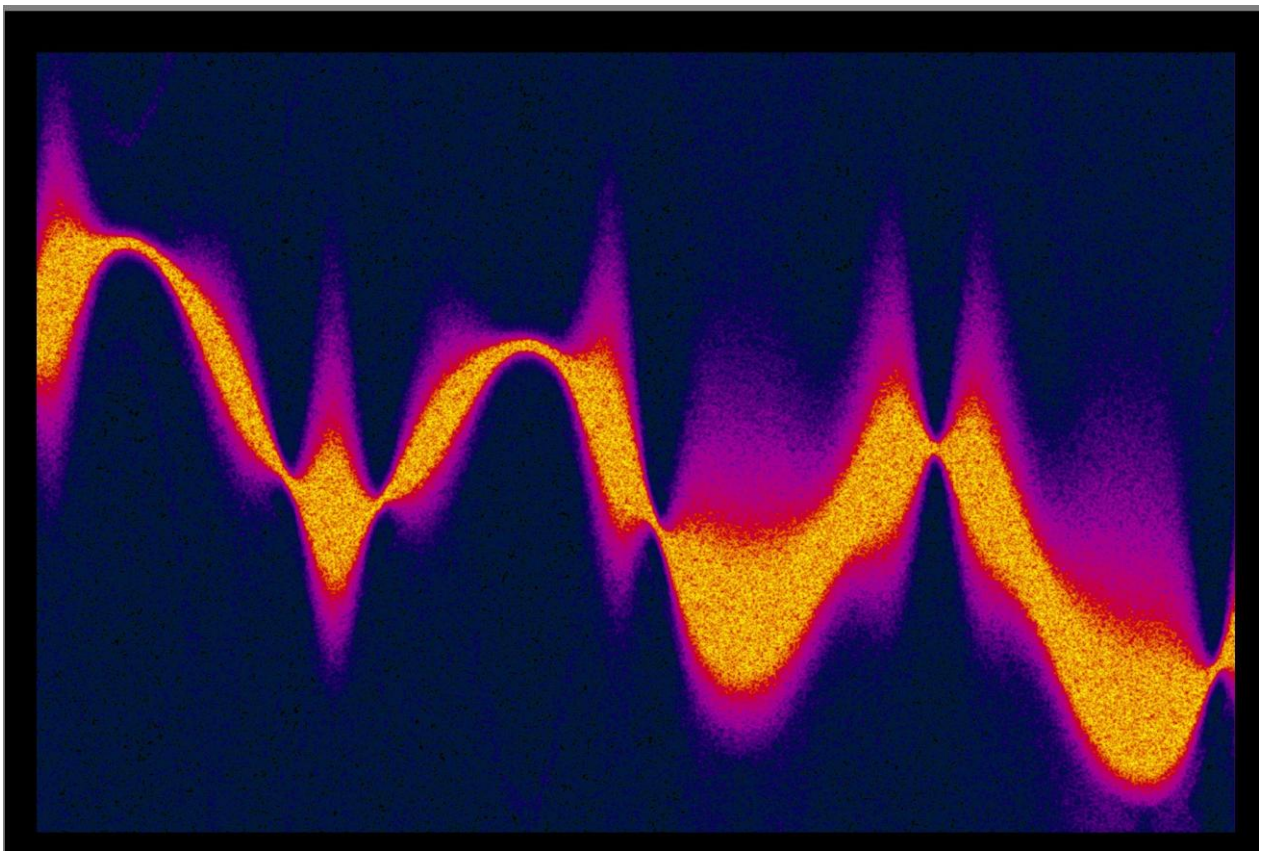
Sound Design



natuur
leven
technologie

Gecertificeerde
NLT module
voor vwo

Colofon

De module Sound Design is bestemd voor de lessen Natuur, Leven en Technologie (NLT). De module is op 17 juni 2010 gecertificeerd door de Stuurgroep NLT voor gebruik op het vwo in domein H, Materialen, proces- en productietechnologie. Het certificeringsnummer van de module is 4123-058-VH.

De originele gecertificeerde module is in pdf-formaat downloadbaar via ► http://www.betavak-nlt.nl.
Op deze website staat uitgelegd welke aanpassingen docenten aan de module mogen maken, voor gebruik in de les, zonder daardoor de certificering teniet te doen.

De module is gemaakt in opdracht van het Landelijk Ontwikkelpunt NLT.
Deze module is ontwikkeld door

- Rutger Teunissen, sonoloog
- Willem II, Tilburg, Teun van Heesch (natuurkunde), Sandra Smolders (wiskunde), Hans de Swart (wiskunde)
- Damstede, Amsterdam, Coen van der Kamp (wis-, natuurkunde). Peter Staats (muziek).

Aangepaste versies van deze module mogen alleen verspreid worden, indien in dit colofon vermeld wordt dat het een aangepaste versie betreft, onder vermelding van de naam van de auteur van de wijzigingen.

Materialen die leerlingen nodig hebben bij deze module zijn beschikbaar via het vaklokaal NLT:
►http://www.vaklokaal-nlt.nl/

© 2010. Versie 1.0
Het auteursrecht op de module berust bij Stichting Leerplan Ontwikkeling (SLO). SLO is derhalve de rechthebbende zoals bedoeld in de hieronder vermelde creative commons licentie.
De auteurs hebben bij de ontwikkeling van de module gebruik gemaakt van materiaal van derden en daarvoor toestemming verkregen. Bij het achterhalen en voldoen van de rechten op teksten, illustraties, enz. is de grootst mogelijke zorgvuldigheid betracht. Mochten er desondanks personen of instanties zijn die rechten menen te kunnen doen gelden op tekstgedeeltes, illustraties, enz. van een module, dan worden zij verzocht zich in verbinding te stellen met SLO.
De module is met zorg samengesteld en getest. Landelijk Ontwikkelpunt NLT, Stuurgroep NLT, SLO en auteurs aanvaarden geen enkele aansprakelijkheid voor onjuistheden en/of onvolledigheden in de module.

Ook aanvaarden Landelijk Ontwikkelpunt NLT, Stuurgroep NLT, SLO en auteurs geen enkele aansprakelijkheid voor enige schade, voortkomend uit (het gebruik van) deze module.
Voor deze module geldt een
Creative Commons Naamsvermelding-Niet-commercieel-Gelijk delen 3.0 Nederland Licentie
►http://creativecommons.org/licenses/by-nc-sa/3.0/nl

Foto voorpagina: sonogram, variatie op opgave 3 van toets 4.

