter zgodnjih pojavov glasbenega ustvarjanja. Podoben je *enharmonični ekvivalenci*, nekaj manj *transpozicijski ekvivalenci*, še manj pa *inverzni ekvivalenci*, ki se v splošnem uporablja le v kontrapunktu in teorijah glasbene množice oz. atonalne glasbe. Koncept, po katerem ima ton dve dimenziji (višina tona – absolutna frekvenca; razred tona – relativen položaj znotraj oktave), sam po sebi vključuje krožnost oz. cikličnost oktave. Tako so npr. vsi toni C♯ v kateri koli oktavi del *istega razreda* tonov, čeprav imajo različne višine, tj. *različne absolutne frekvence* [5].

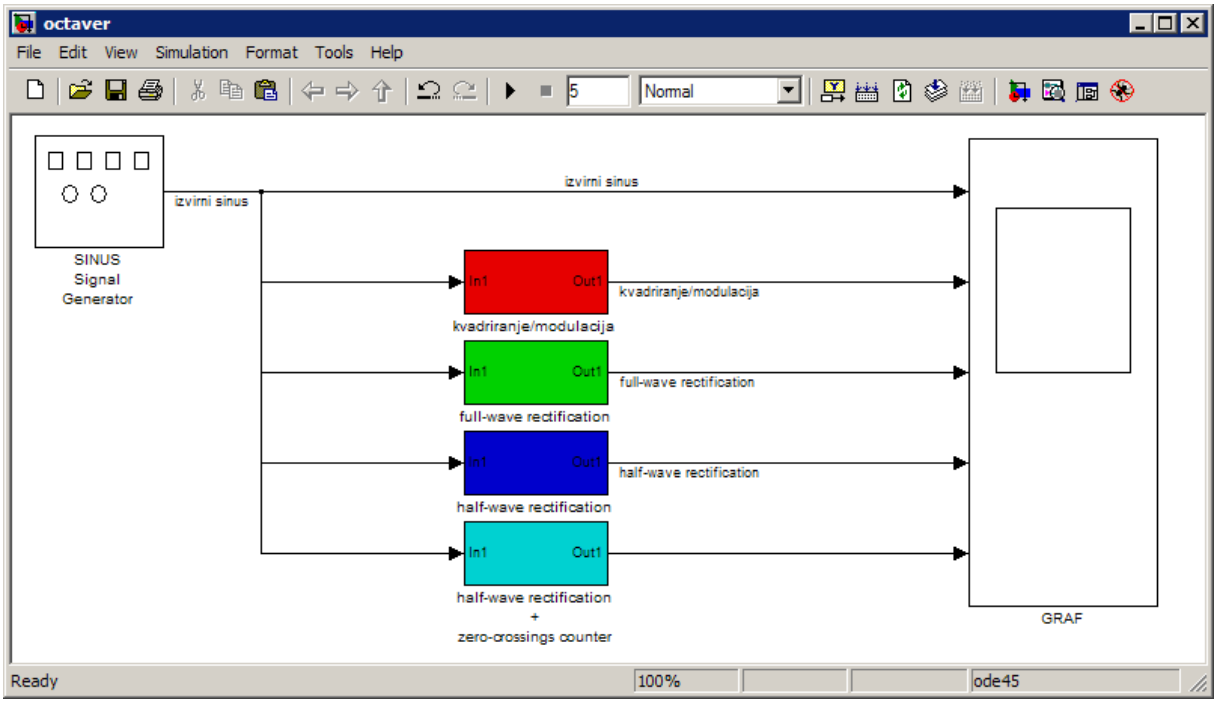
- Oktavno ekvivalenco poleg ljudi doživljajo tudi opice, biološka podlaga pa je očitno oktavno mapiranje nevronov v slišnem talamusu možganov sesalca. Zaznavanje oktavne ekvivalence v samoorganizirajočih nevronskih omrežjih se lahko brez poučevanja ustvari zaradi izpostavljenosti uglašenim tonom, torej poslušanju oktav in drugih intervalov. Razlog za to je akustična struktura teh not.
- Raziskave so pokazale doživljanje oktavne ekvivalence tudi pri podganah (Blackwell in Schlosberg, 1943), dojenčkih (Demany in Armand, 1984) in glasbenikih (Allen, 1967) in kot tudi *odsotnost* doživetja pri škorcih (Cynx, 1993), 4–9 let starih otrocih (Sergeant, 1983) ter ljudeh, ki nimajo glasbene izobrazbe [5].

Poleg opisovanja odnosa med dvema tonoma se beseda oktava uporablja tudi kot oznaka za niz not, ki se nahajajo med parom dveh not, oddaljenih za oktavo. V diatoničnih lestvici (in ostalih standardnih heptatoničnih lestvicah zahodnjaške glasbe) je sedem not – če pa štejemo še oktavo, dobimo osem not, zato ime oktava (iz latinščine: *octavus*, ta iz *octo*, kar pomeni osem). Druge lestvice imajo lahko različno število not, ki pokrivajo obseg oktave (torej število not med dvema oktavo oddaljenima notama), npr. kromatična lestvica (12 not), klasična arabska lestvica (17, 19, celo 24 not), toda vseeno uporabljamo besedo oktava [5].

Pri igranju inštrumenta lahko oktava pomeni tudi poseben učinek – sočasno igranje dveh not, ki sta oddaljeni za eno oktavo. Ta učinek lahko glasbenik doseže z inštrumentom oz. s posebnimi dodatki. Nekateri inštrumenti so namenoma uglašeni oz. celo ustvarjeni v ta namen, npr. dvanajststrunska kitara oz. oktavne orglice [5].

V večini primerov zahodnjaške glasbe je oktava razdeljena na 12 poltonov, ki so med seboj enako oddaljeni (po metodi ekvivalentnega temperiranja – *twelve tone equal temperament*). Klavir ima 88 tipk – poltonov, kar da skupno $7\frac{1}{3}$ oktav [2, 5, 7].

Sledi opis izvedbe modela različnih implementacij oktaviranja v Simulinku. Na naslednji sliki vidimo sistem znotraj okolja Simulink. Ta sistem vsebuje generator sinusnega signala kot izvor, kot ponor pa je določen gradnik, ki izriše diagrame. V tem primeru se torej izriše izvirni sinusni signal in še štirje obdelani signali. Za vsakega od načinov oktaviranja je izdelan nov blok, ki ga dvakrat kliknemo, da se odpre. Graf učinkov je naveden na koncu tega poglavja.

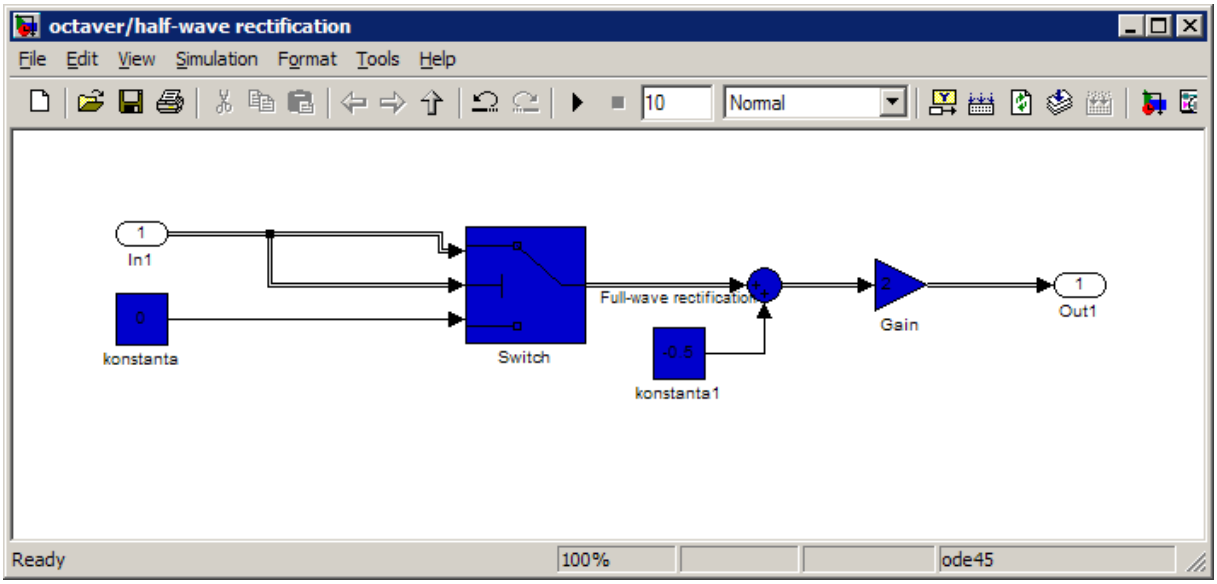


Slika 2: Sistem učinkov oktaviranja v Simulinku

3.3 Polvalno usmerjanje (half-wave rectification)

Ta učinek dosežemo tako, da ohranimo pozitivne vrednosti vhodnega signala, negativne pa postavimo na ničlo [6, 11]. Kot je razvidno iz slike, opisani postopek dosežemo tako, da na zgornji in spodnji vhod stikala povežemo izvorni signal in ničelno konstanto. Na sredinski vhod stikala povežemo signal, s katerim krmilimo izbiro zgornjega oz. spodnjega vhoda. V tem primeru je to tudi izvorni signal, stikalo pa je nastavljeno tako, da na izhod usmeri zgornji vhod, če je izvorni signal pozitiven oz. enak nič. V nasprotnem primeru na izhod usmeri spodnji vhod. Prišteta konstanta –0,5 centrira signal, množenje z 2 pa ga raztegne na maksimalen razpon amplitude.

Ta operacija ima za posledico ustvarjanje sodih večkratnikov osnovne frekvence, ki jih v izvornem signalu ni bilo. Zato bo v izhodnem signalu slišati popačenje.

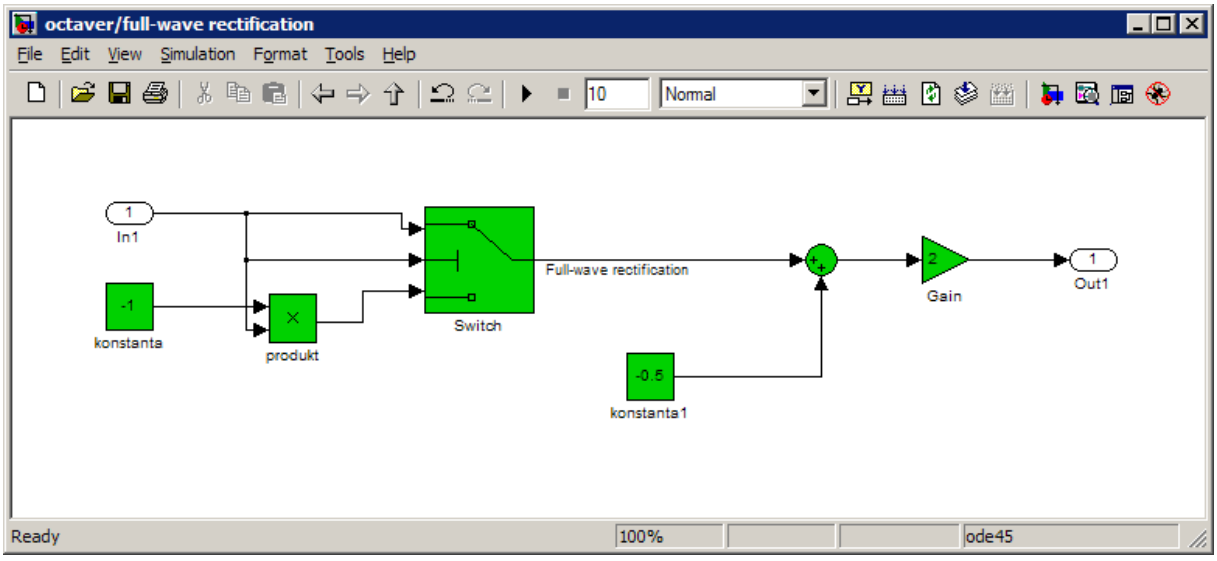


Slika 3: Model polvalnega usmerjanja v Simulinku

Čeprav je simulacija s sinusnim generatorjem potekala po pričakovanjih, je preizkus z zvočno datoteko razočaral. Sicer sem že pred tem vedel, da bo zvok popačen zaradi ostrih sprememb smeri krivulje (točke nezveznosti), vendar pa je popačenje takšno, da je učinek povsem neuporaben. Tudi v kombinaciji z distorzijami ne pride v poštev, saj je popačenje povsem nemuzikalično – je namreč neodvisno od glasnosti signala.

3.4 Polnovalno usmerjanje (*full-wave rectification*)

Tudi tukaj znova ohranimo pozitivne vrednosti, negativnim pa spremenimo predznak. Ta operacija ustvarja sode večkratnike osnovne frekvence, ki pa v novem signalu ni več prisotna [6, 11]. Podobno kot pri polvalnem usmerjanju je tudi tukaj stikalo, ki preklaplja glede na pozitivno oz. negativno vrednost signala. Razlika je v tem, da je spodnji vhod stikala tokrat vhodni signal, pomnožen z –1, torej dobimo ravno obratni signal. Ker pa stikalo preklopi na spodnji vhod, če je signal negativen, dobimo na izhodu same pozitivne vrednosti. Takemu izhodu je zopet potrebno dodati odmik –0,5 ter ga raztegniti na maksimalen razpon amplitude.



Slika 4: Model polnovalnega usmerjanja v Simulinku

Podobno kot pri polvalnem usmerjanju sem tudi tukaj pričakoval popačenje, ki pa se je žal zopet izkazalo kot neuporabno za glasbene namene. Na različnih internetnih forumih, kjer se srečujejo navdušenci nad glasbenimi učinki in njihovo samostojno gradnjo, je mogoče najti nekaj razprav o elektronskih izvedbah (torej takšnih s pasivnimi elektronskimi elementi) pol- in polnovalnega usmerjanja v namene oktaviranja. Vsi so izkusili različne stopnje popačenja, ki ga pa poskušajo omiliti na različne načine – večina jih vključuje mešanje z izvornim signalom in dodajanje distorzije, ki prikrije popačenje čistega signala.

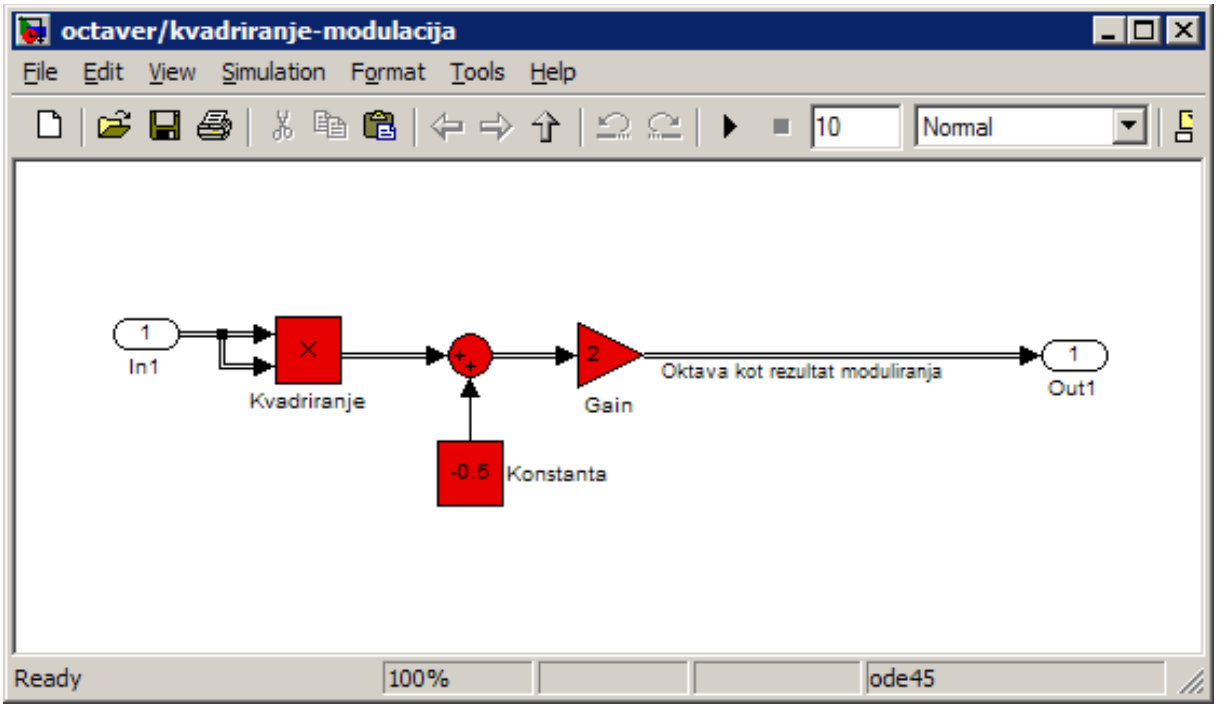
3.5 Oktavno moduliranje

Drugačna izvedba pa si sposodi idejo moduliranja: vhodni signal množi s samim sabo, rezultat pa je signal z dvakrat višjo frekvenco. Dokaz je enostavna trigonometrična formula:

$$\cos(f_1) * \cos(f_2) = \frac{1}{2}\cos(f_1 + f_2) + \frac{1}{2}\cos(f_1 - f_2). \qquad (1)$$

V primeru, da signal vsebuje samo eno frekvenco, postane drugi člen desne strani formule (1) konstanta z vrednostjo 0,5, kar zlahka odpravimo tako, da od desne strani odštejemo 0,5 in jo nato pomnožimo z 2, da dobimo maksimalno amplitudo. Če vhodni signal ni idealen sinusni signal (torej tak brez primesi), potem rezultat ni več tako idealen in dobimo učinek obročnega modulatorja (*ring modulator*). Njegov opis je naveden v poglavju 6.2.

Na sliki torej vidimo vhodni signal, ki se množi s samim sabo. Rezultatu odštejemo konstanto –0,5, izid pa z elementom ojačenja pomnožimo z dve.