Inhoudsopgave

1 Intro sound design 1
1.1 Sound design 1
1.2 Geluid bewerken 5
1.3 Geluid opwekken 8
1.4 Spraak- en zangsynthese 15
1.5 "Is er nog leven na MIDI?" 18
1.6 Sound design en NLT 22
Woordenlijst 29
2 Horen en zien 30
2.1 Inleiding 30
2.2 De hoofdpagina van WaveWizard 31
2.3 Periode, frequentie, amplitude 34
2.4 Scoop en Spectrum 40
2.5 Grondtoon, boventonen, harmonischen 44
2.6 Sonogram 46
Samenvatting 50
Woordenlijst 50
3 Sampling en de wiskundige representatie van digitaal geluid 51
3.1 Inleiding 51
3.2 Sampling 51
3.3 Werken met het commando-venster 58
Samenvatting 63
Woordenlijst 64
4 Versterken, mixen en echo's toevoegen 65
4.1 Versterken 65
4.2 Mixen 74
4.3 Echo 77
4.4 Echo-feedback 81
Samenvatting 86
Woordenlijst 88
5 Filters 89
5.1 De amplitudekarakteristiek A(f) 89
5.2 "Witte" geluiden 91
5.3 Laagdoorlaat- en banddoorlaatfilter 94
5.4 A(f) afgeleid uit differentievergelijking 95
Samenvatting 102
Woordenlijst 102
6 Sequencing en MIDI 104
6.1 Inleiding 104
6.2 Een reeks tonen 104

6.3 Stemming 109
6.4 Toongenerator aansturen met een MIDI-bestand 111
Samenvatting 116
Woordenlijst 117
7 Tonen waarvan de klankkleur verandert 118
7.1 Inleiding 118
7.2 De snaarplukvergelijking 118
7.3 Tijdvariante filters 125
Woordenlijst 128
Bijlage 1 URL-lijst 129

1 Intro sound design

Naast figuren vind je in dit hoofdstuk zo'n 30 kortere en langere geluidsvoorbeelden. Zonder die voorbeelden zul je de tekst waarschijnlijk totaal onbegrijpelijk vinden! Dus, zodra je een voorbeeld tegenkomt, luister er dan direct even naar vóórdat je verder leest. Zorg daarom dat je bij het doorwerken van de tekst de computer of een MP3-speler bij de hand hebt!
Geluidsvoorbeelden worden in de tekst aangegeven door dit symbool ► en zijn te vinden op vaklokaal NLT.

1.1 Sound design

Sound design is een creatief-technisch terrein dat zich bezig houdt met het ontwerp van geluid. Wat je je daar zo ongeveer bij moet voorstellen, illustreren we aan de hand van twee voorbeelden.

Figuur 1: Google Earth, "Waterlabyrint, Nijmegen".

Voorbeeld 1. Waterlabyrint
Aan de Waalkade in Nijmegen bevindt zich het Waterlabyrint (figuur 1: Google Earth, zoek op "Waterlabyrint, Nijmegen"). Het bestaat uit een ronde vijver met daarin een lang wandelpad over een stenen muurtje dat in vele cirkelvormige kronkels naar het centrum loopt. Als je daar, in het centrum, in je handen klapt, hoor je een bijzonder geluidseffect, nl. een extreem kort zoemtoontje: *"Bzzz..."*. (Luister naar ►H1_Waterlabyrint_Bzzz.mp3.)
Dat komt omdat elke cirkel een soort "akoestische lens" vormt die het geluid focust in het centrum. Omdat de muurtjes op onderling gelijke afstanden liggen, bereiken de reflecties van de handklap je oor met regelmatige tussenpozen, dus ***periodiek***. En dat hoor je als een ultrakort toontje.

Nu kan iemand zich afvragen: "als een handklap in het Waterlabyrint klinkt als een toontje, hoe zal een piano er dan klinken?"
Je kunt met de computer een piano-opname zo bewerken, dat het lijkt alsof het gaat om een opname van een piano die zich bevindt in het centrum van het Waterlabyrint! Luister achtereenvolgens naar ►H1_Piano_Menuet.mp3 en ►H1_Waterlabyrint_Piano_Menuet.mp3.

In plaats van een piano kun je ook opnames van andere geluidsbronnen, bijv. een stem of een ruisende branding, met dezelfde computerbewerking laten klinken alsof die geluidsbronnen zich in het centrum van het labyrint bevinden.
Die bewerking zou je kunnen noemen: ***het Waterlabyrint-effect***. In hoofdstuk 4 leer je echo's toevoegen aan een geluid; je kunt dan ook het effect van het Waterlabyrint zelf maken (zie de extra opdrachten op ►URL1).

Voorbeeld 2. Bedrijfsvideo
Op een varkensfokkerij is een bedrijfsvideo gemaakt. Daarop is een rondleiding te zien langs de stallen, de geautomatiseerde voederinstallatie, de biggetjesafdeling, en tot slot een kijkje in het laboratorium waar mensen in witte jassen in de weer zijn met reageerbuisjes. Deze afdeling is nodig voor de ondersteuning van kunstmatige inseminatie. Een microscoopopname laat beelden zien van zaadcellen die met hun staartjes heen en weer slaan en kronkelig door het beeld zwemmen.
Een ***componist / sound designer*** wordt ingehuurd om muziek en achtergrondgeluiden bij de beelden te maken. De directeur van het bedrijf heeft daarbij een bijzondere wens: hij wil graag die zaadcellen hóren zwemmen! Natuurlijk begrijpt hij wel dat je, zelfs wanneer je de allergevoeligste microfoon vlakbij de microscoop zou houden en het geluid maximaal zou versterken, helemaal niets hoort. Maar toch stáát hij erop dat er passend geluid gemaakt wordt bij die microscoopbeelden van zwemmende cellen.

1. Opdracht: sound designer
Neem eens aan dat jij die componist / sound designer uit voorbeeld 2 bent. Wat voor "passend" geluid stel jij je voor bij die microscoopbeelden (je vindt daar beelden van op YouTube)? Wat voor soort geluiden zou je maken als je alleen je stem mocht gebruiken als klankopwekker? (Dat laatste is iets waar sound designers meestal het eerst aan denken, bij een dergelijke opdracht!). Zie ook de slotopdracht 71.

Deze voorbeelden kun je op verschillende manieren van elkaar onderscheiden:

In voorbeeld 1 gaat het om een ***bewerking*** van een bestaand geluid, nl. dat van een piano; er wordt een effect aan toegevoegd. In voorbeeld 2 moet een geluid worden ***opgewekt***.
In voorbeeld 1 wordt gestreefd naar een zo goed mogelijke overeenstemming met de werkelijkheid; naar ***realisme*** dus. In voorbeeld 2 is sprake van ***geluidsfictie***: de sound designer zal een geluid moeten gaan verzinnen, want het gaat om iets dat nu eenmaal geen geluid maakt.

Sonificatie en visualisatie

Er zijn heel veel voorbeelden van gevallen waarin een sound designer wordt gevraagd geluiden te bedenken bij dingen die helemaal geen geluid maken, zoals in voorbeeld 2. Dat wordt soms wel aangeduid met ***sonificatie***, als tegenhanger van ***visualisatie***.
Sonificatie wordt vaak toegepast om het bedienen van apparaten en software gemakkelijker te maken. Het is dan de bedoeling dat de gebruiker door gemakkelijk herkenbare geluidjes op iets attent wordt gemaakt. Bijvoorbeeld dat het papier van de printer op is, of dat je eerst een venster moet sluiten voordat je een andere taak kunt laten uitvoeren, of dat je vergeten bent om een veld op een formulier in te vullen.

Sonificatie van wiskundige objecten

Stel je voor dat de directeur van dat bedrijf uit voorbeeld 2 niet één, maar drie sound designers vraagt om een geluid te maken bij de microscoopbeelden. Waarschijnlijk krijgt hij dan ook drie verschillende sonificaties te horen, want de sound designers zullen zich alle drie iets anders voorstellen bij de beelden.
Maar er bestaan ook objecten die geen geluid maken, maar waarvan de sonificatie altijd hetzelfde is en volkomen onafhankelijk van sound designers. Die zeer bijzondere klasse van objecten bestaat uit wiskundige formules!
Je leerde bij wiskunde dat je sommige wiskundige objecten, bijvoorbeeld functies, kunt visualiseren. In het geval van een functie bestaat zo'n visualisatie uit een grafiek. Maar je kunt (dankzij de computer) een functie ook hoorbaar maken - vanaf hoofdstuk 4 ga je dat ook zelf doen. Net zoals er bij elke functie slechts één grafiek hoort, en niet twee of drie, hoort er bij elke functie ook slechts één geluid! De klank ervan wordt uitsluitend bepaald door de definitie van de functie en niet door wat een of andere sound designer toevallig in zijn hoofd heeft!

Voorbeeld.
Je hebt bij wiskunde kennis gemaakt met de sinusfunctie. Daar hoort een grafiek bij die je al heel vaak hebt gezien: een fraaie kromme (sinus

betekent kromming, bocht). De sonificatie van de sinusfunctie heet simpelweg ***sinustoon*** en staat bekend als de meest pure toon. Als je eenmaal weet hoe een sinustoon klinkt, zul je het geluid gemakkelijk herkennen. Waarschijnlijk heb je bij natuurkunde al kennis gemaakt met de ***toongenerator*** - een elektronisch apparaat waarmee je tonen kunt opwekken, ook sinustonen. Luister naar ►H1_Sinustonen.mp3. Je hoort hier sinustonen, die elk 5 seconde duren, met toonhoogten resp. 100, 300, 1000, 3000 en 9000 Hz. (afkorting Hz: zie paragraaf 2.3).

De sinustoon is trouwens een toon met zeer bijzondere eigenschappen. Een van de belangrijkste stellingen uit de theorie over trillingen is dat je elk geluid kunt (na)maken door een aantal sinustonen met verschillende toonhoogten en verschillende volumes gelijktijdig te laten klinken! Denk daar eens aan als je glaswerk hoort rinkelen of je natuurkundedocent hoort praten of een fietsband hoort leeglopen. Allemaal mixen van sinustonen! Een geluidsvoorbeeld vind je in paragraaf 1.2 bij **a**, waarin een stemgeluid (dat eerst is omgezet in zo'n mix van sinustonen) eens flink door de gehaktmolen wordt gehaald. In opdracht 35 en 36 ga je complexe golfvormen maken door sinustonen te mixen.

Differentiëren werkt verhelderend!

Ook heb je bij wiskunde kennis gemaakt met een bijzondere bewerking van een functie namelijk ***differentiëren***: het bepalen van de afgeleide. Van die afgeleide functie kun je ook weer een grafiek maken en de grafiek van de afgeleide heeft een duidelijke relatie met de grafiek van de oorspronkelijke functie.
Als elke functie een eigen, unieke sonificatie heeft, dan geldt dat uiteraard ook voor de afgeleide. Is er nu ook sprake van een duidelijke relatie tussen de sonificatie van een functie en die van haar afgeleide, net zoals bij die grafieken?
Dat blijkt inderdaad het geval! De bewerking differentiëren heeft een duidelijk herkenbaar effect op de klank: die wordt namelijk altijd helderder! Je weet dat je een functie vaak meerdere keren, of zelfs oneindig vaak, kunt differentiëren. Dat betekent dat de sonificaties daarvan steeds weer iets helderder klinken! Dus je mag stellen: differentiëren werkt letterlijk verhelderend!
Geluidsbewerkingen die de ***klankkleur*** van een geluid veranderen (bijvoorbeeld helderder of doffer maken) noemen we ***filters***.
Differentiëren kun je dus opvatten als een filterwerking.
Luister naar ►H1_Gedifferentieerde.mp3. Je hoort twee tonen die elk 5 seconden duren en na elkaar klinken, met een pauze ertussen van 2 seconden. De eerste toon is de sonificatie van een bepaalde functie, de tweede die van de afgeleide van die functie. Let op het klankverschil.

Meer hierover in hoofdstuk 4 en 5. Ook verderop in dit hoofdstuk komen we nog uitvoerig terug op filters, want ze zijn veruit de belangrijkste bewerking die je op geluid kunt toepassen.

Sound design doe je met de computer

Sound design heeft betrekking op elke vorm van manipulatie met geluid. Sound design producten kom je overal tegen; in speelfilms, in software (vooral games!), in multimediale producties, zoals de bovengenoemde bedrijfspresentatie, in radio- en tv-programma's, en op internetsites.
In deze module zullen we muziek beschouwen als een heel bijzondere en tevens de meest fundamentele vorm van sound design; meer hierover vanaf paragraaf 1.3.
Sound design producten worden tegenwoordig bijna uitsluitend gemaakt met de computer: er bestaat geen productiemethode die flexibeler en krachtiger is. Elke vorm van sound design kun je uitvoeren op je eigen PC. Je kunt gerust stellen: sound design doe je met de computer.

We zullen in de rest van deze module zien dat alle bewerkingen die je bij sound design verricht niets anders zijn dan ***sonificaties van wiskundige objecten en wiskundige bewerkingen***. Je hebt de computer niet alleen nodig om er de bijbehorende berekeningen mee uit te voeren, maar ook om het klinkende resultaat hoorbaar te maken.

1.2 Geluid bewerken

We zagen zojuist dat het in sound design in principe om twee dingen gaat:

- (bestaande) geluiden bewerken
- (nieuwe) geluiden opwekken.

In deze paragraaf geven we voorbeelden van het eerste: bewerkingen van geluid.

(a) Vrouwenstem verandert in mannenstem
►H1_Kyma_stemvervorming.mp3
Je hoort de stem van Carla Scaletti; zij ontwierp samen met Kurt Hebel de wereldwijd gebruikte sound design software Kyma. Ze begint met te vertellen dat ze haar stemgeluid eerst heeft omgezet in een mix van 128 sinustonen, waar ze vervolgens flink mee aan de slag gaat.
Ze verandert bijvoorbeeld al pratend haar stem ook even in een mannenstem. In hoofdstuk 5 ga je zelf spraak vervormen (opdracht 50d). Eén van de effecten die je hier hoorde, zul je dan gemakkelijk herkennen en kunnen namaken.