Slika 5: Model oktavnega moduliranja v Simulinku

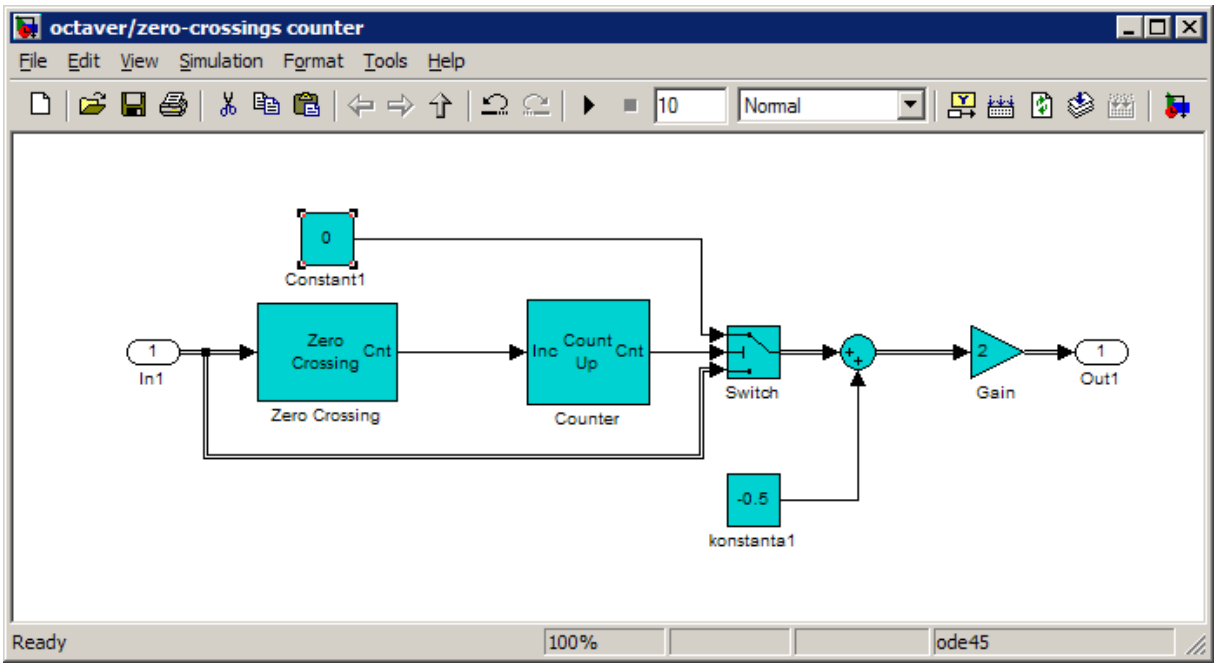
Za razliko od ostalih ta ni razočaral pri preizkusu z zvočno datoteko. Vedenje učinka je skoraj identično trdo ožičeni realizaciji iz seminarske naloge Harmonični množilnik: posamezne note se preslikajo oktavo višje, pri akordih oz. več sočasnih notah pa dobimo popačen zmazek (mimogrede, podobno se vedejo tudi vse ostale tovrstne naprave). Učinek obročnega modulatorja pride do izraza šele, če hkrati zaigrani noti nista oktavna večkratnika (torej npr.

frekvenci 100 Hz in 300 Hz). Digitalna različica je pri posameznih notah stabilnejša, kar lahko pripisujemo matematično eksaktnemu postopku izračuna, za razliko od odstopanj pri elementih LRC.

Poudariti je še treba, da je učinek na zvočni datoteki morda manj slišen, ker signal vsakega inštrumenta poleg temeljne frekvence sestavljajo še večkratniki te frekvence. Rezultat je v bistvu začetni signal brez temeljne frekvence, saj se vse dvigne oktavo višje – slišati je, kot bi imeli skoraj identičen signal, ki smo ga obdelali z visokoprepustnim filtrom, s katerim smo odrezali temeljno frekvenco. Vendar se v pravilnost delovanja posameznih naštetih rešitev lahko prepričamo tako, da si ogledamo graf simulacije.

3.6 Štetje prečkanj ničle

Če štetje prečkanj ničle uporabimo na pol- oz. polnovalno izenačenem signalu, lahko na nič postavimo vnaprej določeno število pozitivnih valov. Tak signal ima temeljno frekvenco za oktavo nižje od vhodnega signala [11].



Slika 6: Model štetja ničel v Simulinku

Vhodnemu signalu štejemo prečkanje ničle z dvema blokoma: prvi ima za izhod vrednost 1, dokler je vhod enak nič, sicer je izhod nič. Drugi blok je števec, ki je nastavljen tako, da se proži ob pozitivni fronti vhoda. Števec se ponastavi, ko doseže vrednost 3 – tako je simulirano štetje prečkanj ničel in deljenje rezultata po modulu 4. Stikalo prepušča zgornjo vejo (konstanto enako nič), če je števec enak 1 ali več. Izhodni signal stikala zopet maksimiramo, tako kot v prejšnjih primerih. Rezultat je tak, kot bi pri polvalnem usmerjanju izničili vsak drugi val.

Kljub uspešnemu testiranju s sinusnim generatorjem preizkus z zvočno datoteko ni bil zadovoljiv. Datoteko je sicer možno predvajati, vendar vsebuje nepredvidljive prekinitve in je zato neuporabna.

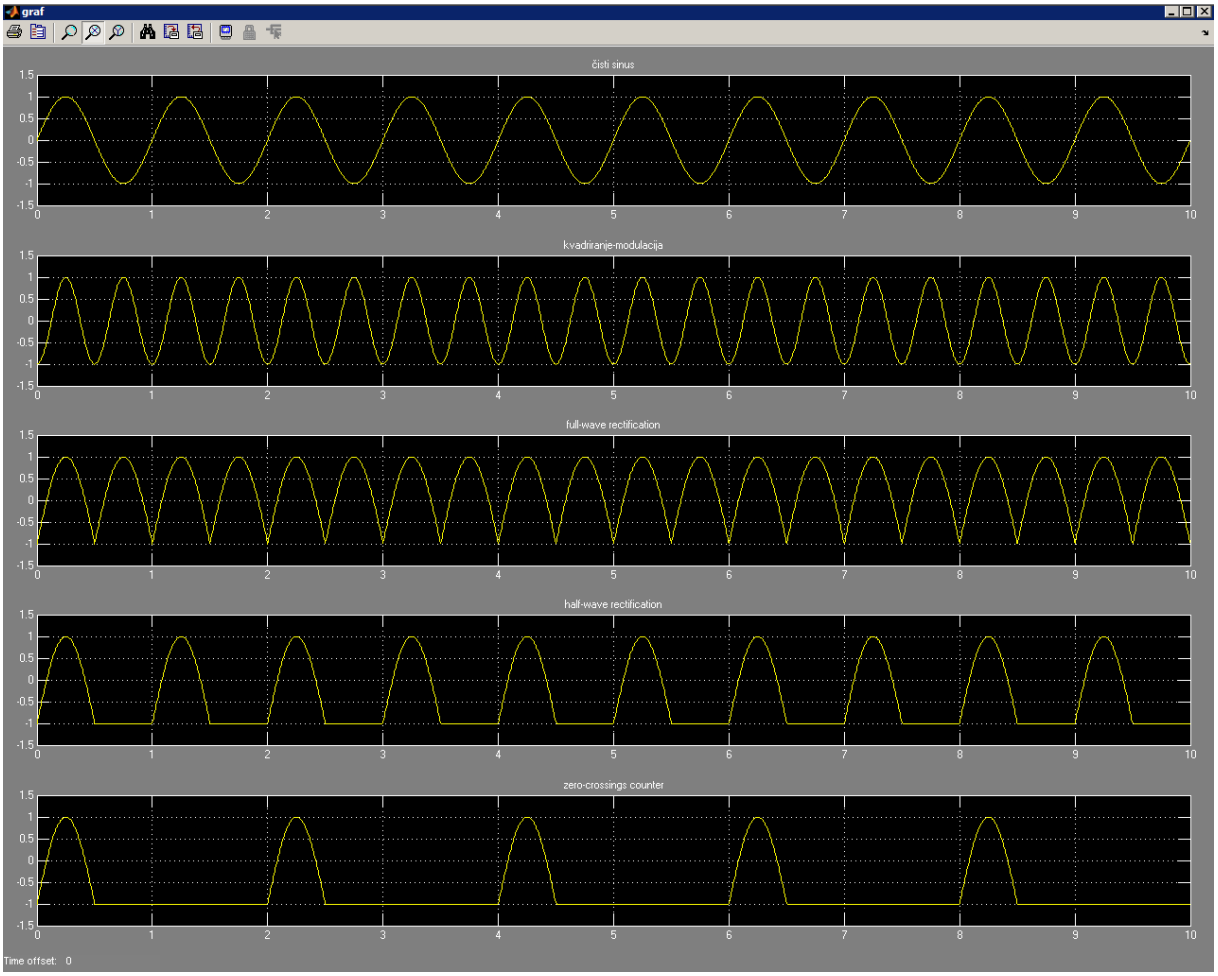
3.7 Graf družine učinkov oktaviranja

Za konec opisa družine *octaverjev* želim prikazati še graf, ki je rezultat simulacije prej opisanih učinkov. Prvi diagram prikazuje neobdelani sinusni signal, ostali diagrami pa prikazujejo sinusni signal po prehodu skozi opisane modele oktaviranja. Trajanje simulacije je 10 sekund, generator signala pa ustvarja sinus s frekvenco 1 Hz, amplitude od –1 do 1. Tako nizko frekvenco sem izbral zaradi jasnejšega grafa. Kakšna bolj realna frekvenca bi zahtevala tudi višje vzorčenje grafa (v tem primeru znaša vzorčenje 1 kHz), to bi pa pomenilo tudi podaljšanje trajanja simulacije.

Drugi diagram je rezultat modulacije s kvadriranjem in odlično prikazuje dvakrat višjo frekvenco od vhodne.

Tretji in četrti diagram sta izida polno- in polvalnega usmerjanja. Pri obeh je razviden rezultat operacije nad negativnim delom signala: v prvem primeru negativnem delu spremenimo predznak, v drugem primeru pa ga izničimo.

Zadnji diagram je rezultat štetja ničel. Če ga primerjamo z diagramom polvalnega usmerjanja, je razvidno, da je poleg negativnih delov signala izničen tudi vsak sodi pozitivni del.



Graf 1: Graf učinkov oktaviranja v Simulinku

4 Učinek distorzije oz. popačenja

4.1 Distorzija in nelinearno procesiranje

Zvočni učinki algoritmov za dinamično procesiranje, simulacije elektronk, preobremenitev ojačevalnih stopenj in distorzij za kitaro spadajo v kategorijo nelinearnega procesiranja. (Ne)namerno ustvarjajo (ne)harmonične frekvenčne komponente, ki niso prisotne v izvornem signalu (neharmonične zato, ker niso nujno v harmoničnem sorodstvu z vhodnim signalom in ustvarjajo disonance). Harmonično popačenje povzročajo nelinearnosti znotraj naprave. Večino teh naprav nadzorujemo tako, da spreminjamo parametre in sočasno poslušamo izhodni signal. Za doseganje rezultatov, ki so všečni širši množici poslušalcev, je potrebno veliko poslušanja ter izkušenj pri snemanju. Uporaba teh naprav je umetnost zase in ena od glavnih orodij snemalcev in izvajalcev glasbe.

Nelinearno procesiranje je naziv za algoritme oz. naprave, ki na nelinearen način procesirajo signale v analogni oz. digitalni domeni [11]. Za sinusni vhod z znano frekvenco f_1, periodo vzorčenja T in amplitudo A

$$x(n) = A\sin(2\pi f_1 Tn), \qquad (2)$$

je izhod linearnega sistema enak

$$y(n) = A_{izh}\sin(2\pi f_1 Tn + \varphi_{izh}), \qquad (3),$$

kar je spet sinusna funkcija, z amplitudo A_{izh} in faznim zamikom φ_{izh}. Izhod nelinearnega sistema pa je vsota sinusoid z amplitudami od A_0 do A_N

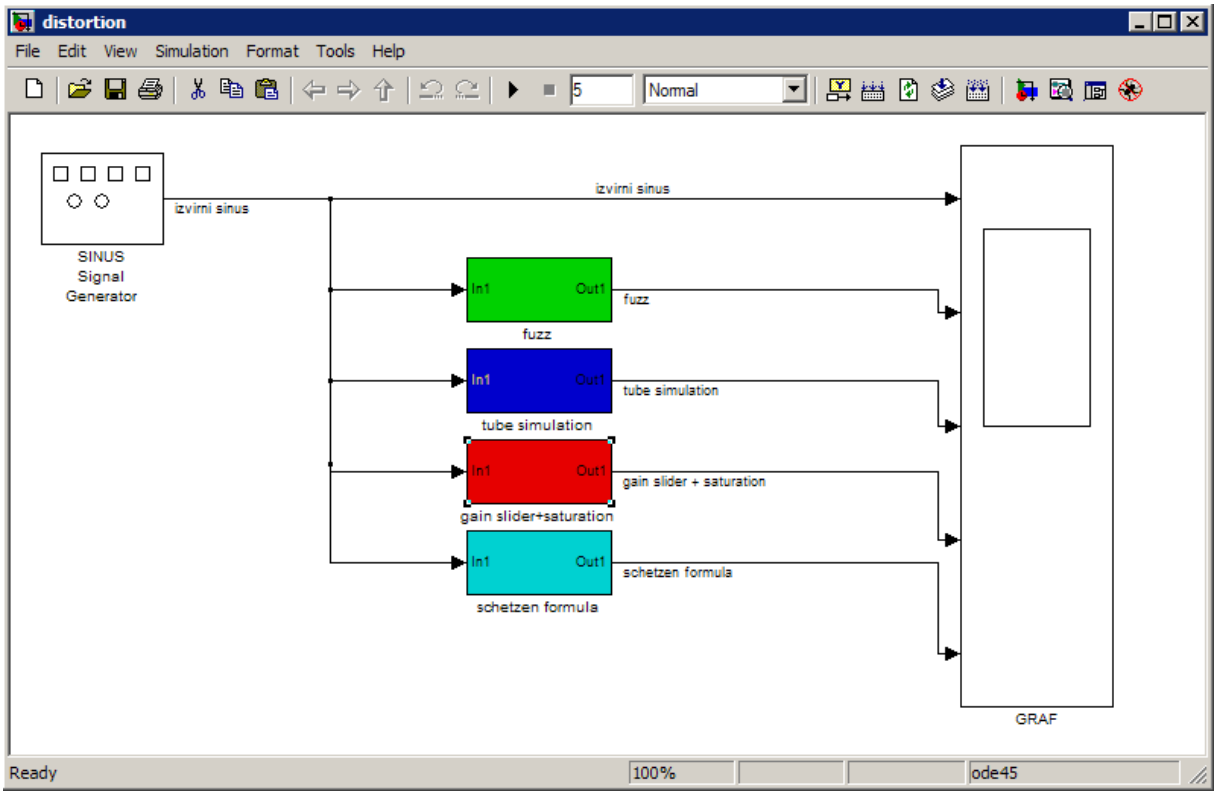
$$y(n) = A_0 + A_1\sin(2\pi f_1 Tn) + A_2\sin(2 * 2\pi f_1 Tn) + \cdots + A_N\sin(N * 2\pi f_1 Tn). \quad (4)$$

Digitalno procesiranje je zasnovano na linearnih, časovno nespremenljivih sistemih. Privzetek o linearnosti in časovni nespremenljivosti velja za različne tehnične sisteme, posebno za sisteme, kjer sta vhodni in izhodni signal omejena znotraj določenega amplitudnega obsega. Veliko analognih naprav za procesiranje zvoka pa ima v svojem delovanju tudi nelinearnosti, npr. ojačevalci z elektronkami, analogni efekti, analogni tračni snemalniki, zvočniki, konec koncev tudi človeški slušni mehanizem. Simulacije in kompenzacije teh nelinearnosti pa zahtevajo nelinearno procesiranje signala, kar pa s sabo prinese tudi teorijo nelinearnih sistemov (Volterrajeva vrsta kot razširitev Taylorjeve) [11].

Raziskave potekajo predvsem na področjih približkov in simulacijskih tehnik nelinearnih procesorjev ter njihovih shem za implementacijo.

Zvok električne kitare spreminjamo tako, da »distorziramo« (tj. popačimo, zvijemo) obliko vala. Ti učinki prištejejo dodatne tone oz. harmonične elemente v izvorni signal, posledica tega sta daljše trajanje zaigranih not (t. i. *sustain* se izboljša) in bogatejši ton.

Ustvarjene modele sem združil v sistemu *distortion*. Enako kot pri prejšnjem sistemu učinkov oktaviranja je tudi tukaj kot izvor signala uporabljen sinusni generator. Kot ponor je ponovno uporabljen blok, ki izriše diagrame signalov, v tem primeru izvirni sinusni signal in štirje obdelani signali. Štirje različni modeli distorzij so prikazani kot bloki, katerih vsebino vidimo, če jih dvakrat kliknemo.