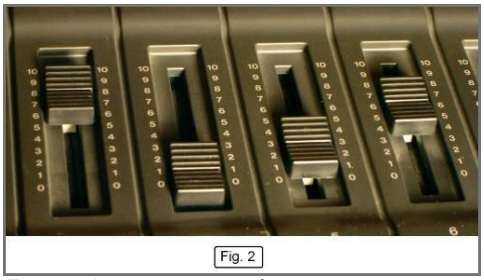
2. Opdracht: Carla-Kyma

a. Probeer eens te noteren wat je Carla in dit geluidsfragment hoort zeggen.
b. Dat zal je waarschijnlijk niet helemaal lukken, want één fragment is niet verstaanbaar. Op welk tijdstippen begint en eindigt dat fragment?

(b) Morphing: kat-baby, Harleeuw, mammie-voetballer

►H1_Kyma_morphing.mp3

In voorbeeld (a) werd één geluid aan een aantal verschillende bewerkingen onderworpen. Maar je kunt ook twee geluiden samen bewerken en ze op een heel natuurlijk klinkende manier geleidelijk laten overvloeien, zodat er sprake lijkt van een echte gedaanteverwisseling. Deze techniek staat bekend als ***morphing***. (Het morphing-effect ken je misschien van afbeeldingen. Op de Engelstalige Wikipedia zie je een morphing van twee gezichten: dat van ex-president George Bush jr. naar dat van gouverneur / acteur Arnold Schwarzenegger.)



Figuur 2: mengpaneel.

Let wel: morphing is niet hetzelfde als ***cross fading***, waarbij het ene geluid langzaam zachter en het andere langzaam harder wordt. Cross fading kun je bijvoorbeeld doen met een mengpaneel (figuur 2), waarbij je één volumeregelaar (fader) langzaam naar achteren, en tegelijkertijd de andere langzaam naar voren schuift. Zie verder hoofdstuk 4.

Het verschil tussen morphing en cross fading is goed uit te leggen aan de hand van de overgang van een lage toon naar een hoge toon. Bij cross fading hoor je ze tijdens de overgang steeds allebei, alleen wordt de ene langzaam harder en de andere langzaam zachter. Bij morphing van een lage naar een hoge toon hoor je steeds maar één toon die langzaam hoger wordt.

Ook dit morphing voorbeeld is gemaakt met Kyma. Je hoort eerst de stem van Carla overvloeien in die van Kurt. Dan het miauwen van een kat, dat overgaat in babygehuil. Vervolgens verandert een didgeridoo (blaasinstrument van Aboriginals) in een mondharp, een Harley-Davidson motor in een leeuw, een drumstel in een klok.

Daarna hoor je een voorbeeld dat afkomstig is van een Engelse radioreclame voor chips: een bekende voetballer probeert, vermomd als moeder, een jochie een zakje chips af te troggelen. Als de peuter geen krimp geeft, valt de gefrustreerde voetballer lelijk uit z'n moederrol, morft terug naar zichzelf en grist scheldend het zakje weg.

(c) Vocoding mouth tube: "Feel all right?"; snurkend standbeeld

Het komt heel vaak voor dat een geluid zijn typerende klank ontleent aan de ruimte waarin het klinkt, bijvoorbeeld zang klinkt in een kerk heel anders dan in een zandbak.

Je stem zelf is trouwens een nog veel mooier voorbeeld van een geluid in een ruimte. Je stembanden bevinden zich onderin je spraakkanaal (mond- en keelholte); je hoort nooit je stembanden op zichzelf, maar steeds zoals ze klinken in het spraakkanaal.
Het spraakkanaal maakt weliswaar zelf geen geluid, maar zonder mond- en keelholte kun je niet articuleren en dus niet spreken. Bij spraak draait het dus volledig om de ruimte waarin de stembanden klinken! Wil dat zeggen dat je je stembanden niet per se nodig hebt om te kunnen praten? Inderdaad, want als je fluistert maak je geen gebruik van je stembanden, maar je bent wel verstaanbaar.

Sommige mensen moeten hun stembanden missen en hebben een klein luidsprekertje in hun keel dat tonen maakt zoals je stembanden doen. Datzelfde effect kun je ook langs een minder drastische weg bereiken. Je verbindt een klein luidsprekertje met een trechter waar een slangetje aan vast zit. Het andere uiteinde van het slangetje stop je losjes in je mond, zodat het luidsprekergeluid hoorbaar wordt via je mond. Als je nu praatbewegingen maakt dan wordt het luidsprekergeluid vervormd tot verstaanbare spraak.

Deze techniek, de ‘mouth tube’ of ‘talk box’, wordt in de popmuziek al zo'n 40 jaar gebruikt en staat in de telecommunicatie bekend als ***vocoding***. De grap is dat je je gitaar letterlijk kunt laten praten! Daarbij stuur je de tonen van je gitaar naar het luidsprekertje, neemt het slangetje in je mondhoek en zegt een paar woorden.
In het volgende voorbeeld van een live concertregistratie, zingt de gitaar het publiek toe en vraagt: do you feel all right? Vervolgens dezelfde vraag een octaaf lager.
►H1_Mouth_tube.mp3

Met de computer kun je ook vocoding doen; je hebt zelfs veel meer flexibiliteit en ook een veel betere geluidskwaliteit dan met de mouth tube. Voor een aflevering van de Muppet show moest het geluid gemaakt worden van een snurkend standbeeld. De sound designer maakte twee geluidsopnamen: (1) rollende rotsblokken, (2) snurkende man. Van dat snurkgeluid werd door de vocoder software een soort "bewegende ruimte" gemaakt, waarin de rollende rotsblokken vervolgens kunnen klinken. Dat geeft het effect alsof de snurker een aantal rotsblokken in z'n luchtwegen heeft zitten, die heftiger beginnen te rollen naarmate je dieper in- of uitademt. Zo gaan de rollende rotsblokken over in rochelende of snurkende rotsblokken.

Vocoding is heel iets anders dan een mix van twee geluiden! Bij een ***mix*** hoor je twee of meerdere geluiden tegelijk. Bijvoorbeeld "kantine-"geluid is een mix van stemmen, serviesgoed, koffieautomaat, airco etc. Vocoding

daarentegen zou je kunnen omschrijven als: geluid dat klinkt in een ruimte waarvan de wanden kunnen bewegen. Dat klinkt ingewikkelder dan het is. Het meest alledaagse voorbeeld van vocoding is spraak: je stembanden trillen in je spraakkanaal, waarvan de wanden voortdurend bewegen terwijl je spreekt. Het woord vocoding verwijst daar ook naar: het is een samentrekking van voice encoding.
Vooral ten behoeve van telecommunicatie zijn er in de loop van de tijd verschillende effectieve vocoder-technieken ontwikkeld. De zin daarvan is dat een communicatiekanaal daardoor veel meer capaciteit krijgt, want een bijzonder bijeffect van de vocoder is dat het een sterke datacompressie mogelijk maakt.
Hoe je geluiden mixt, lees je in hoofdstuk 4.

►H1_Vocoding_snurkend_standbeeld.mp3

- Meer over Vocoding en de mouth tube lees je op ►URL2.
- Op Wikipedia vind je een artikel over de mouth tube onder de naam talk box.

(d) Tijd-strek, toonhoogteverschuiving en "Donald Duck-effect"
Misschien heb je wel eens op een ouderwetse grammofoon een langspeelplaat op een te hoog toerental afgespeeld. Er gebeuren dan drie dingen: de afspeeltijd wordt korter, de tonen worden hoger en klankkleuren "verschuiven", d.w.z. dat bijvoorbeeld een stem "Donald Duck-achtig" gaat klinken (zoals duikers die helium in hun longen hebben). De technische term voor dat laatste is ***formantverschuiving***.
Deze drie effecten treden altijd samen op. Het blijkt een hele toer om ze juist afzonderlijk te realiseren, bijvoorbeeld wel hoger, maar niet sneller en geen Donald Duck, of niet sneller, niet hoger maar wel Donald Duck. Met de computer zijn ze wèl te scheiden, zelfs via heel verschillende technieken, en dat levert allerlei interessante geluiden op. In paragraaf 1.4, voorbeeld (a) hoor je spraak geleidelijk sneller en hoger worden, zonder dat het Donald Duck-effect optreedt.

1.3 Geluid opwekken

In paragraaf 1.2 hebben we een aantal voorbeelden besproken van geluidsbewerkingen. Maar we hebben ook al gezien dat je bij sound design vaak een klank "uit het niets" opbouwt, zoals het "geluid" van zwemmende zaadcellen. Dat noemen we opwekken, genereren of ***synthetiseren***.
Daarvan is vooral sprake bij een van de meest boeiende, uitdagende en veelzijdige toepassingsgebieden van sound design: muziek!
Muziek wordt wel eens kort omschreven als ***organised sound*** (bijvoorbeeld door componist John Cage), en dat ligt natuurlijk niet zo ver af van sound design. Dat je met behulp van elektronische middelen heel goed muzikale

klanken kunt opwekken, is al lang bekend; denk aan de elektrische gitaar en het elektronisch orgel die al ruim een halve eeuw bestaan.

Synthesizer

In de popmuziek wordt sinds de jaren 60 veel gebruik gemaakt van een heel bijzonder elektronisch orgel met kolossaal veel instelmogelijkheden. Daar kun je geluiden mee maken die nog nooit eerder in de muziek werden gehoord. Een dergelijk elektronisch "superorgel" heet ***synthesizer***.
Het Griekse woord syn-these betekent letterlijk: samen-stelling. In het Latijn: com-positie. Dus de woorden componist en synthesizer betekenen precies hetzelfde: samensteller. In de muziek hebben een componist en een synthesizer alleen maar een verschillend werkterrein: een componist "stelt noten samen", en een synthesizer stelt de klanken van die noten samen.
Het leuke van een synthesizer is dat je er eindeloos mee kunt experimenteren en dat je er echt totaal waanzinnige klanken mee kunt maken. Toch is dat laatste niet voor elke muzikant even belangrijk. Vandaar dat er sinds de jaren 90 een vereenvoudigde variant van op de markt is verschenen: het ***keyboard***. Met een keyboard kun je niet zelf klanken samenstellen, wel kun je kiezen uit enkele honderden voorgeprogrammeerde klanken, die vooral voor (pop)muziek interessant zijn.

Om een indruk te krijgen van de klankmogelijkheden van een synthesizer, geven we een vijftal voorbeelden (a - f):

(a) Toeteren
►H1_Traffic_jam.mp3
Dit nummer, Traffic jam, begint met getoeter - niet geheel onbegrijpelijk, gelet op de titel. Maar het geluid is geen opname van echte claxons. Het wordt live, ter plekke op het podium, gemaakt door de toetsenman, op een synthesizer.

3. Vraag
Waaruit blijkt dat dit nooit echte claxons kunnen zijn?

Merk op dat het geluid van een claxon een toon is. Bij de synthesizer is de toon een centraal begrip, want onze muziek is vooral een muziek "met tonen" (in tegenstelling tot bijvoorbeeld sommige Afrikaanse muziek die meer is georiënteerd op ritme en ***slagklanken*** (slagklanken: denk aan allerlei ***percussie-instrumenten*** zoals trommels, de marimba, de xylofoon, de shaker, het wood block). We zullen verderop in dit hoofdstuk dan ook zien dat de toongenerator (die je al leerde kennen bij natuurkunde, zie paragraaf 1.1) een onmisbaar onderdeel is in elke synthesizer.

(b) Vette, coole synthi-klanken

►H1_Vet_en_cool_Synth.mp3

Zoals je in dit voorbeeld hoort, kunnen de klanken van een synthesizer heel grappig zijn en ook veel extremer dan die van gewone muziekinstrumenten. Het geluid dat je hier hoort is, in tegenstelling tot het geluid in het eerste voorbeeld, niet afkomstig van een echte, "fysieke" synthesizer, maar van een computer die het geluid van een synthesizer zo goed nadoet dat je geen enkel verschil met het werkelijke instrument kunt horen. Daarom maken we voortaan, als we het hebben over een synthesizer, geen onderscheid tussen het echte apparaat en de computersimulatie ervan.

(c) Zwemmende zaadcellen

Luister naar ►H1_Zwemzaad.mp3. Hier hoor je een sonificatie van de microscoopbeelden van paragraaf 1.1, voorbeeld 2. Ook dit geluid is gemaakt met de computer en wel met de educatieve audio-software ***WaveWizard***, waarmee je vanaf hoofdstuk 2 zelf intensief aan het werk gaat.

(d) Appassionata

Met de computer kun je een synthesizer dus gemakkelijk nadoen. Maar met het nabootsen van echte, akoestische muziekinstrumenten, zoals een piano of een viool, ligt het minder eenvoudig. De geluiden die een synthesizer opwekt, mogen dan wel veel extremere klankmogelijkheden hebben dan de instrumenten die we al eeuwen in de muziek gebruiken, maar niettemin zijn ze, heel anders dan je misschien geneigd bent te denken, veel simpeler van opbouw. Echte muziekinstrumenten daarentegen wekken uiterst ingewikkelde, "onnavolgbare" trillingspatronen op. Juist daarom klinken ze zo natuurlijk en expressief. Je bent daar zo aan gewend dat je de geluiden van een synthesizer vaak een beetje primitief vindt. Daarom is het een van de grootste uitdagingen van sound design om computerklanken net zo natuurlijk, rijk, complex en "gewoon" te laten klinken als akoestische instrumenten. Daarvoor is het nodig dat je instrumenten heel goed bestudeert en intensief onderzoek doet naar de manier waarop een instrument nu precies een klank opwekt.