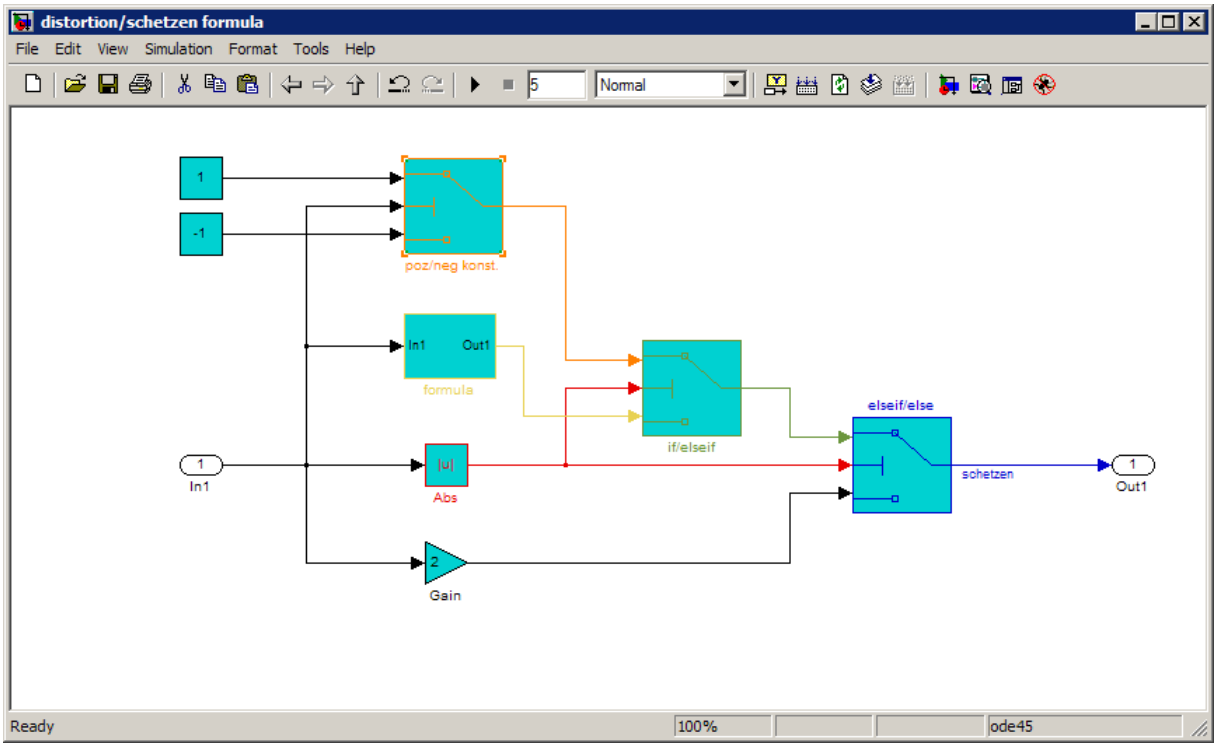
Slika 7: Sistem učinkov distorzije v Simulinku

4.2 Simetrično mehko rezanje (*simetrical soft clipping*)

Za simulacije preobremenjenega ojačevalca moramo izvesti simetrično mehko rezanje vhodnih vrednosti. Enega od možnih pristopov ponuja t. i. formula Schetzen [11]:

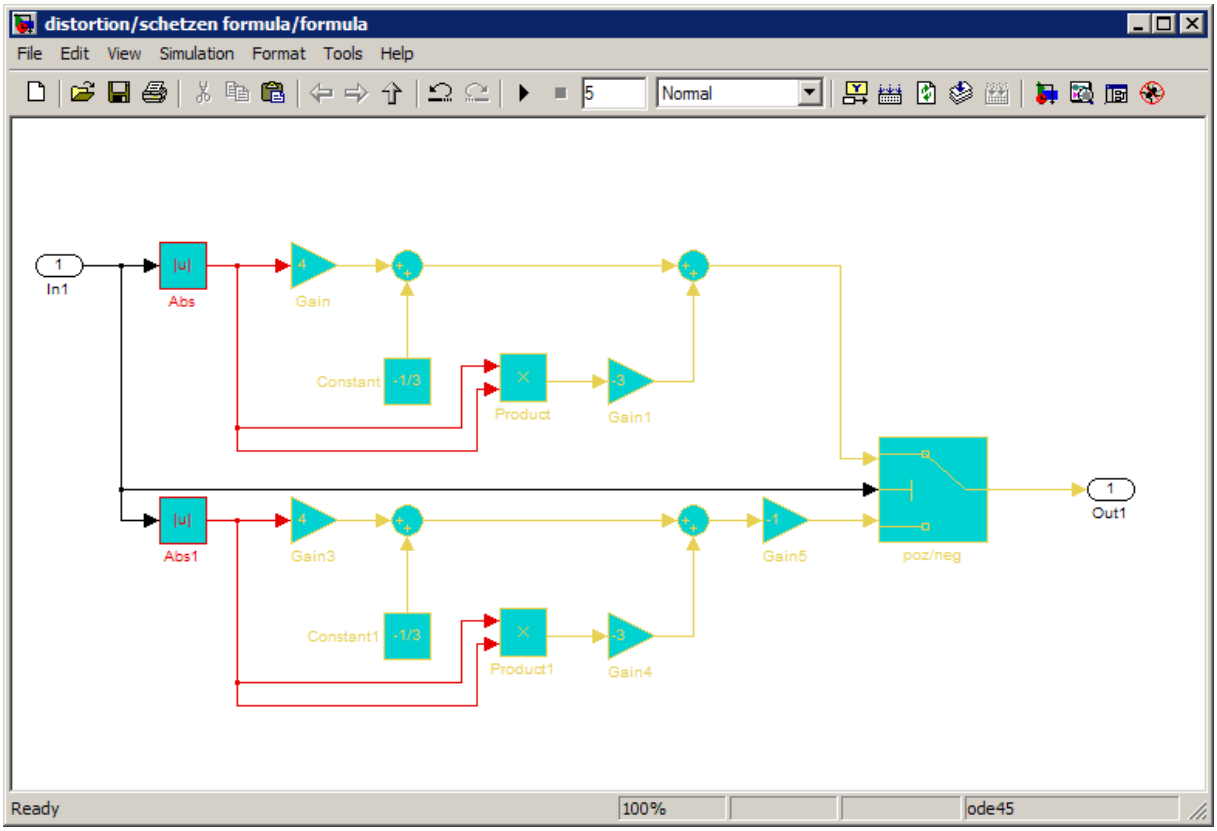
$$f(x) = \begin{cases} 2x \; za \; 0 \leq |x| \leq \frac{1}{3}; \\ \frac{3-(2-3x)^2}{3} \; za \; \frac{1}{3} < |x| < \frac{2}{3}; \\ 1 \; za \; \frac{2}{3} \leq |x| \leq 1. \end{cases} \qquad (5)$$

Do prve tretjine se vhod množi z dve, karakteristična krivulja pa je v linearnem območju. Med prvo in drugo tretjino je izhod rahlo stisnjen, v zadnji tretjini pa je izhodni signal enak 1. Formula je v Simulinku uresničena z naslednjim modelom.



Slika 8: Model formule Schetzen v Simulinku

Desno stikalo izbira med prvo vrstico formule (5) in ostalima dvema, glede na absolutno vrednost vhodnega signala. Podobno je s srednjim stikalom, ki izbira med drugo in tretjo vrstico. Tretja vrstica je realizirana z levim stikalom, ki izbira konstanti glede na predznak vhodnega signala. Drugo vrstico sem realiziral kot nov blok, ki sem ga poimenoval *formula*. Sledi slika tega bloka.



Slika 9: Blok *formula* v Simulinku

Vsebino oklepaja v števcu druge vrstice formule (5) sem razvil, da sem jo lahko realiziral z bloki. Stikalo na desni izbira glede na predznak vhodnega signala.

Ta model je uspešno prestal tako testiranje s sinusnim generatorjem, kot tudi preizkus z zvočno datoteko. Pri slednjem so posamezni toni slišati glasnejši in širši, akordi pa povzročijo nežno distorzijo, ki se hitro razčisti s pojemanjem glasnosti signala. Zaradi gladke krivulje končni izid ni tako brenčeč kot naslednji primer.

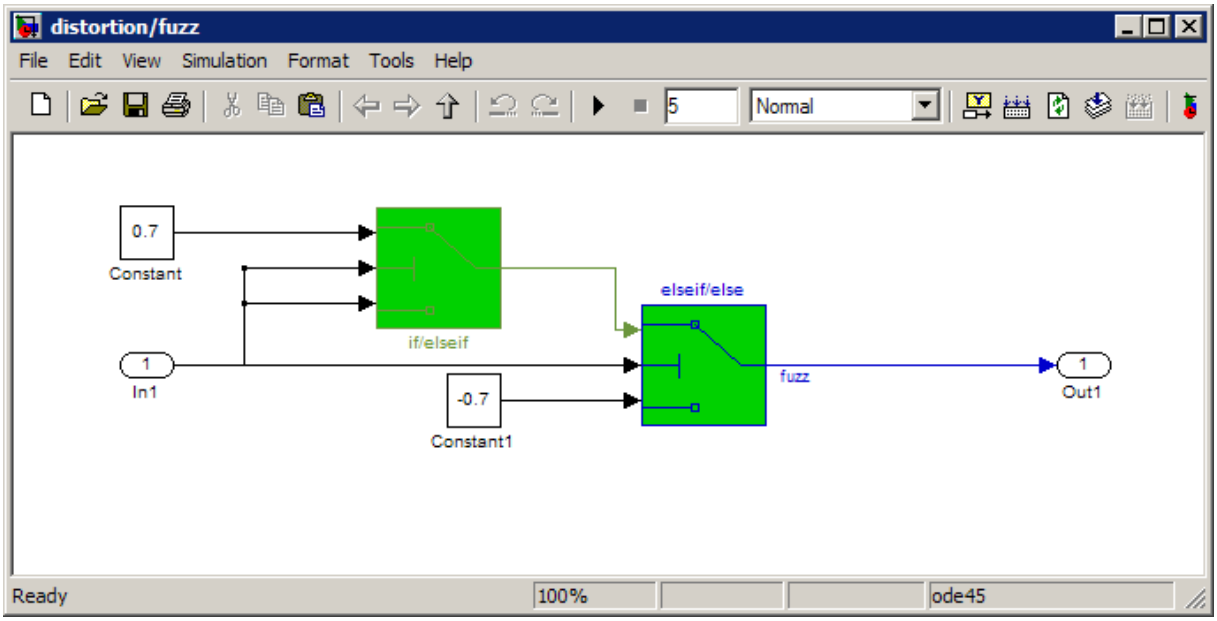
4.3 Rezanje (*clipping*) signala

Za učinek *fuzz* sem se zgledoval po algoritmu, ki učinek doseže z rezanjem signala [1]:

```
/*****************************************************/
/* fuzz()                                            */
/*****************************************************/
/* Function takes an input in range -32767 to + 32767  */
/* from the ADC and limits the range                 */
/*****************************************************/

int fuzz(int ADC_value)
      {
      float input;
      float output;
/* Convert ADC input to value between -1.00 and +1.00 */
      input = (float) ADC_value / 32767.0;

      if (input > 0.7)
            {output = 0.7;}
```

```
        else if (input < -0.7)
                {output = -0.7;}
        else
                {output = input;}

/* Convert from 1.00 to 32767.0 */
        return ((int)(output * 32767.0)) ;
        }
```

Kot je razvidno iz naslednje slike, je zgornja psevdokoda prenesena v bloke Simulinka. Levo stikalo preverja, ali je vhodni signal večji od 0,7, v tem primeru preklopi izhod na konstanto. Desno stikalo pa preverja, ali je izvorni signal manjši od –0,7. V primeru, da je, preklopi izhod na konstanto, sicer pa je izhod kar izhod levega stikala. Pretvorba tipov vhodnih in izhodnih vrednosti v tem primeru ni bila potrebna.



Slika 10: Model učinka *fuzz* v Simulinku

Tudi ta model je uspešno prestal oba preizkusa, sinusni generator in zvočno datoteko. Obdelana zvočna datoteka za razliko od primera v 4.2 zveni bolj popačeno in sintetično, skoraj na meji kvadratnega vala. Brenčeča karakteristika zvoka se manifestira tudi pri lomljeni krivulji signala. Ker dodane frekvence zvenijo disonančno, se pri uporabi tega učinka redkeje igrajo tri- in štiriglasja s terčnimi razdaljami, pogosteje pa enoglasja ter enostavna dvoglasja s kvintno oz. oktavno razdaljo. Eden prvih in najbolj znanih primerov uporabe tega učinka je vsekakor pesem *I can't get no satisfaction* skupine *The Rolling Stones* [4].

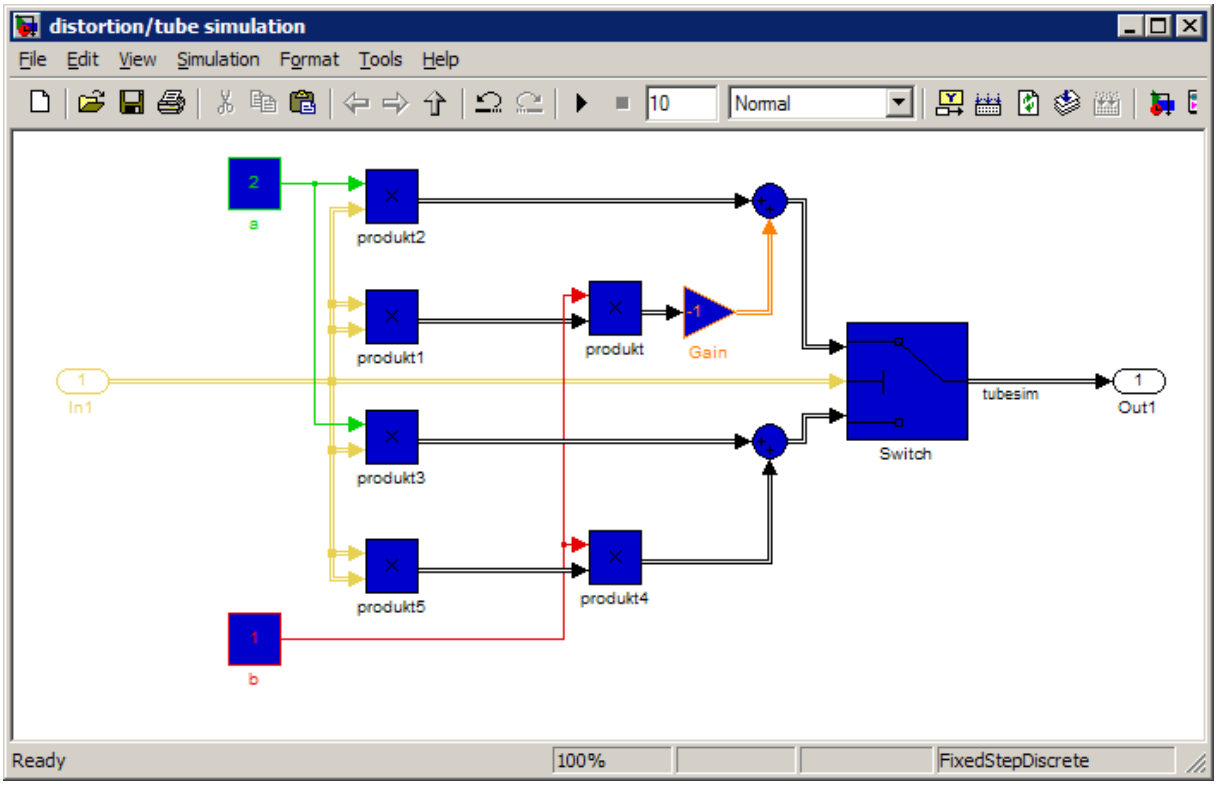
4.4 Simulacija učinka elektronk

Naprave z elektronkami so prevladovale v signalnem procesiranju v prvi polovici prejšnjega stoletja, v zadnjih desetletjih pa doživele prebujenje. Eden najbolj splošnih efektov za električno kitaro je namreč kar ojačevalec sam, sploh pa takšen z elektronkami – gre za preobremenitev vhodne in končne stopnje, kar ustvari posebno distorzijo. Na zvok ojačevalca poleg samih elektronk vplivata še samo vezje ojačevalca in šasija zvočne skrinje z zvočnikom (ki je lahko tudi »obdelan«). Poleg dveh najbolj znanih kitar (Fender Stratocaster in Gibson Les Paul) je pri določanju standardnega zvoka električne kitare imelo pomembno vlogo nekaj

ojačevalcev na elektronke (Fender, Vox, Marhsall, Mesa-Boogie). Učinek ojačevalca na elektronke simuliramo z uporabo nelinearne prenosne funkcije [1]:

```
/*****************************************************/
/* valve_sound()                                     */
/*****************************************************/
/* Function takes input in range -32767 to + 32767 and */
/* calculates the output from y = a*x + b*x*x where    */
/* x = input from ADC and y = output                   */
/* Use different values of a and b to alter the effect */
/* preizkušeno a=b=4; a=3.5 b=4.5                      */
/*****************************************************/

#define a 2.000
#define b 1.000

int valve_sound( int ADC_value, float a, float b)
      {
      float input;
      float output;

/* Convert ADC input to value between -1.00 and +1.00 */
      input = (float) ADC_value / 32767.0;

      if (input >= 0.0)         /* Positive input values */
            {output = a * input - b * input * input;}
      else                              /* Negative input values */
            {output = a * input + b * input * input;}

/* Convert from 1.00 to 32767.0 */
      return ((int)(output * 32767.0)) ;
      }
```

Zgornjo psevdokodo sem brez posebnih težav prevedel v Simulinkov model. Stikalo preklaplja med dvema možnostma glede na predznak vhodnega signala, zgornja in spodnja veja pa se razlikujeta le po predznaku kvadratnega člena. Pretvorba tipov ponovno ni bila potrebna.



Slika 11: Model simulacije učinka elektronk v Simulinku

Kot je razvidno, tukaj pozitivne vhodne vrednosti množimo s parametrom a, nato pa jim odštejemo s parametrom b pomnožene oktavo višje harmonske komponente. Podobno je pri negativnih vhodnih vrednostih, le da jim tukaj oktavo višje harmonične elemente prištevamo (oktavno moduliranje je razloženo v poglavju 3.5).