►H1_Beethoven_Bank.mp3. In dit voorbeeld hoor je het derde deel van de "Appassionata", een van de bekendste pianosonates van Beethoven.

Natuurkundig model van een muziekinstrument

Zeker als je enkele malen echt heel goed gaat luisteren, begint het je op te vallen dat het hier en daar een beetje vreemd klinkt. Dat is niet zo verwonderlijk, want het geluid dat je hier hoort, is helemaal niet afkomstig van een piano! Je luisterde naar wat we aan het eind van paragraaf 1.1 "sonificaties van wiskundige objecten en wiskundige

bewerkingen" noemden! Die objecten en bewerkingen beschrijven elk het trillingsgedrag van een bepaald onderdeel van de piano (bijvoorbeeld de snaren, de hamers, het klankbord, etc.) en ook van de manier waarop die onderdelen op elkaar reageren. Zo'n samenhangend geheel van bewerkingen waarin verschillende onderdelen van een muziekinstrument en de wisselwerkingen ervan met andere onderdelen wordt beschreven, noemen we een ***natuurkundig model*** van dat instrument. Hoe beter en gedetailleerder het model, des te realistischer de klank.
Het voorbeeld dat je hier hoorde, is het resultaat van een promotieonderzoek aan de Technische Universiteit van Helsinki. Het proefschrift en ook nog een aantal andere geluidsvoorbeelden vind je op ►URL3.

4. Opdracht: computerpiano versus echte piano
Vergelijk deze computerpiano met een echte piano waarop hetzelfde stuk wordt gespeeld, ►H1_Beethoven_Biss.mp3. Probeer aan te geven wat, volgens jou, de sterke kanten en wat de zwakke kanten zijn van deze computerbenadering van de piano.

(e) Sampler-piano
Een vraag die misschien bij je is opgekomen terwijl je luisterde naar die computerpiano is: "Waarom zo moeilijk doen met natuurkundige modellen en sonificaties van ingewikkelde formules! Het kan allemaal toch veel simpeler en praktischer? Je pakt gewoon een microfoon en maakt een afzonderlijke opname van elke toets van een echte piano. De opnames van al die toetsen sla je digitaal op. Je bouwt een digitaal apparaat met toetsen zoals op een piano of een keyboard en een geluidsgeheugen waar al die opnames in zitten. Dat apparaat zorgt ervoor dat bij elke toets die je aanslaat de corresponderende geluidsopname weer wordt afgespeeld. Als je meerdere toetsen tegelijk aanslaat, dan worden er simpelweg ook meerdere opnames tegelijk afgespeeld (gemixt dus). Daarmee beschik je over de meest natuurgetrouwe, realistische weergave van een piano!"

Dat is inderdaad een heel voor de hand liggend en goed uitvoerbaar idee. En het bijbehorende apparaat bestaat dan ook al zo'n 25 jaar! Het heet ***sampler*** en geeft muzikaal goed bruikbare klanken. De sampler is zelfs zo succesvol dat je er in afgeslankte vorm een aantreft op elke computer-geluidskaart! Daar zitten natuurlijk geen echte pianotoetsen op, maar alleen een soort interne, "elektronische" toetsen. Je kunt er ook niet zelf opnames mee maken, maar je kunt wel gebruik maken van opnames die op de geheugenchips van de geluidskaart al zijn opgeslagen.

In het volgende voorbeeld hoor je, ditmaal voor de derde keer, dat pianostuk, gespeeld door de sampler van je eigen geluidskaart. Elke toon die je hoort, is dus een opname van een echte pianotoon!
►H1_Beethoven_Sampler.mid.

(f) Orkest-in-blik

Niet alleen van de piano, maar ook van elk ander muziekinstrument kun je alle tonen afzonderlijk opnemen en in een sampler stoppen. De geluidskaart beschikt over een groot geheugen waar al die opnames in zitten. De sampler kan ook de geluiden van een heleboel verschillende instrumenten tegelijk afspelen. Zo kun je met een sampler ook het geluid van een heel orkest nadoen!

Hier heb je twee voorbeelden van instrument-ensembles afkomstig van de sampler op je geluidskaart:
►H1_West_Side_Story_MIDI.mid
►H1_Pink_Panther_MIDI.mid

Vergelijk ze eens met opnames van de echte ensembles:
►H1_West_Side_Story.mp3
►H1_Pink_Panther.mp3

Het resultaat is, eerlijk gezegd, nogal onthutsend! Tegen alle verwachting in hoor je dat de verschillen tussen de echte instrumenten en de sampleropnames toch wel heel erg groot zijn! De samplermuziek (.mid) klinkt best aardig, totdat je de echte opnames (.mp3) hoort. Daarmee vergeleken klinken de samplergeluiden ineens een beetje steriel, miezerig en ingeblikt. Onwillekeurig denk je bij het horen van die samplermuziek aan plastic rozen of dennengeur uit een doosje. Maar hoe kan dat nou!? Het zijn toch echte geluiden, die in de sampler zitten?

Ja, het zijn inderdaad echte geluiden. Maar als je nog een paar keer goed gaat luisteren en er een beetje over gaat nadenken, begin je te beseffen dat we allerlei dingen over 't hoofd hebben gezien. Dingen die zowel te maken hebben met de klank als met de muzikale voordracht. Over dat laatste gaan we het hebben in paragraaf 1.5. Nu kijken we alleen naar de klank.

Laten we eerst het geluid van een piano eens onder de loep nemen. Het idee van de sampler is dat je van elke toets een geluidsopname maakt. Dat klinkt heel voor de hand liggend, maar toch blijkt dat in de praktijk totaal onvoldoende te zijn! De reden daarvan is: naarmate je op een piano een toets harder aanslaat, klinkt de toon ook luider. Nu zou je zeggen dat het zeker mogelijk zou moeten zijn om ook de toetsen van zo'n samplerapparaat ***aanslaggevoelig*** te maken; dat betekent dat de toon die je hoort harder wordt naarmate je met meer kracht op de toets slaat. En dat is inderdaad goed uitvoerbaar: samplers zijn uitgerust met een aanslaggevoelig toetsenbord. Maar daarmee ben je er nog niet. Want naarmate je op een piano een toets harder aanslaat, wordt de toon niet alleen luider, maar ook helderder! En daarmee wordt het eenvoudigweg

een andere trilling, een ander geluid! En dat zou je weer opnieuw moeten opnemen!

"Klankwet"

We stuiten hier op een zeer belangrijke vuistregel die geldt voor heel veel trillingsverschijnselen, onder andere de klankopwekking in veel muziekinstrumenten. We noemen die regel de "klankwet" en formuleren die als volgt:

Tonen van muziekinstrumenten gaan vaak steeds helderder, briljanter klinken naarmate het volume ervan toeneemt.

Ook het omgekeerde is waar: hoe zachter een toon, des te doffer de klank ervan. Kort gezegd zou je de Klankwet dus ook kunnen omschrijven met: "hoe harder, hoe helderder" of met: "hoe zachter, hoe doffer."
Ook bij blaasinstrumenten kun je die koppeling van toonvolume en klankkleurverandering goed horen; bij koperblaasinstrumenten, zoals een trompet of een hoorn, begint de toon zelfs bij een zeker volume echt te "schetteren".
Je bent met dit soort verschijnselen zo vertrouwd dat het je niet eens opvalt. Pas als je met de computer gaat proberen om een trompet na te doen en daarbij de toon alleen maar harder maakt en niet scherper, hoor je direct dat er iets niet klopt.
In opdracht 5 illustreren we het effect.

5. Opdracht: klankwet demo
Beluister de volgende geluidsvoorbeelden:
a. ►H1_Toon_alleen_harder_zachter.mp3
Deze toon wordt alleen maar steeds harder en vervolgens alleen maar zachter en voldoet dus niet aan de Klankwet.
b. ►H1_Klankwet_demo.mp3
Hier hoor je een toon eerst harder en helderder worden en vervolgens zachter en doffer. Deze toon voldoet dus wel aan de klankwet. De toon is veel levendiger en doet meer denken aan een echt muziekinstrument (ook al is het dat duidelijk niet).
c. ►H1_Piano_hard_zacht.mp3
Wat je hoort, is twee keer dezelfde pianotoon, eerst zacht en dan hard aangeslagen. De zachte klinkt dof en donker; de harde toon klinkt scherper, agressiever, briljanter.

Conclusie pianotoets

Nu terug naar onze pianosampler. Je zou van elke pianotoets niet één, maar een heleboel, tientallen of zelfs honderden opnames moeten maken om de klank van alleen die ene toets vast te leggen! Bij elke toetskracht

hoort dan een opname. Alleen al voor het geluid van de piano zou je dan al vele duizenden opnames moeten maken!

Doorklinkpedaal

Maar stel dat je al die duizenden opnames inderdaad hebt gemaakt, dan nog zul je bedrogen uitkomen. Want een ander effect van de piano kun je niet vangen in die opnames. Als je namelijk op een piano het ***doorklinkpedaal*** intrapt en slechts één toets aanslaat, dan gaan ook de snaren van sommige andere toetsen door resonantie zachtjes, maar duidelijk hoorbaar, meeklinken. Sla je een heel akkoord aan, dan gaan er talloze snaren meeklinken. Daaraan dankt de piano z'n karakteristieke, ruimtelijke doorklinkeffect: een ware lawine van geluid komt op je af! Zonder dit effect is een piano geen piano meer. Ooit een stuk van Chopin horen spelen waarbij geen doorklinkpedaal werd gebruikt? En, wat vond je ervan?

Een heel bekend pianoproefje is het volgende. Doe de bovenklep van de piano open, zodat je de snaren kunt zien. Trap het doorklinkpedaal in. Hou je hoofd zo dicht mogelijk bij de snaren en schreeuw keihard: "hallo!!". Je hoort je stem galmen en echoën alsof je in een kerk staat. Kennelijk heeft het stemgeluid een groot aantal snaren in trilling gebracht.
In het volgende fragment wordt drie maal achter elkaar op het klankbord van de piano geklopt met een vilten pianohamertje (dat normaal tegen een snaar slaat): je hoort een dof, gesmoord geluid. Dan wordt het doorklinkpedaal ingetrapt en volgt er nog een vierde klop tegen het klankbord.
►H1_Piano_pedaal.mp3

Het is onmogelijk om dit doorklinkeffect te realiseren met de samplertechniek. Een samplerpiano bevat daarom nog geen fractie van de klankmogelijkheden van een echte, akoestische piano en zal dus nooit natuurlijk klinken!

Sampler of natuurkundig model?

Ook andere instrumenten zijn om vergelijkbare redenen niet volledig natuurgetrouw weer te geven met de samplertechniek. We hoeven die redenen hier niet allemaal te bespreken; in elk geval levert de samplertechniek hoogstens een oppervlakkige, ruwe benadering van de echte instrumenten en is deze techniek dus veel minder realistisch dan je op het eerste gezicht zou denken.

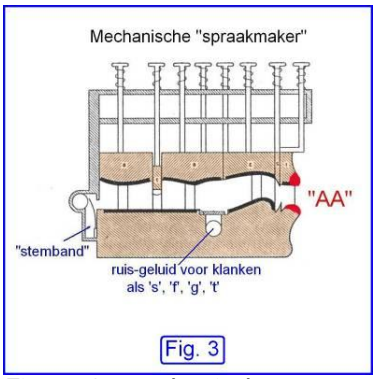
En hoe zit het nu met die natuurkundige modellen? Kun je daar de klankwet dan wel mee realiseren? En dat effect van het doorklinkpedaal? Het antwoord luidt in beide gevallen: absoluut! Als je goed luisterde, heb je ze in dat computerpiano voorbeeld van Beethoven ook alle twee al

kunnen opmerken. In die zin klonk het natuurkundige model - wat toch niets anders is dan de sonificatie van wiskundige objecten en wiskundige bewerkingen! - al een stuk realistischer en overtuigender dan de echte opnames in de sampler!!!!

Kortom: als we enige hoop mogen koesteren dat we ooit in staat zullen zijn om de fantastische en waanzinnig complexe klanken van muziekinstrumenten met de computer te evenaren - of misschien zelfs te overtreffen - dan zijn we aangewezen op natuurkundige modellen. Wel spelen sampleropnames een onmisbare, ondersteunende rol in die modellen; een microfoon is immers ook op te vatten als een meetinstrument en uit het geluid dat je ermee registreert, is allerlei belangrijke informatie af te leiden over het trillingsgedrag van een muziekinstrument. Die informatie blijkt zeer goed te kunnen worden ingepast in het natuurkundige model!

1.4 Spraak- en zangsynthese

Spraaksynthese (het opwekken van spraak met de computer) is een minstens even boeiende uitdaging voor sound design als muziek. We hadden het in paragraaf 1.2 al over de vocoder, die niet alleen muzikaal interessant is maar ook uitermate nuttige toepassingen heeft in de telecommunicatie.