Niti pri tem modelu nisem izkusil težav pri preizkušanju s sinusnim generatorjem in zvočno datoteko. Rezultat je močno podoben prvemu primeru (formuli Schetzen), z razliko da tukaj lažje spreminjamo parametra a in b (amplitudi), hkrati pa dosežemo tudi večje popačenje in glasnost signala. Empirične izkušnje kažejo, da vrednosti parametrov a in b ne smeta biti preveč narazen (razlika naj bo 1 ali manj):

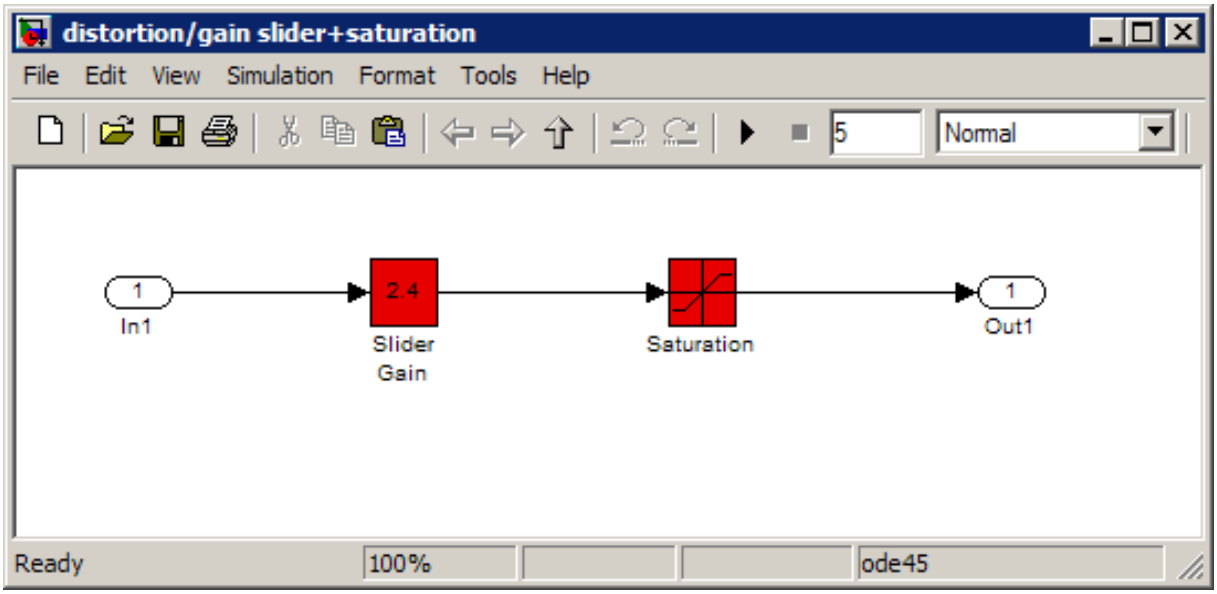
- povečevanje izključno parametra a ne naredi nič;
- povečevanje izključno parametra b povečuje prisotnost oktavo višjih harmonskih komponent in stopnjo popačenja signala, vendar ga tudi naredi občutno tišjega;
- hkratno povečevanje obeh naredi signal glasnejši in bolj popačen.

Tudi tukaj izhodni signal dobi nežno popačenje ob večji amplitudi oz. več sočasno zaigranih tonih, izhodni signal pa se lepo razčisti z zmanjševanjem amplitude vhodnega signala.

4.5 Simetrično nasičenje

Na idejo drugačne izvedbe sem prišel med brskanjem po knjižnici Simulinkovih blokov. Naletel sem na element ojačenja, ki pa ga lahko spreminjamo z drsnikom (seveda mu predhodno nastavimo spodnjo in zgornjo mejo), brez potrebe po dostopanju in spreminjanju lastnosti bloka ter shranjevanju modela pred zagonom simulacije, kot je navada pri običajnem elementu ojačenja. Drugi blok se imenuje *saturation* oz. nasičenje – nastavimo mu spodnjo in zgornjo mejo, deluje pa tako, da omejuje vhodni signal: vse vrednosti vhodnega signala, ki so

višje od zgornje meje, se preslikajo v zgornjo mejo; analogno se zgodi z vrednostmi, nižjimi od spodnje meje. Vrednosti med določenima mejama nedotaknjene zapustijo element. Preostalo je le še zaporedno vezati oba elementa in preizkusiti različne parametre ojačenja.

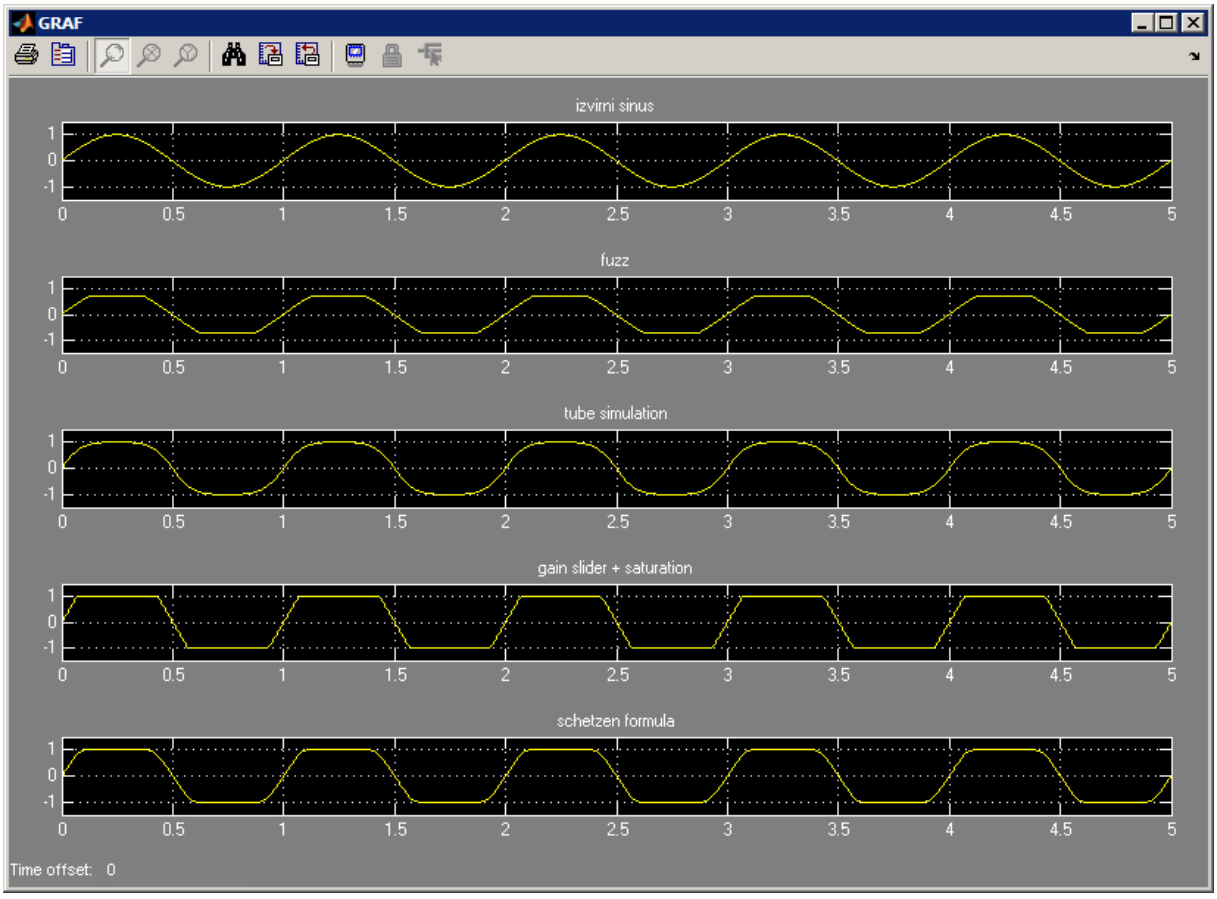


Slika 12: Model simetričnega nasičenja v Simulinku

Pravzaprav sem ustvaril fleksibilnejšo različico učinka *fuzz* (razložen v poglavju 4.3). Amplitudo izhodnega signala sem omejil med –1 in 1 z blokom *saturation*, s spremenljivim ojačenjem pa uravnavam strmino krivulje med spodnjo in zgornjo mejo. Vse navedeno pri učinku *fuzz* velja tudi tukaj, s poudarkom, da lahko v tem primeru dosežemo veliko izrazitejši učinek.

4.6 Graf družine učinkov distorzij

Na koncu opisa družine učinkov popačenj navajam še graf, ki je rezultat simulacije opisanih učinkov. Prvi diagram je neobdelani sinusni signal, ostali diagrami pa prikazujejo sinusni signal po prehodu skozi različne modele popačenja. Trajanje simulacije je 5 sekund, generator signala pa ustvarja sinus s frekvenco 1 Hz, amplitude od –1 do 1. Iz grafa so lepo razvidne vse operacije, ki jih nad vhodnim signalom izvedejo posamezni modeli. Ponovno sem za potrebe jasnega prikaza uporabil zelo nizko frekvenco sinusnega signala.



Graf 2: Graf učinkov distorziranja v Simulinku

Kot je razvidno iz zgornje slike, tukaj signalu ne spreminjamo frekvence. Če primerjamo prvi diagram z vsakim naslednjim, lahko opazimo naslednje:

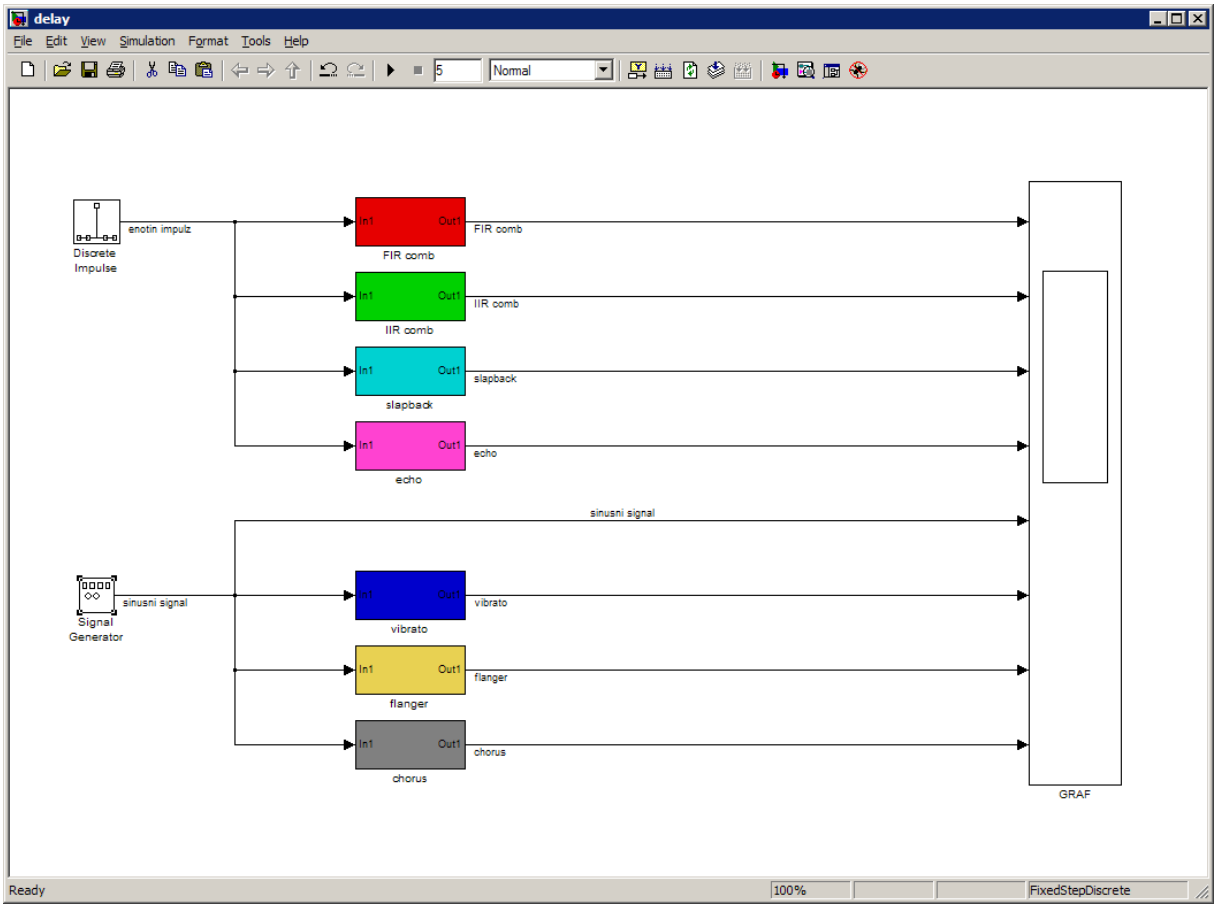
- pri drugem diagramu je razvidno rezanje amplitud, ki so absolutno večje od 0,7; jasno so opazne tudi nezveznosti krivulje; tudi po videzu mu je podoben četrti diagram, kot je že prej opisano v besedilu;
- tretji in peti diagram sta zelo podobna, le da ima peti strmejši prehod med ekstremoma vrednosti amplitude, posledično pa se tudi dlje časa zadržuje na teh vrednostih; tudi zvok zato vsebuje več harmonikov, ki izvorno niso prisotni.

5 Učinki zakasnitev

5.1 Odboji in zakasnitve

Zakasnitve različnih vrst lahko izkusimo v akustičnih prostorih, kjer se zvočni signal odbija od površin in prišteva k zvočnemu signalu na njegovem izvoru. Če je površina zelo oddaljena, slišimo odboj oz. eho; če je površina bližje, slišimo spremembo v barvi zvoka; med vzporednima površinama imamo lahko tudi ponovljene odboje [11].

Vsi opisani in realizirani bloki so vključeni v sistem *delay*, ker gre za učinke iste družine. Kot izvor se tokrat poleg generatorja sinusnega signala pojavi še generator enotinega impulza. Oba izvora dobro delujeta z vsemi modeli, toda vezava je takšna zaradi prikaza učinkov na diagramih, ki jih izriše ponor. Pri prvih štirih modelih učinek na sinusnem signalu namreč ni lepo razviden, podobno pa velja za zadnje tri modele in enotin impulz.



Slika 13: Sistem učinkov zakasnitve v Simulinku

5.2 Filter s končnim enotinim odzivom

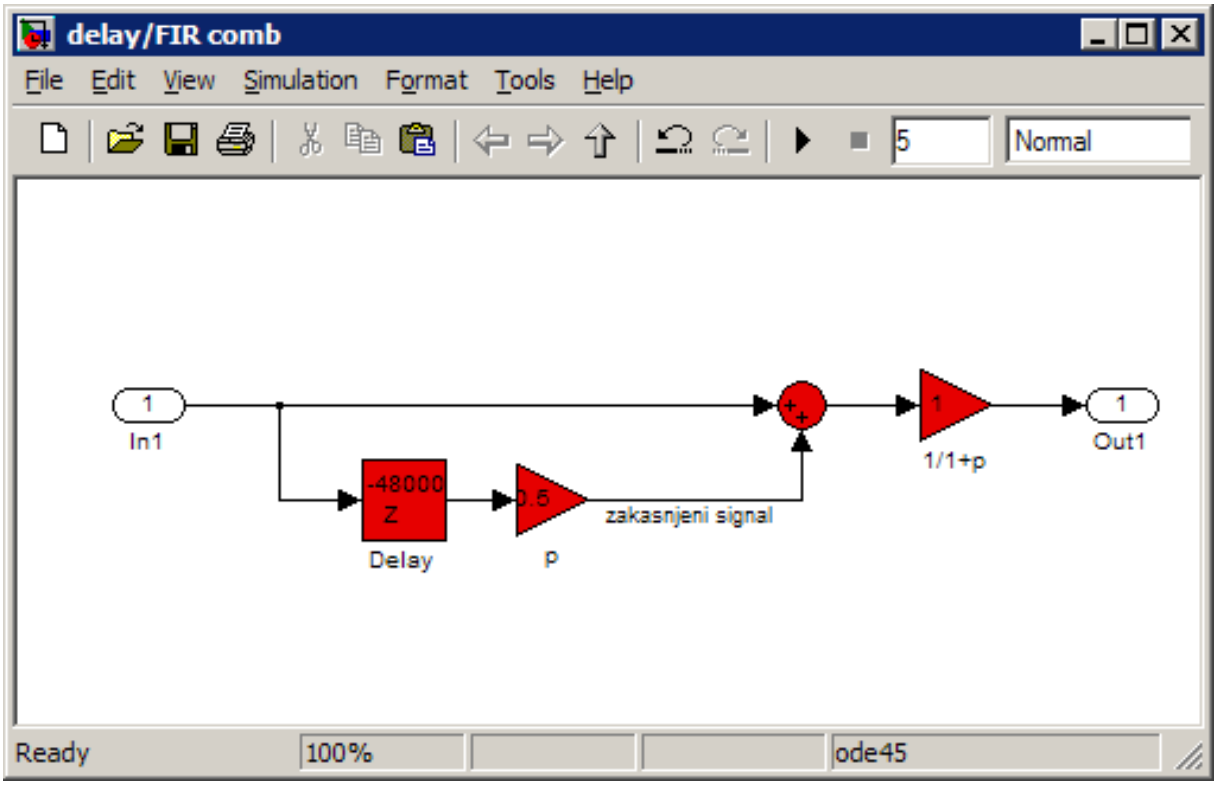
Najenostavnejša shema vsebuje zakasnilni element, ki vhodni signal zakasni (čas zakasnitve t je prvi parameter), ter element za nastavitev amplitude zakasnjenega signala (relativna amplituda zakasnjenega signala glede na prvotni signal je drugi parameter – učinek namreč slišimo le, če zakasnjeni signal prištejemo k prvotnemu). Rezultat sheme je t. i. *FIR comb filter* z zelo enostavnima diferenčno enačbo in prenosno funkcijo [11]:

$$y(n) = x(n) + p * x(n - M),\ M = t * f_s \qquad (6)$$

$$H(z) = 1 + p * z^{-M} \qquad (7)$$

Spremenljivka M je tukaj število vzorcev, za katerega zakasnimo signal (čas želene zakasnitve, pomnožen s frekvenco vzorčenja). Za pozitivne vrednosti p filter ojači vse frekvence, ki so večkratnik 1/t, zmanjša pa vse frekvence, ki ležijo vmes. Prenosna funkcija takega filtra je videti kot niz konic, od tod ime glavnik. Za negativne vrednosti p je situacija ravno obratna: filter zmanjša večkratnike 1/t, ojači pa frekvence vmes. Celotno ojačenje je med 1 – p in 1 + p.

Za velike vrednosti *t* ne zaznamo efekta glavnika (*comb*), zaradi premajhne medsebojne razdalje frekvenc, ki jih efekt ojači – slišimo le ponovljen vhodni signal; za male vrednosti *t* pa uho ne loči več časovnih dogodkov in lahko dobro razloči spektralni učinek glavnika.



Slika 14: Model *FIR comb* v Simulinku

Slika prikazuje prvega izmed blokov v modelu efektov, zasnovanih na zakasnitvenih elementih. Gre za najenostavnejšega izmed vseh: signal zakasnimo za poljubno število vzorcev in ga prištejemo k prvotnemu. Z elementom ojačitve p uravnavamo količino zakasnjenega signala v izhodnem signalu, z zadnjim elementom ojačitve pa amplitudo izhodnega signala zadržimo znotraj mej intervala [–1, 1].

V praktičnem delu sem parameter zakasnitve M nastavil na 16000 vzorcev, kar pri vzorčenju s 44,1 kHz znaša nekaj več kot 0,3 sekunde – tukaj je efekt že dobro slišen. Preostali parameter je p, ki je lahko npr.:

- enak 1 – pri dovolj počasni melodiji se posamezni toni lahko prekrivajo in dobimo učinek kánona, kar je eden od možnih načinov uporabe tega učinka.
    - kánon predstavlja svojevrstno petje, kjer glavnemu vokalu sledi v določenem časovnem intervalu (načeloma po enem ali dveh taktih) ena ali več imitacij;

kánon petje z dejanskimi imitacijami (se pravi z določenim odstopanjem od melodije glavnega vokala) je redko, navadno govorimo o popolnem ponavljanju oz. o ponavljanju na določenem glasbenem intervalu (se pravi, da se identična melodija za nekaj tonov dvigne oz. spusti).

- enak 0,2, kar je veliko bolj realistično in imamo pravi učinek odboja; takšnega srečamo najpogosteje.

Čeprav pri testiranju s sinusnim signalom zakasnitve iz grafa ni bilo moč razbrati (za grafični prikaz učinka sem uporabil blok enotinega impulza), pa učinek zablesti pri preizkusu z zvočno datoteko. Slišati ni nobenega zmanjšanja kakovosti signala, učinek pa je uporaben za vse možne kombinacije eno- in večglasij.

Za specifične nastavitve določenih parametrov so se uveljavila posebna imena tega učinka [11].

5.2.1 Slapback

Če uporabimo filter KEO s časom zakasnitve 10–25 ms (pri vzorčenju s 44,1 kHz torej znaša zakasnitev 441–1203 vzorcev), slišimo hitro ponavljanje, čemur pravimo *slapback* oz. podvajanje.

5.2.2 Echo

Če uporabimo filter KEO s časom zakasnitve 50 ms (torej znaša zakasnitev 2205 vzorcev), slišimo eho oz. odboj. Tako majhne razlike med ehom in *slapback-om* so sicer skoraj nezaznavne človeškem ušesu (čeprav dobro vidne na grafu), vendar nekaj milisekund določa namen učinka, saj tako narekujejo desetletja izkušenj strokovnjakov v glasbeni in filmski industriji.

5.3 Filter z neskončnim enotinim odzivom

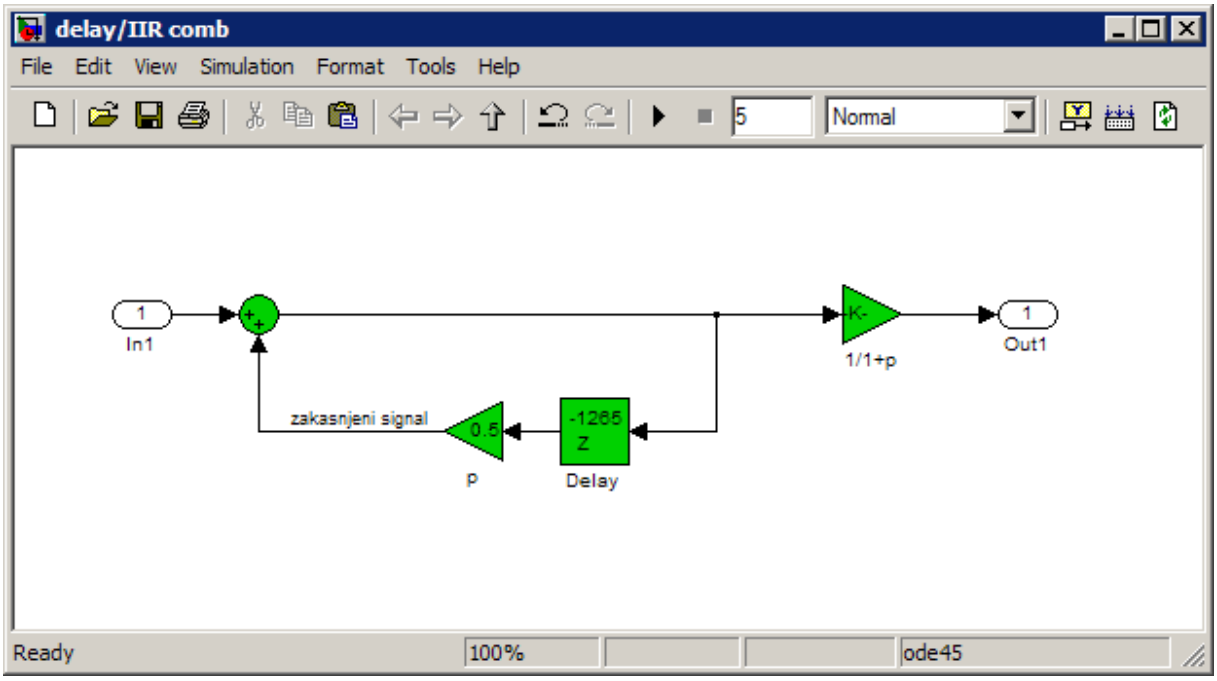
Druga možnost realizacije zakasnitve je *IIR comb filter*, ki ga opisujeta nič bolj zapleteni enačbi [11]:

$$y(n) = x(n) + p * y(n - M),\ M = t * f_s \qquad (8)$$

$$H(z) = 1 - p * z^{-M} \qquad (9)$$

Zaradi povratne zanke je časovni odziv filtra neskončno dolg – po vsaki časovni zakasnitvi *t* se na izhodu pojavi kopija vhodnega signala z amplitudo p^n, kjer je n število prehodov signala skozi zakasnitveni element. Pogoj za stabilnost sistema je seveda |p|<=1, sicer bi amplituda signala naraščala v neskončnost. Ta filter NEO podobno vpliva na frekvence kot filter KEO, glavna razlika pa je, da zaradi povratne zanke amplituda signala lahko zelo narašča.

Zgradba tega bloka je skoraj identična prejšnjemu, uporabljeni so enaki gradniki. Razlika je edino v tem, da je signal zakasnjen v povratni zanki, ne pa vzporedno, kot je v prejšnjem primeru.



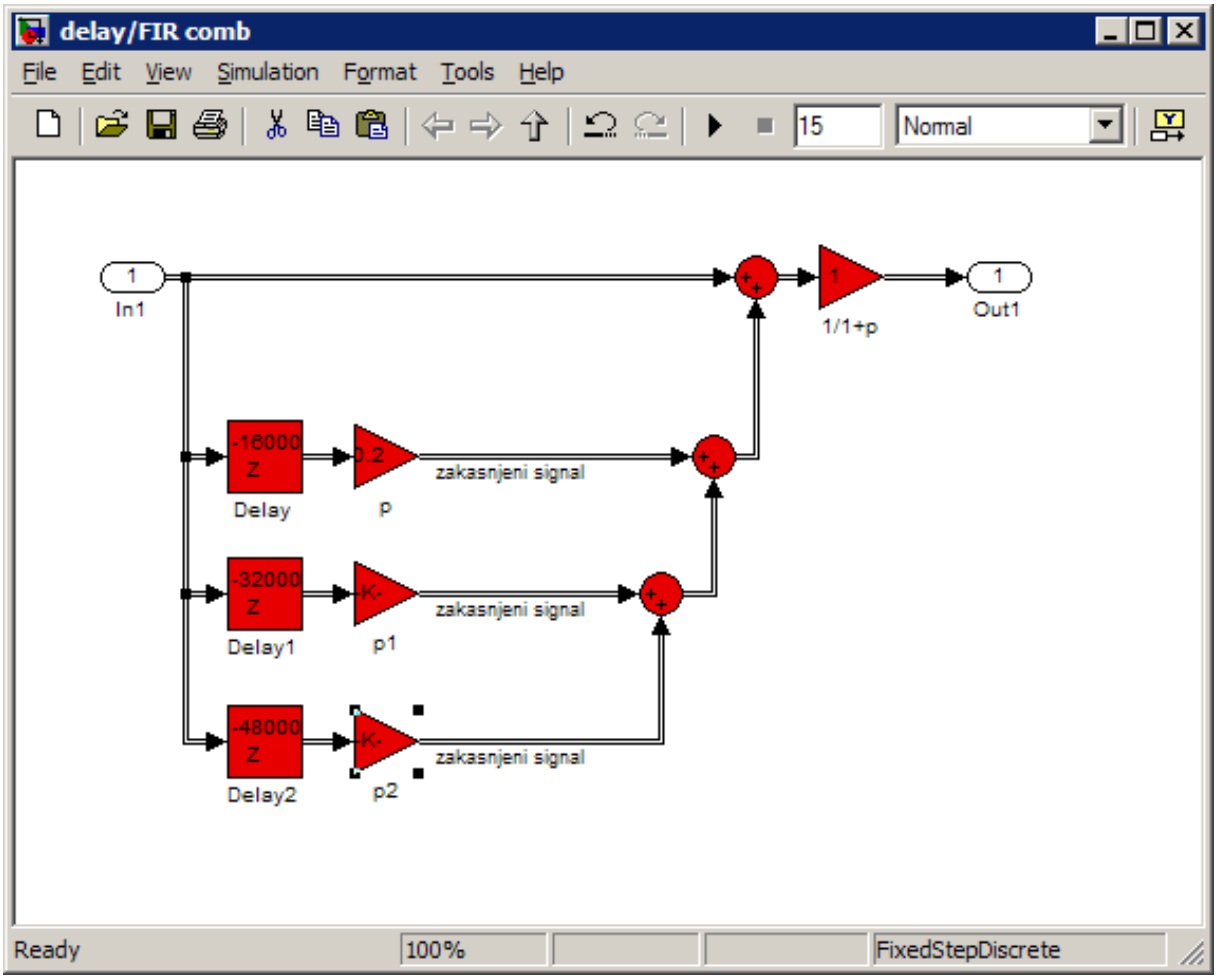
Slika 15: Model *IIR comb* v Simulinku

Pri praktičnem poskusu z zvočno datoteko program Matlab preneha z delovanjem in zahteva ponoven zagon, razlog pa je očitno povratna zanka. Matlaba ne poznam dovolj, da bi podrobneje ocenil razlog – vgrajena zaščita proti neskončnem odzivu ali pa napaka zaradi začetnih pogojev rekurzivnega dela kode. Vseeno se bolj nagibam k začetnim pogojem kot razlogu, saj se Matlab enako odziva na vsako povratno zanko. Dejstvo pa je, da v času 0 še ni česa poslati po povratni zanki.

Neuspeh popolne realizacije modela močno obžalujem, saj bi zaradi povratne zanke in posledično »naravnega« odmiranja signala (z vsakim obhodom zanke je ponovljeni signal tišji) ta učinek bil še najbolj realističen. Nedelovanje me preseneča, saj je simulacija z generatorjem sinusnega signala oz. blokom enotinega impulza kot izvorom in grafom kot ponorom delovala brezhibno (glej graf sistema učinkov zakasnitev, ki sledi).

5.3.1 Simulacija učinka filtra NEO z več filtri KEO

Odmiranje signala bi lahko realiziral tudi s filtrom, ki ima končen odziv, vendar bi potreboval več blokov: po enega za vsak nov odboj, vsak naslednji pa bi imel za prvotno zakasnitev večji parameter M od svojega predhodnika in s p pomnožen parameter p svojega predhodnika. Primer izvedbe je prikazan na naslednji sliki. Uporabljenih je več zaporednih zakasnilnih linij, ki imajo različno zakasnitev.



Slika 16: Model simulacije *IIR comb* s *FIR comb* v Simulinku

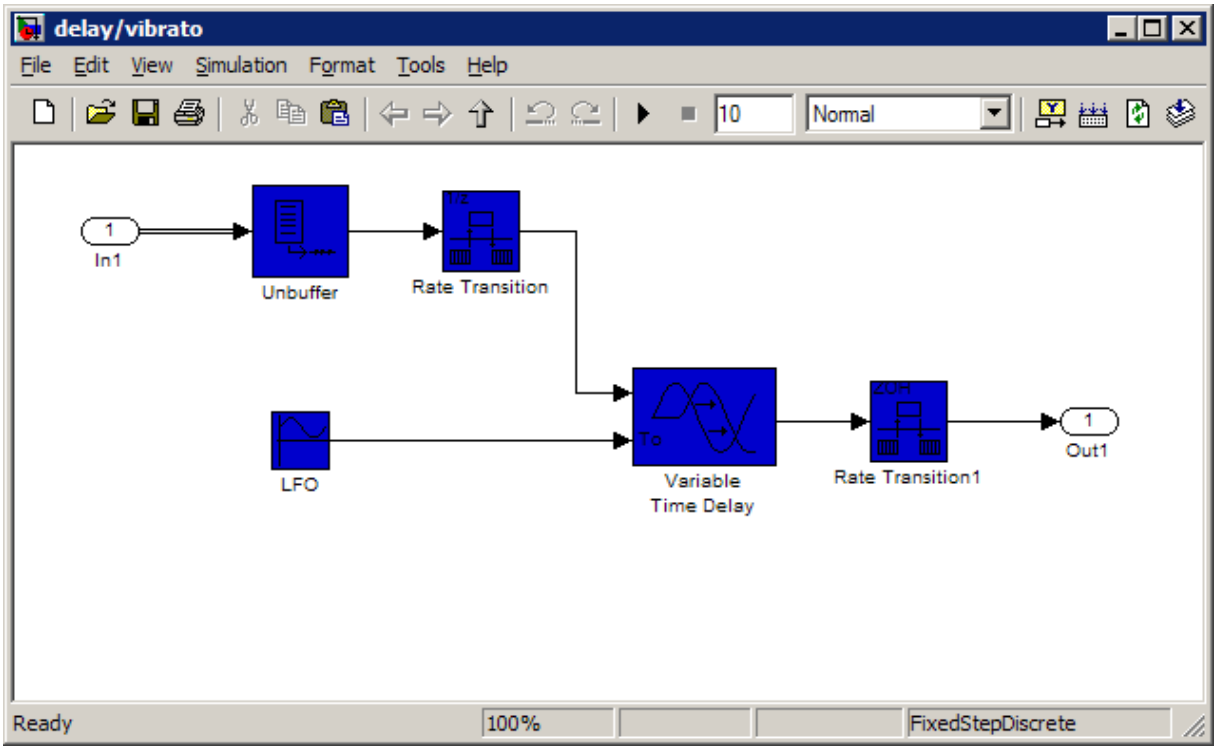
Taka realizacija sicer deluje po pričakovanjih, tako s sinusnim generatorjem kot tudi z zvočno datoteko, je pa nerodna za realizacijo, saj imamo namesto dveh več parametrov; višje vrednosti dveh parametrov realizacije NEO (večjo zakasnitev in glasnejšo ponovitev signala) pa smo primorani simulirati z dodajanjem več zaporednih zakasnilnih linij. Število linij lahko sicer matematično določimo, vendar moramo za dodajanje zakasnilnih linij posegati v načrt modela. Pri realizaciji s filtrom NEO je še dodatna prednost, da bi bilo mogoče omenjena parametra povsem preprosto povezati v morebitni grafični vmesnik. Dodajanja zakasnilnih linij ni mogoče izvesti, lahko pa bi jih že vnaprej namestil več, kot je potrebno – tistim, ki jih ne potrebujemo, bi parameter glasnosti ponovitve signala nastavili na nič. To pa bi za posledico imelo obsežen, nepregleden in uporabniku neprijazen vmesnik. Povsem jasno je, da je realizacija s filtrom NEO neprimerno elegantnejša.

5.4 Možne razširitve znotraj družine učinkov zakasnitev

Nadgradnja sistema je možna v več smereh. Zvočni signal lahko zajemamo oz. shranjujemo v stereo–tehniki, kar pomeni, da signal sestavljata dva kanala, ki se sočasno predvajata. Če ločeno obdelujemo vsakega od kanalov, lahko dobimo stereo–zakasnitev: (t. i. *ping-pong delay* ima na vsak kanal po en filter KEO, eden od dveh pa ima dvakrat večjo zakasnitev od drugega – poslušalcu se zdi, da se signal »odbija« levo in desno. S specifičnimi parametri filtra pa lahko dosežemo učinke, ki so zasnovani na zakasnitvah oz. izkoriščajo stranske pojave tega učinka (vse efekte lahko realiziramo tako s filtri KEO kot tudi filtri NEO).

5.4.1 Vibrato

Dopplerjev efekt dojemamo kot spreminjanje višine tona glede na spreminjanje razdalje med izvorom in našimi ušesi. V našem primeru je spreminjanje razdalje enako spreminjanju časa zakasnitve; če pa ta čas periodično spreminjamo, dosežemo periodično spreminjanje višine tona, kar je natanko učinek vibrata [11]. Za to potrebujemo nizkofrekvenčni oscilator, ki bo krmilil čas zakasnitve. Pomemben podatek je tudi, da v tem primeru poslušamo le zakasnjeni signal, izvornega pa povsem izključimo iz sheme. Tipične vrednosti so povprečno 5–10 ms za čas zakasnitve ter 5–14 Hz (kar je enako od 10π do 28π radianov) za frekvenco oscilatorja.



Slika 17: Model vibrata v Simulinku

Pri tem učinku zakasnjenega signala ne prištevamo več k prvotnemu, temveč na izhod pošljemo kar zakasnjenega. Uporabljeni blok za zakasnitev omogoča dinamično določanje časa zakasnitve, krmilimo pa ga z nizkofrekvenčnim oscilatorjem. Ta je implementiran z gradnikom sinusne funkcije LFO, ki ji določamo amplitudo, odmik, frekvenco in fazni zamik. Enota frekvence je v radianih, amplituda se razteza od –0,003 do 0,003, odmik pa znaša 0,007. Izhod oscilatorja so torej vrednosti med 0,004 in 0,01 sekunde, ki se po sinusni krivulji spreminjajo s frekvencami med 10π in 28π radianov, kar določimo pred zagonom.

Pri preizkusu s sinusnim generatorjem je na grafu lepo vidno periodično spreminjanje frekvence, zagon simulacije z zvočno datoteko pa je generiral napako. Po dodatnem preučevanju lastnosti blokov sem napako poskusil odpraviti na naslednji način.

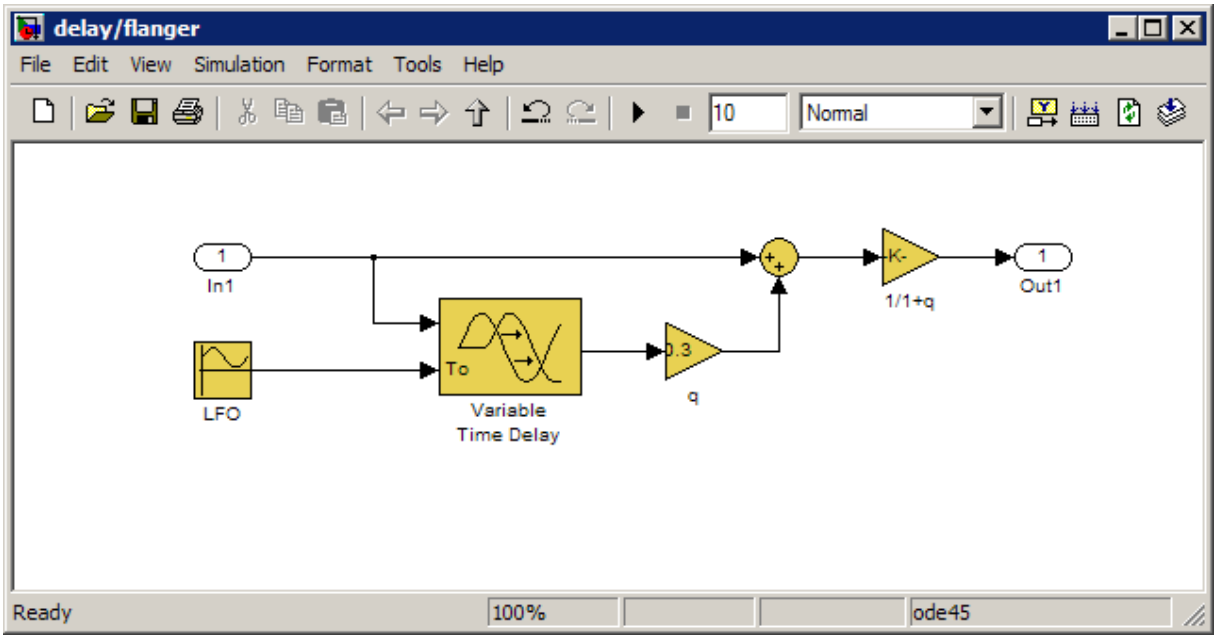
Za razliko od ostalih učinkov iz družine odbojev in zakasnitev je bil pri realizaciji tega potreben še blok *unbuffer*, ki spremeni vhod dimenzij M_i*N (*frame-based*) v izhod dimenzij 1*N (*sample-based*). To pomeni, da je vhod vrstično razcepljen tako, da je vsaka vrstica matrike zdaj neodvisen vzorec. Hitrost, s katero blok prejema na vhodu, je v splošnem manjša od hitrosti, s katero ustvarja izhode. Blok s spremenljivo zakasnitvijo ima namreč vhod tipa *sample-based*, blok, ki bere datoteke wave, pa ustvarja izhod tipa *frame-based*. Blok *rate*

transition prenaša podatke z izhoda enega bloka, ki deluje z določeno hitrostjo, na vhod drugega bloka, ki deluje z drugačno hitrostjo. Potreben je torej za usklajevanje hitrosti oddajanja in sprejemanja podatkov.

Čeprav se simulacija z zvočno datoteko izvede do konca, končnega izdelka ni mogoče predvajati (aplikacija Winamp javi napako s kodo 80070057, o kateri na internetu nisem našel ničesar, kar bi bilo povezano s to temo). Enako se žal zgodi tudi z naslednjima dvema učinkoma, ki sta v uporabi zelo pogosta in sem od njiju veliko pričakoval (razlog je najverjetneje v bloku *variable time delay*). Analogna realizacija je sicer dokaj trivialna, za digitalno pa očitno potrebujem več izkušenj.

5.4.2 Flanger

Če je čas zakasnitve majhen (manjši od 15 ms), hkrati pa ga spreminjamo z zelo nizko frekvenco (npr. 1 Hz), slišimo efekt, podoben preletu letala [11].



Slika 18: Model flanger v Simulinku

Izvedba je enaka kot pri prvem modelu (glej 5.2), le da je blok s konstantno zakasnitvijo nadomeščen z blokom, ki mu dinamično krmilimo zakasnitev. V tem primeru je uporabljen generator sinusa, kot je opisano v 5.4.1.

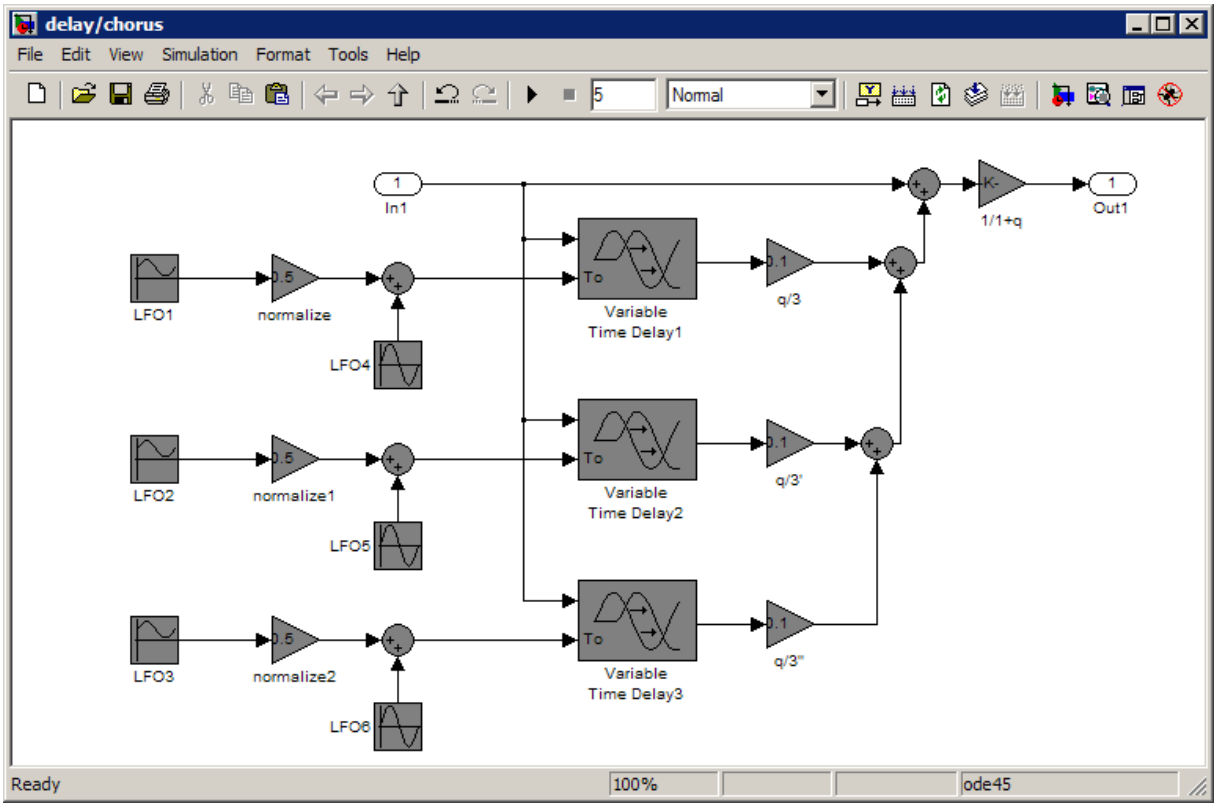
5.4.3 Chorus

Če je čas zakasnitve med 10–25 ms, hkrati ga pa naključno spreminjamo, dobimo učinek zbora [11]. Koliko članov bo štel zbor, je odvisno od števila vzporednih zakasnitvenih linij, ki jih implementiramo. Zaželeno je, da ima vsaka različno zasejan psevdonaključni generator, da se izognemo podvajanju glasov. Pri izvedbi tega učinka sem naletel na težavo, in sicer ravno z generatorjem naključnih števil. Ta je sicer deloval brezhibno, težava pa je, da generator ne tvori zvezne funkcije – vsakič, ko je spremenil čas zakasnitve s tem, da je generiral novo naključno število, je na izhodu učinka prišlo do nezveznosti.

Iz izkušenj s programsko opremo za snemanje in postprodukcijsko obdelavo lahko povem, da se bo to slišalo kot neprijetno pokanje ob vsaki spremembi časa zakasnitve. Česa takega si ne

moremo privoščiti v nobenem primeru (razen, če je ravno to pokanje cilj). Zato predlagam naslednjo rešitev: vsak zakasnitveni element krmilimo z generatorjem sinusnega signala, od katerih ima vsak različno frekvenco, vsakemu pa prištevamo še en sinusni signal, spet vsak s svojo frekvenco. Menim, da takšna rešitev daje dovolj naključne spremembe zakasnitve – konec koncev govorimo tukaj o intervalih, ki merijo nekaj milisekund. Zaradi prepletanja treh takih signalov z izvirnim menim, da nekega ponavljanja ne bo opaziti, sploh zato, ker gre za učinek na glasbenem materialu, ki pa je le tisto, kar se posluša. Tudi pri tovrstnih analognih napravah gre za generatorje sinusnega signala z različnimi frekvencami in zamiki.

Po testiranjih na zvočnem materialu bi bilo jasneje, kolikšno je optimalno število članov zbora.



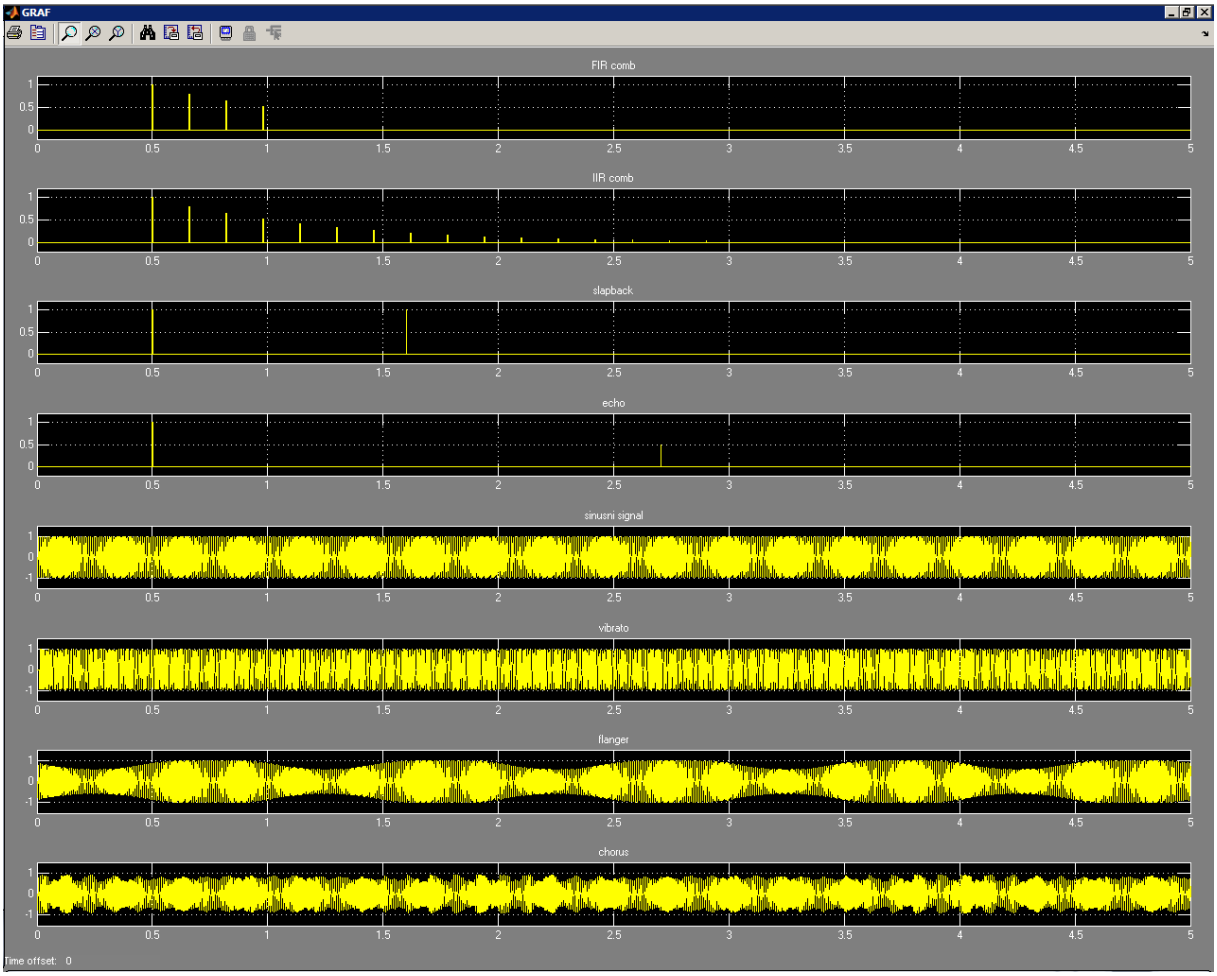
Slika 19: Model chorus v Simulinku

5.5 Graf družine učinkov zakasnitev

Za potrebe testiranja sem oba izvora in vse bloke povezal na isti graf. Tako najlažje primerjamo posamezne učinke. Pri drugem diagramu (*IIR comb*) je lepo razvidno, kako enotin impulz eksponentno pojema z zelo enostavno realizacijo, prvi diagram (*FIR comb*) pa ima toliko ponovitev, kolikor je vzporednih zakasnitvenih linij. Tretji in četrti diagram (*slapback* in *echo*) se na grafu jasno razlikujeta. Graf zajema 1000 vzorcev na sekundo, prav toliko pa jih proizvede enotin impulz. Njena enica se pojavi z zakasnitvijo 500 vzorcev, da je dogodek razvidnejši.

Čeprav prvi štirje učinki niso bili primerni za grafični prikaz s sinusnim signalom, so ostali trije odlično prikazani. Tokrat frekvenca 1 Hz (kot npr. pri oktaviranju ali distorziranju) ni bila primerna, saj so frekvence posameznih komponent znotraj modelov imele višjo frekvenco. Zato sem za sinusni signal izbral frekvenco 880 Hz, tj. ton A5, ki ga prikazuje peti

diagram. Pri šestem diagramu (*vibrato*) je opaziti periodično spreminjanje frekvence; zadnja dva diagrama (*flanger* in *chorus*) pa prikazujeta periodično spreminjanje amplitude signala. Pri sedmem diagramu je ta posledica enega nizkofrekvenčnega oscilatorja, pri osmem pa treh oscilatorjev z različnimi frekvencami. Vseeno je ob pazljivejšem pregledu mogoče opaziti periodičnost.



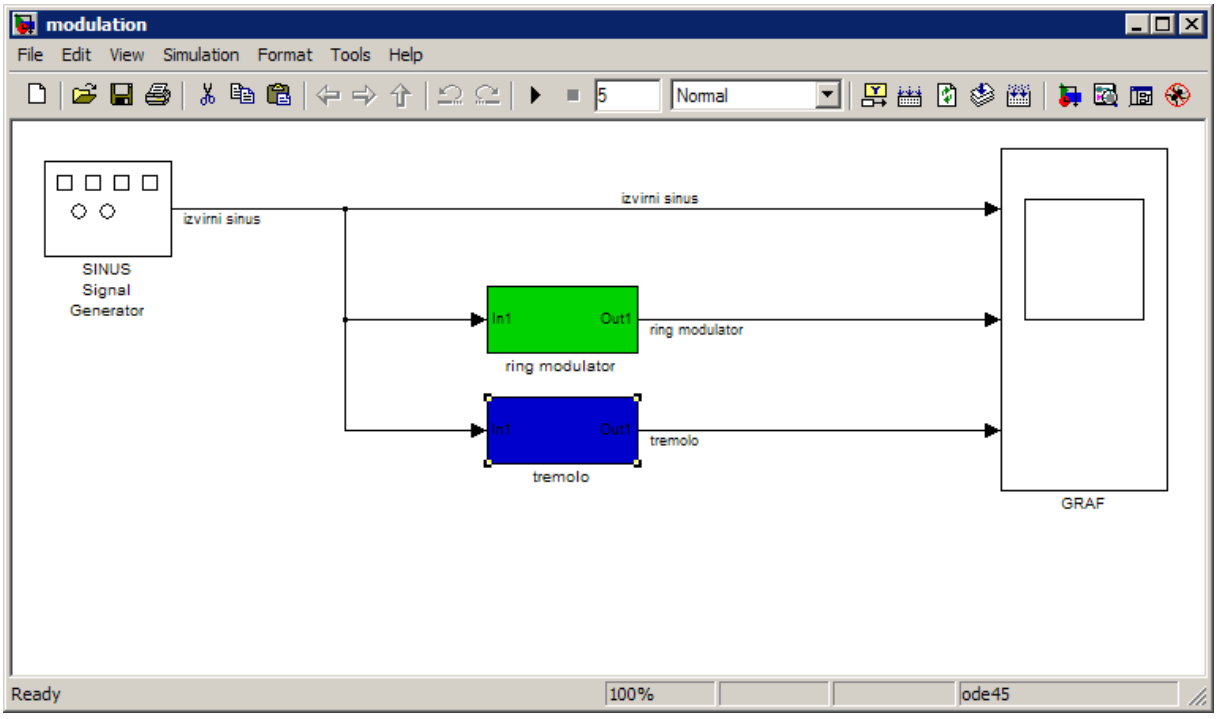
Graf 3: Graf učinkov zakasnitve v Simulinku

6 Učinki modulacije

Modulacija je proces, pri katerem parametre sinusnega signala (tj. amplitudo, frekvenco in fazo) spreminjamo z zvočnim signalom. Pri telekomunikacijah to pomeni »prestaviti frekvenčni spekter signala v drug frekvenčni pas«, za kar je ustvarjenih veliko načinov, nekaj teh pa je našlo svojo uporabo tudi v digitalnih efektih za inštrumente [11].

6.1 Moduliranje v glasbi

Modulacijske tehnike v procesiranju zvoka se pretežno uporabljajo z zelo nizkimi premiki frekvence zvočnega signala – nekatere filtre oz. zakasnitvene linije lahko razumemo kot amplitudno (*wah-wah*, *phaser* in *tremolo*) oz. fazno (*vibrato*, *flanger* in *chorus*) modulacijo. Združevanje enostavnih amplitudnih, faznih ter modulatorjev *single side band* vodi do zapletenih zvočnih učinkov. Enako obstajajo tudi demodulatorji, ki iz vhodnega signala razberejo njegove parametre, potrebne za nadaljnjo obdelavo.

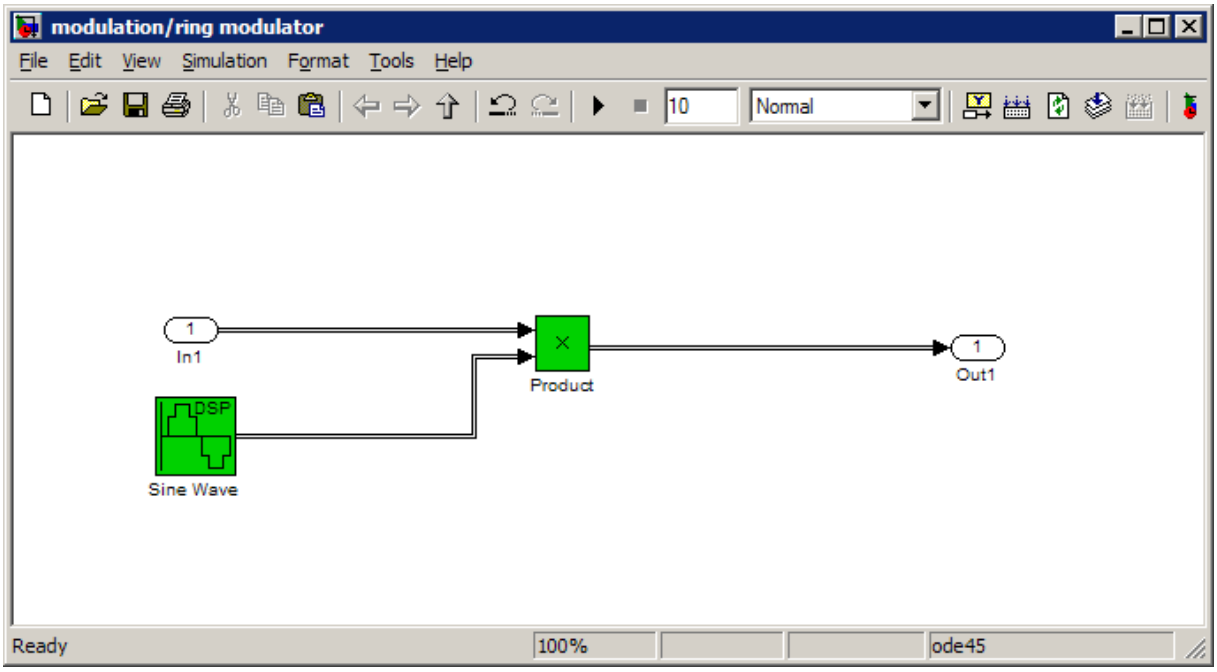


Slika 20: Sistem učinkov moduliranja v Simulinku

Sistem modulatorjev obsega samo dva modela, ki pa imata enaka izvor in ponor kot prejšnji sistemi, torej generator sinusnega signala in pa blok, ki izriše diagrame neobdelanega signala in obdelana signala.

6.2 Obročni modulator (*ring modulator*)

Zvočni signal x(n) pomnožimo s sinusnim signalom m(n), ki ima frekvenco f_c. Prvemu členu pravimo *modulator*, drugemu pa *nosilec*. V analogni domeni je to včasih predstavljalo težavo, toda znotraj računalnika je taka operacija navadno množenje. Če sta oba signala sinusna, s frekvencama f_c in f_x, potem kot rezultat slišimo vsoto in razliko frekvenc hkrati, torej dve frekvenci $f_1=f_c + f_x$ ter $f_2=f_c - f_x$. Če modulator sestavlja več frekvenc, se bo spekter zrcalno preslikal okoli frekvence nosilca, ki ga pri taki modulaciji ne slišimo [11].



Slika 21: Model ring modulator v Simulinku

Kot je razvidno iz slike, je izvedba izjemno enostavna: vhod pomnožimo s sinusnim signalom. Vseeno se je po uspešni simulaciji zgodila težava pri zagonu z zvočno datoteko. Tako sem ugotovil, da blok *to wave file* zahteva, da imata oba vhoda bloka *product* enako frekvenco vzorčenja (v mojem primeru 44,1 kHz) in enako število vzorcev na okvir (v mojem primeru 256). Do tega zaključka sem prišel šele, ko sem ustvaril zvočno datoteko signala, ki ga sicer ustvarja blok *DSP sine wave*, in na vhoda bloka za množenje pripeljal dve zvočni datoteki (to sem storil kot poskus, ker prvotna izpeljava s slike ni delovala). Simulink je ob napaki sicer izpisal opozorilo, vendar v tem primeru povsem neuporabno (enako opozorilo sem dobil že pri nekaterih modelih v prej opisanih družinah učinkov). Simulacija je uspevala, če je na vhod priključen le po en izvor, zato sem primerjal njune lastnosti in prišel do te ugotovitve. Tako sem v model lahko vrnil blok *DSP sine wave*, ki je veliko fleksibilnejši od dodatne zvočne datoteke, ki bi jo moral na novo ustvariti za vsako želeno spremembo frekvence modulatorja.

Učinek je torej uspešno realiziran, prav tako tudi preizkus z zvočno datoteko. Po različnih preizkusih lahko rečem, da je učinek uporaben za vokal (glasovi vesoljcev v filmih so v temelju modulirani, čeprav to ni edini učinek) in za inštrumente, vendar pri slednjih ne za »stalno«. Namreč gre za dokaj nemuzikalično popačenje, ki pa je po naravi nepredvidljivo, saj je taka navadno tudi slika nosilca. Pri uporabi z inštrumenti je učinek uporaben za krajše vložke, ki potem s svojo disonanco pripomorejo k napetosti izvedbe.

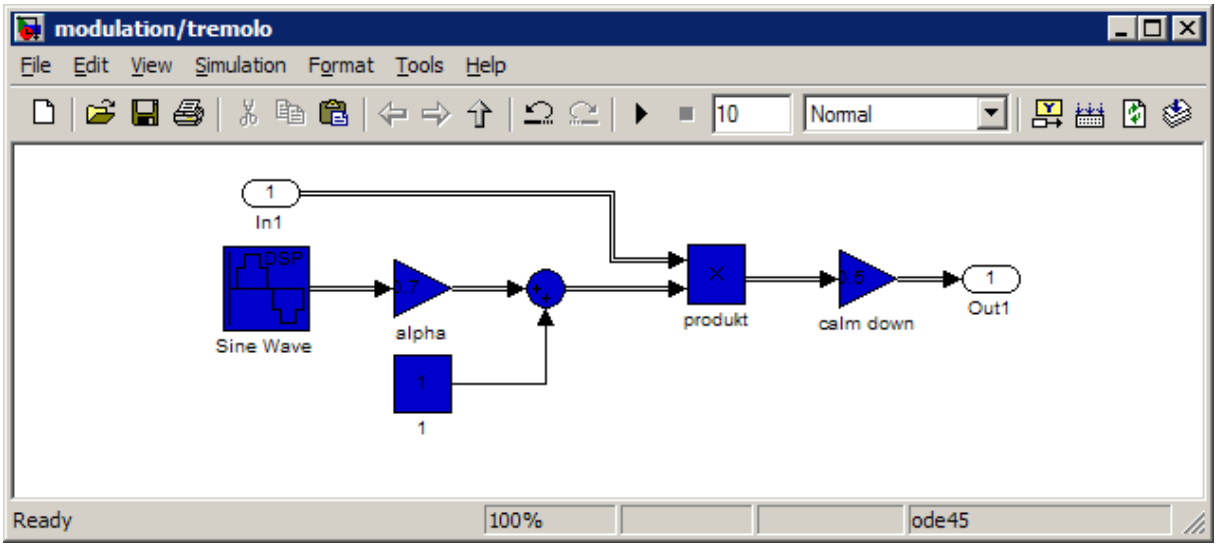
Še nekaj besed o frekvenci, s katero moduliramo – po preizkusih sodeč morata biti frekvenca modulatorja in nosilca čim bolj različni, če želimo npr. ohraniti razumljivost govora. Bolj kot se frekvenci bližata, bolj je signal popačen.

6.3 Amplitudni modulator

Tega je bilo v analogni elektroniki laže realizirati, zato je v uporabi veliko dlje časa. Implementiramo ga lahko s formulo

$$y(n) = (1 + a * m(n)) * x(n), \qquad (10),$$

kjer privzamemo, da je največja amplituda m(n) enaka 1. Koeficient a določa intenzivnost modulacije, ki je največja pri a = 1, učinek pa je onemogočen pri a = 0. Tipična shema vsebuje zvočni signal x(n) kot nosilec in nizkofrekvenčni oscilator kot modulator m(n). Amplituda izhodnega signala se spreminja glede na spreminjanje amplitude oscilatorja. Za razliko od obročnega modulatorja tukaj lahko slišimo tudi frekvenco nosilca, toda upoštevati moramo naš slušni sistem – če moduliramo s frekvencami pod 20 Hz, slišimo modulacijo v časovni domeni (spreminjanje amplitude – *tremolo*); če pa moduliramo z visokimi frekvencami, slišimo ločene spektralne komponente (razliko, nosilca in vsoto).



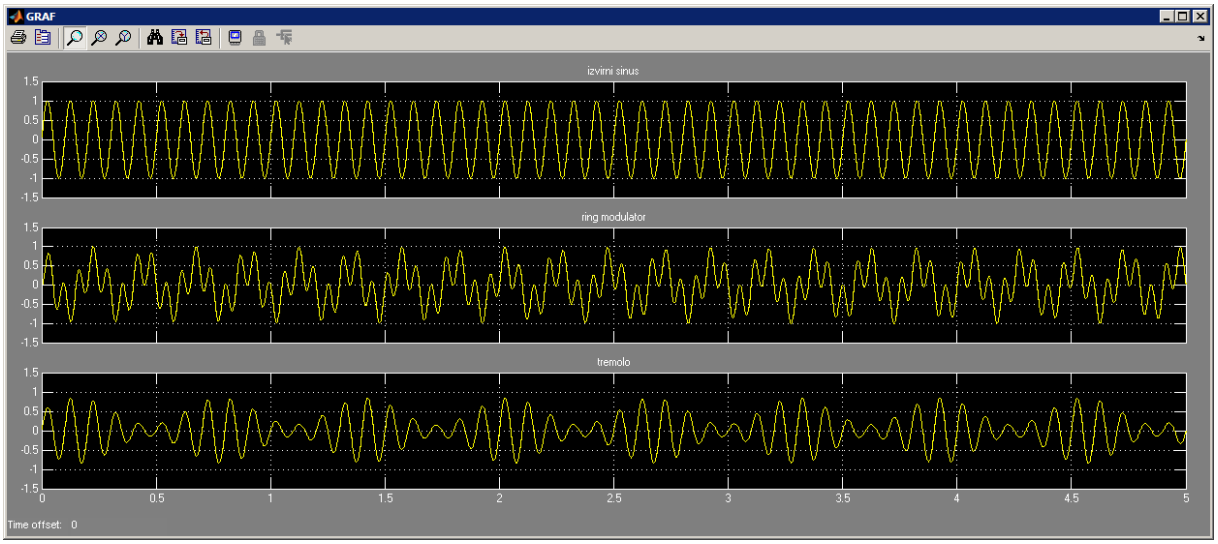
Slika 22: Model tremolo v Simulinku

Podobno kot prejšnji učinek družine modulatorjev se je tudi ta posebno dobro izkazal pri simulaciji in preizkusu z zvočno datoteko. Enako je tudi ta najprej uspešno prestal simulacijo, nato pa ni povzročal težav pri realnem preizkusu, kar je seveda razumljivo, saj se od prejšnjega modela razlikuje le po tem, da sinusni signal zdaj več ne niha med vrednostma –1 in 1, temveč med 0,3 in 1,7 – vse ostalo je praktično identično.

Če sem frekvenco bloka *DSP sine wave* postavil na pribl. 5 Hz, se je jasno slišal učinek *tremolo* (periodično spreminjanje amplitude) – intenzivnost učinka je mogoče uravnavati z blokom ojačenja *alpha*. Frekvence od 440 Hz do 1760 Hz so izvirni frekvenci dodale še modulirane (vsoto in razliko), kar je zanimivo in je zvoku dalo karakteristiko prostora oz. skoraj zborovski učinek. V primeru uporabe višjih frekvenc pa izvorni signal prevzame vlogo modulatorja, zvok pa se po popačenosti približa prej opisanemu učinku obročnega modulatorja (opisanega v poglavju 6.2).

6.4 Graf družine učinkov modulacij

Pri tej družini učinkov mi je bil graf še posebno v pomoč, saj je bilo zelo zanimivo in poučno opazovati spremembe parametrov. Poleg družine distorzij je verjetno tale družina učinkov najbolj »vidna« na grafu. Generator signala je ustvarjal sinusni signal s frekvenco 10 Hz in amplitudo 1, prikazan pa je na prvem diagramu. Drugi diagram prikazuje učinek obročnega modulatorja. Periodičnost je jasno vidna, ker sta oba signala čista sinusa. Pri tretjem diagramu (*tremolo*) je dobro viden učinek periodičnega spreminjanja amplitude.



Graf 4: Graf družine učinkov moduliranja

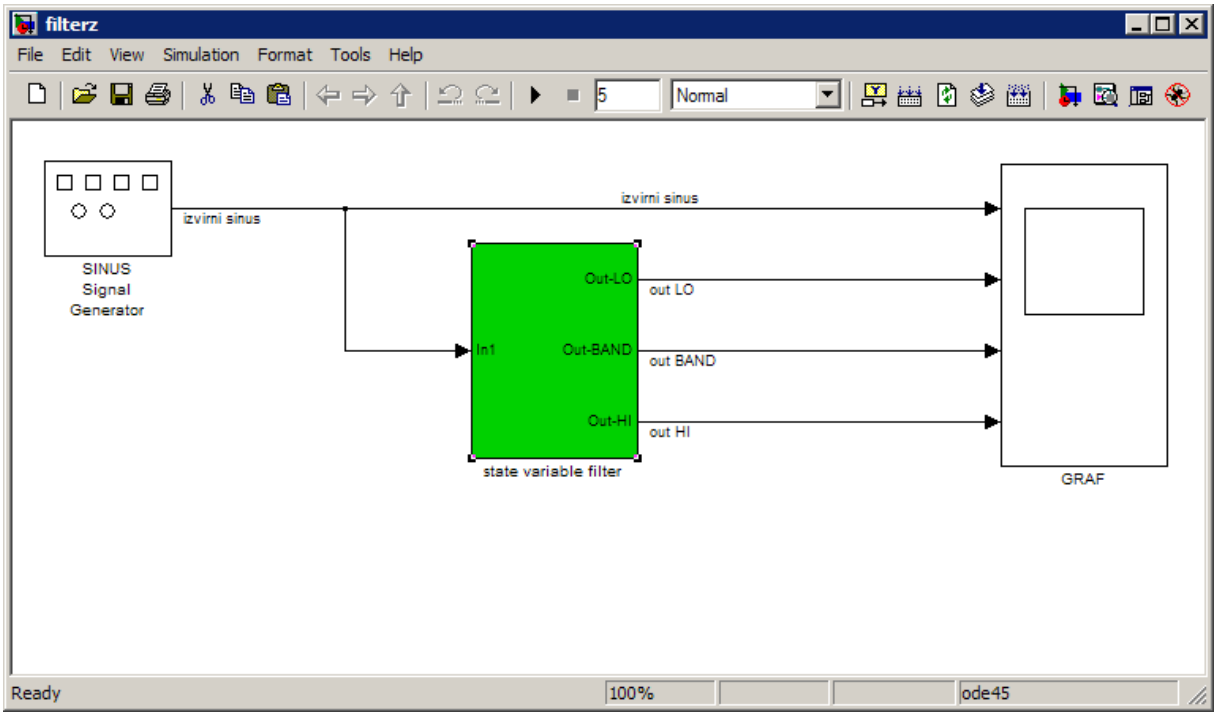
7 Učinki filtriranja

7.1 Opis filtrov in njihove rabe

Termin *filter* ima lahko veliko pomenov, v splošnem pa ga lahko razumemo kot način izbire, s katerim iz večje množice izločimo posamezne elemente z določenimi lastnostmi. Če to razlago uporabimo znotraj polja digitalnih zvočnih učinkov na signalu v frekvenčni domeni, lahko signal vidimo kot množico delov različnih frekvenc z različnimi amplitudami. Filter bo izbral dele glede na frekvence, ki jih želimo zavrniti, ohraniti oz. poudariti – z drugimi besedami: filter bo spremenil amplitudo teh delov glede na njihovo frekvenco [11].

Tak filter je obvezno linearen, saj kakršna koli nelinearnost povzroči prisotnost frekvenc, ki jih v izvornem signalu sploh ni (glej poglavje 4 o distorzijah).

Nizkoprepustni (*lowpass*) filtri analogno zgornjemu opisu izberejo frekvence, ki so nižje od mejne frekvence, ter zmanjšajo frekvence, višje od f_c. Z opisi načrtovanja tovrstnih filtrov se na tem mestu ne bomo ukvarjali, saj nas predvsem zanima uporaba filtrov v glasbene namene. Enako je izpuščena implementacija nizkoprepustnega filtra v Simulinku, saj ta že vsebuje različne bloke, s katerimi izbiramo implementacijo (od bilinearne transformacije do načrtovanj z okenskimi funkcijami) in določamo vse parametre.

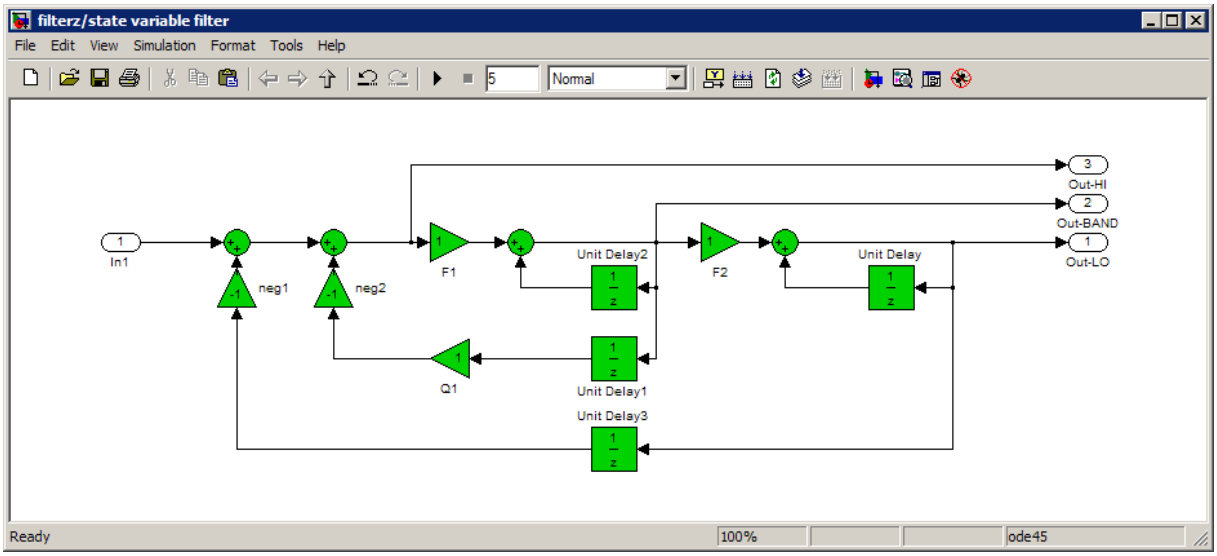


Slika 23: Sistem učinkov filtriranja v Simulinku

Pri tej družini nisem mogel ustvariti grafa, iz katerega bi bil dobro razviden njegov učinek – predvsem zato, ker tega ni mogoče zadovoljivo prikazati v časovni domeni. Poskusil sem z generiranjem vsote treh frekvenc (40 Hz, 400 Hz in 4000 Hz), na grafu pa je sicer jasno razviden učinek nizkoprepustnega filtra, ostali pa na prvi pogled niso razumljivi. Zato pa je zvočni učinek toliko zadovoljivejši.

7.2 State variable filter

Čeprav obstaja več različnih implementacij, ima vsaka svoje dobre in slabe strani, implementacijo pa moramo izbirati glede na namen. Za glasbene namene si želimo filter, pri katerem bi imeli neodvisen nadzor nad mejno frekvenco in dušilnim faktorjem – metoda, ki izvira iz tehnologije analognega računalništva, lahko reši to težavo. Gre za strukturo *state variable filter*, ki je sicer dražja od strukture *Sallen&Key* (nizkoprepustni filter drugega reda, realizacija analognega vezja zahteva najmanjše število komponent), vendar ima neodvisno uglaševanje komponent za mejno frekvenco in faktor dušenja (F in Q). Še več – hkrati imamo na voljo tri vrste izhodov: nizkoprepustnega, visokoprepustnega in pasovno prepustnega [11].



Slika 24: Model *state variable filter* v Simulinku

Kot je razvidno iz slike, so uporabljeni zelo enostavni elementi (seštevanje, množenje za koeficiente F_1, F_2 in Q_1 ter spreminjanje predznaka, zakasnitev). Ta struktura ni le posebno učinkovita z vidika samega filtriranja, temveč ima zelo enostavna razmerja med kontrolnimi parametri in koeficienti.

$$F_1 = F_2 = 2 * \sin\left(\pi * \frac{f_c}{f_s}\right) \qquad (11)$$

$$Q_1 = 2 * \zeta \qquad (12)$$

F_1 in F_2 iz enačbe (11) sta torej enaka (v [11] sta oba bloka označena enako, toda Simulink ne dovoljuje enakega poimenovanja dveh različnih blokov, zato sta bloka ojačenja pred tretjo in četrto vsoto različno poimenovana). f_c je mejna frekvenca, f_s je frekvenca vzorčenja, ζ iz enačbe (12) pa je faktor dušenja. Zelo pomemben podatek je, da velja formula (11) za računanje v radianih. Q_1 je realiziran kot blok ojačenja v drugi povratni povezavi. Na prikazani sliki imajo F_1, F_2 in Q_1 še napačne vrednosti, tj. 1.

Vseeno je treba posvetiti pozornost stabilnosti tega filtra, saj za višje mejne frekvence in večje faktorje dušenja postane nestabilen. Obstaja t. i. meja uporabnosti:

$$F_1 < 1 - Q_1. \qquad (13)$$

V večini primerov glasbenih aplikacij to ne predstavlja težave, saj so uglaševalne frekvence nizke v primerjavi z vzorčnimi (npr. f_c = 400 Hz, f_s = 44100 Hz), faktor dušenja pa je navadno

nizka vrednost (npr. 0,25). Ta filter se je izkazal kot zelo uporaben, med drugim tudi za gladke prehode med ekstremi nastavitev, strojno pa je implementiran tudi v sintetizatorjih, kjer je posebno uporaben ravno zaradi ločenega uglaševanja mejne frekvence in faktorja dušenja.

Poudaril bi še, da je simulacija z zvočno datoteko trajala zelo dolgo (skoraj dve uri za deset sekund zvočnega zapisa), zato tudi ni bila preveč primerna za primerjavo več različnih nastavitev.

7.3 Izenačevalniki

Za razliko od nizkoprepustnih filtrov, ki zmanjšujejo slišni spekter nad določeno frekvenco, izenačevalniki poudarjajo določene frekvenčne pasove, medtem ko ostali pasovi ostanejo nespremenjeni. Navadno so zgrajeni iz zaporedno povezanih filtrov *shelving* in *peak* prvega in drugega reda; te filtre lahko nadzorujemo neodvisno. Filtri *shelving* višajo/nižajo nizke ali visoke frekvenčne pasove s parametroma mejne frekvence f_c in ojačenja G. Filtri *peak* višajo/nižajo srednje frekvenčne pasove s parametri mejne frekvence f_c, pasovne širine f_b in ojačenja G. Pogosto uporabljen tip filtra je filter *peak* s konstantnim Q, kjer je Q razmerje med mejno frekvenco f_c in pasovno širino f_b. Tedaj se ob spreminjanju mejne frekvence sorazmerno povečuje tudi pasovna širina in obratno [11].

Tovrstni parametrični filtri se uporabljajo za parametrične izenačevalnike, oktavne izenačevalnike (npr. f_c = 31,25; 62,5; 125, 250, 500, 1000, 2000, 4000, 8000, 16000 Hz) in vse vrste izenačevalnih naprav v mešalnih konzolah, zunanji opremi, pedalih in vgradnih vezjih za inštrumente.

Obstajajo tabele za izračun omenjenih koeficientov, ki jih tu ne bom navajal, ker izenačevalnika niti nisem poskusil realizirati. Prejšnji model me je glede te družine povsem zadovoljil in prepričal, da bi tudi izenačevalnik deloval podobno dobro. Model izenačevalnika bi ustvaril, če bi ga imel priložnost preizkusiti tudi v realnem času. Tedaj bi zanj ustvari tudi primeren uporabniški vmesnik. Preizkušanje različnih nastavitev simulacije izenačevalnika se mi zdi preveč trivialno za raziskovalne namene, saj gre konec koncev za enega najbolj razširjenih učinkov, ki je v takšni ali drugačni obliki na voljo v praktično vseh napravah za predvajanje zvoka. Filtriranje je še vedno eno najbolj uporabljenih orodij pri snemanju in produkciji zvoka. Vsekakor pa je uspešna uporaba zelo odvisna od izkušenj in znanja človeka, ki jih uporablja.

8 Zaključek in možnosti razširitve

Čeprav so vsi opisani učinki že na voljo končnim uporabnikom v analogni oz. digitalni obliki, sem glavno motivacijo pri delu črpal iz želje po razumevanju delovanja posameznih učinkov in dojemanju odločitve za določeno implementacijo. Kar sem do sedaj le poslušal kot uporabnik, sem dobil priložnost tudi sam ustvariti in preizkušati različne možnosti.

Realiziral sem pet družin učinkov, za katere menim, da so najpogostejše oz. najpomembnejše. Znotraj vsake sem preizkusil bodisi različne implementacije enakega učinka, bodisi različne nastavitve ključnih parametrov, ki povzročijo, da je enaka realizacija slišati povsem drugače – pri tem je možno tudi, da končni izid prevara poslušalca in ta meni, da gre za dve popolnoma različni napravi oz. implementaciji.

Tako kot vsak projekt je tudi ta izdelavi utrpel nekaj kompromisov. S to trditvijo merim predvsem na širino in globino raziskovanja. Moral sem namreč omejiti število družin učinkov, saj je teh seveda veliko več, njihova izvedba pa nič manj zanimiva od teh, ki sem jih realiziral v diplomskem delu. Vseeno je končni odločitvi botrovalo dejstvo, da poskusim prikazati učinke, ki so zelo razširjeni in zato morda že od prej znani laičnem poslušalcu. Iz podobnega razloga sem moral omejiti število učinkov znotraj posamezne družine – le-to je namreč v obratnem sorazmerju s številom družin učinkov. Število opisanih družin učinkov bi zagotovo bilo manjše, če bi bilo možno uporabiti razvojno ploščico in stvari preizkusiti v resničnem času. Tedaj bi moral zmanjšati število družin učinkov na račun razvoja uporabniškega vmesnika, ki bi bil za realno uporabo nepogrešljiv.

Pri reševanju težav sem uporabil znanje, pridobljeno med študijem računalništva. Na praktičen način sem poglobil specifično polje računalništva, s katerim se večina študentov sreča le kot poglavjem na papirju. Samostojno sem se naučil uporabe novega delovnega okolja, čeprav tukaj učenja ni nikoli konec (kot pri vseh stvareh v življenju). Hkrati pa sem s to temo vseeno šel po manj utrjeni poti, saj obstaja zelo malo tovrstne literature. Večina ljudi se pri samogradnji odloča za analogno pot, ki je pogosto cenejša in dostopnejša širši množici (da ne omenjam dejstva, da so shematike za skoraj vsako tovrstno napravo na voljo na internetu). Digitalna realizacija je domena podjetij, ki svoje izdelke (tako strojno kot programsko opremo) ne prodajajo ravno poceni, zato so triki obrti poslovna skrivnost. Tako se s tem ukvarja le manjša množica akademskih navdušencev, ki ima na voljo znanje in včasih tudi potrebno strojno oz. programsko opremo.

Pri vsem tem pa so mi bile v neprecenljivo pomoč izkušnje, ki sem jih pridobil kot aktiven glasbenik – na odrih in v studijih sem namreč od 15. leta starosti, zadnja štiri leta pa sem dijak Konservatorija za glasbo in balet v Ljubljani. Udejanjanje v različnih zasedbah in projektih ter snemanje več plošč mi je dalo širok vpogled v različne načine uporabe učinkov, kar sem lahko srečno združil z zaključkom študija na Fakulteti za računalništvo in informatiko.

8.1 Kaj in kako naprej?

Vsaka opisana družina učinkov je po svoje porodila nekaj zamisli o prihodnjem razvoju, preden jih navedem, pa navajam možnosti razvoja projekta kot celote.

8.1.1 Uporaba v realnem času

Vsekakor najbolj pereča težava je, da se glasbeni učinki namreč že dolgo namreč uporabljajo le v realnem času. Izjeme so zelo redke, pa še te se pojavljajo, odkar so na voljo računalniške

rešitve za zajemanje in procesiranje zvoka. Redkost uporabe učinkov, ki niso na voljo v realnem času, je takšna, da se z njo nikoli ne bo srečal tudi marsikateri aktivni glasbenik. Kot je že omenjeno v uvodu (glej poglavje 0), se čaka na povezljivost orodij podjetja MathWorks in novejših razvojnih ploščic podjetja TI.

8.1.2 Uporabniški vmesnik, povezovanje učinkov, vrstni red učinkov

Kot sem opisal že prej, je uporabniški vmesnik nepogrešljiv del digitalne realizacije zvočnih učinkov. S tem bi najpomembnejše parametre vsakega modela lahko nastavljal brez posegov v model. Po pregledu ponudbe na tržišču bi bilo zanimivo narediti povzetek splošnih smernic pri ustvarjanju uporabniškega vmesnika. Zelo uporabno bi bilo nadzorovati učinek tudi kako drugače, npr. prek posebnih kontrolerjev (pedali, optični senzorji itd.).

Naslednji korak bi bil v smeri povezovanja učinkov. Zelo redko se namreč uporablja samo en učinek, zato bi bilo zelo zanimivo raziskati možnosti, ki se odpro s sočasno uporabo več učinkov: vpliv posameznih učinkov na naslednika v zaporedni vezavi in smiselnost vzporedne vezave s čistim signalom ter uravnavanje razmerij med dvema. To pa že odpre naslednjo zamisel.

Poleg neoviranega vklopa oz. izklopa učinka iz verige bi bila za implementacijo zelo zanimiva možnost fleksibilnega vrstnega reda učinkov – seveda preko uporabniškega vmesnika in v realnem času.

8.1.3 Druge družine učinkov

Kot sem že omenil, so opisane družine praktično le vrh ledene gore. Vsekakor bi v prihodnosti rad raziskal družino prostorskih učinkov (panorama, Dopplerjev učinek), poskus simulacije 3D–prostora s slušalkami in zvočniki, reverberacijo in ostale prostorske učinke.

8.1.4 Oktava gor, oktava dol

Pri družini učinkov oktaviranja bi najboljši izid učinka višje oktave poskusil izkoristiti za ustvarjanje učinka dveh oktav višje, ki ga tudi pogosto srečujemo. Veljalo bi dobro raziskati tudi učinek nižje oktave, za katerega sicer obstaja nekaj dovolj dobrih analognih rešitev, medtem ko se digitalne še borijo s slabim sledenjem. Tukaj opisano idejo s štetjem ničel pri pol- oz. polnovalnem usmerjanju bi bilo dobro bolje raziskati.

8.1.5 Distorziranje

Simulacijo elektronk bi poglobil v simulacijo celotnega ojačevalca, torej od obnašanja vhodne stopnje, prek tonskih kontrol do simulacije preobremenjene končne stopnje in morda tudi implementacije frekvenčne karakteristike zvočnika.

8.1.6 Zakasnjeni učinki

Najprej bi poskusil odpraviti pomanjkljivosti pri treh učinkih, ki vsebujejo blok s spremenljivo zakasnitvijo. Naslednji korak bi pa bil raziskovanje večpasovne implementacije učinkov zakasnitve. Zelo zanimive učinke lahko namreč dosežemo, če vhodni signal razdelimo na več pasov, te pa obdelamo z različno nastavljenimi zakasnitvami. Izhode tedaj primerno obtežimo in seštejemo.

8.1.7 Uporaba filtrov in spreminjanje parametrov v času

Filtri imajo več parametrov: ojačenje, mejno frekvenco in pasovno širino oz. faktor Q. Če te parametre spreminjamo odvisno od časa, dobimo posebne učinke.

Wah – najpogosteje ga srečamo kot pedal, ki nam omogoča nadzorovanje procesiranja z nogo. Signal gre skozi pasovno prepustni filter s spremenljivo mejno frekvenco in ozko pasovno širino. Ko dodajamo in odvzemamo »plin« na pedalu, spreminjamo mejno frekvenco, zaradi česar nastane učinek, podoben govorjenju *wah-wah*.

Druge implementacije:

- Nastavljivo število vzporedno vezanih pasovno prepustnih filtrov z nastavljivo mejno frekvenco, neodvisno od ostalih, ter nastavljivo hitrostjo preklapljanja med filtri. To je zanimivo kot poseben učinek v časovno zelo kratkem, toda izpostavljenem delu skladbe/nastopa; nikakor ni primeren za daljšo uporabo.
- Mejno frekvenco spreminjamo glede na amplitudo signala, t. i. sledilnik ovojnice (*envelope follower*) – gre za zelo muzikaličen efekt, ki lahko zelo poudari dinamiko igranja.
- Mejno frekvenco spreminjamo z nizkofrekvenčnim oscilatorjem, t. i. *auto-wah*, ki je v praktične namene zanimiv zaradi možne uglašenosti s tempom skladbe ali pa ravno nasprotno, torej neuglašenosti s tempom – odvisno od potreb in okusa.

9 Literatura in viri

[1] (2009) DSP Application. Dostopno na:

http://www.ti.com/sc/docs/general/dsp/programs/shareware/emph_app.htm

[2] (2009) Equal temperament. Dostopno na:

http://en.wikipedia.org/wiki/Twelve-tone_equal_temperament

[3] (2009) Frequencies and Ranges. Dostopno na:

http://www.contrabass.com/pages/frequency.html

[4] (2009) Fuzzbox. Dostopno na:

http://en.wikipedia.org/wiki/Fuzzbox

[5] (2009) Octave. Dostopno na:

http://en.wikipedia.org/wiki/Octave

[6] (2009) Octave effect. Dostopno na:

http://en.wikipedia.org/wiki/Octave_effect

[7] (2009) Piano. Dostopno na:

http://en.wikipedia.org/wiki/Piano

[8] (2009) Scicos: Block diagram modeler/simulator. Dostopno na:

http://www-rocq.inria.fr/scicos/

[9] (2009) Simulink - Simulation and Model-Based Design. Dostopno na:

http://www.mathworks.com/products/simulink/

[10] (2009) usb eZDSP C5505 And MATLAB. Dostopno na:

http://e2e.ti.com/support/dsp/tms320c5000_power-efficient_dsps/f/110/p/11103/43197.aspx#43197

[11] U. Zolzer, *DAFX – Digital Audio Effects*, John Wiley & Sons, 2002, pogl. 1–5.