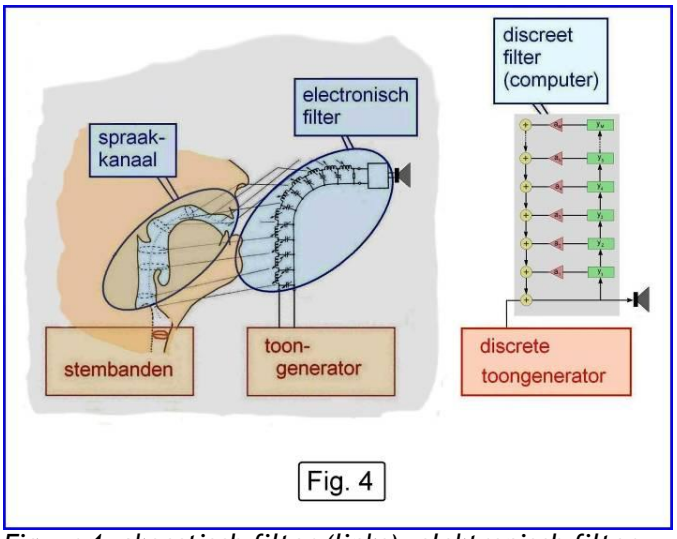
Figuur 3: mechanisch spraakkanaal. Bron: Wikipedia

Mechanisch praathoofd

De oudste vorm van spraaksynthese is de pratende pop. Tegen het einde van de 19de eeuw werden er gummimodellen van het spraakkanaal (mond- en keelholte) gemaakt. Je ziet er een in figuur 3. Linksonder bevindt zich een metalen strip die in trilling kan worden gebracht bij wijze van (enkelvoudige!) stemband. Het spraakkanaal is in een aantal segmenten verdeeld, waarvan je de diameters kunt veranderen door op de pistons boven te drukken (pistons zijn drukknoppen zoals bijvoorbeeld op een trompet). Daardoor ontstaat een bepaald resonantiepatroon dat ook in de echte mond- en keelholte optreedt bij een bepaalde mondstand. Dat resonantiepatroon zorgt ervoor dat je een bepaalde klinker hoort. Met een beetje oefening kon je dit hoofd goed verstaanbaar "cigarette" (met een Engelse r) laten zeggen!

Het spraakkanaal is een filter

Het spraakkanaal voert een bewerking uit op het geluid van de stembanden, waardoor het een andere klankkleur krijgt, bijvoorbeeld een "oe" of een "aa" of een "ie". In paragraaf 1.1 hebben we een bewerking die de klankkleur verandert een filter genoemd. Daarom is je mond- en keelholte een akoestisch filter.



Figuur 4: akoestisch filter (links), elektronisch filter (midden) en discreet filter (rechts).

Elektronisch filter

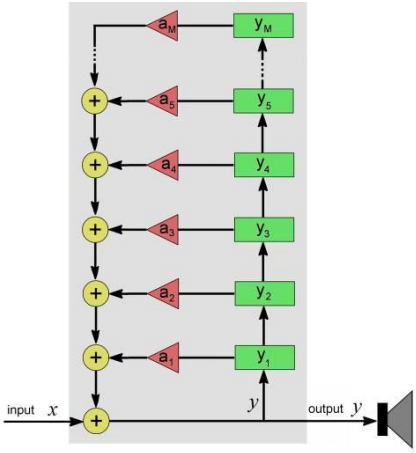
De werking van zo'n akoestisch filter kun je prima nabootsen met een elektronisch filter, dat bestaat uit elektronische componenten zoals weerstanden, spoelen en condensators. De werking van een akoestisch filter en een elektronisch filter zijn in principe gelijk. Beide zijn trillingscircuits waarin golven heen en weer kaatsen en zodoende een bepaald resonantiepatroon vormen dat je waarneemt als een klankkleur. In figuur 4 zie je ze naast elkaar afgebeeld. Om de gelijkenis te benadrukken, is de trillingskring van het elektronische filter net zo gekromd getekend als het spraakkanaal, zodat het lijkt alsof het elektronische filter ook een soort mond- en keelholte vormen. Op de plaats van de lippen is een luidsprekertje getekend.
Als je oog voor detail hebt, dan zie je dat er door de spoeltjes en condensatoren van het elektronische filter pijltjes lopen. Dat is om aan te geven dat het circuit instelbaar is. Net zoals je verschillende klankkleuren krijgt door je mond in verschillende standen te zetten, krijg je ook verschillende klankkleuren door de instellingen van het elektronische filter te veranderen.

Discreet filter

Rechts in figuur 4 zie je nog een derde filtervariant. Dit is een ***discreet*** filter. Ook daarmee kun je, net zoals met het elektronische filter, klankkleuren maken. Toch is de werking ervan totaal anders dan die van het akoestische en het elektronische filter! Door dit circuit bewegen namelijk geen luchtdeeltjes of elektronen heen en weer, maar getallen! Een discreet filter is niets anders dan een rekenschema. De berekeningen zou je met de hand of met een GRM kunnen uitvoeren en zodoende een "aa-" of een "oo-"klank kunnen maken! Het enige dat je daar waarschijnlijk van zal weerhouden is dat je voor elke seconde geluid maar liefst zo'n 2

miljoen optellingen en nog eens 2 miljoen vermenigvuldigingen moet uitvoeren...

Zulke omvangrijke rekenklussen doe je met de computer. Om alle berekeningen uit te voeren die voor het filteren van 1 seconde spraakgeluid nodig zijn, heeft een moderne PC minder dan 1 seconde rekentijd nodig. We zeggen dan dat de berekening kan verlopen in ***realtime***.



Figuur 5: discreet filter.

Ook een discreet filter is instelbaar, net zoals het spraakkanaal en het elektronische filter. In figuur 5 zie je het circuit wat groter afgebeeld. Een rood driehoekje is het symbool voor een vermenigvuldiging. In de driehoekjes staan vermenigvuldigingsfactoren: a_1, a_2, ... a_M. Door die getallen te veranderen ontstaan andere klankkleuren. We zullen discrete filters uitvoeriger bespreken in de hoofdstukken 4, 5 en 7.

Bij spraak staat je mond voortdurend in andere standen; je hoort dan ook voortdurend andere klanken. Als je een discreet filter van klank wilt doen veranderen, hoef je alleen maar andere waarden te nemen voor de factoren a_1, a_2, ... a_M. . Voor goed verstaanbare spraak moet je ze zo'n 50 keer per seconde "verversen". (Hoe je precies aan die waarden voor de vermenigvuldigingsfactoren: a_1, a_2, ... a_M. komt, is weer een ander verhaal...) Zie en hoor voorbeeld (a), hieronder.

Geluidsvoorbeelden computer-spraak en -zang

(a) "Maan, zaag, Fien..."

In paragraaf 1.2, voorbeeld (d), hadden we het over tijd-strek-, toonhoogte- en het Donald Duck-effect. We geven daar nu een voorbeeld van dat we combineren met dat idee van "50 spraakfilters per seconde". We hebben een opname gemaakt van die mooie zin: "Maan, zaag, Fien, vier, hut, juf, ring, Wim, koek, schoen..." (die een halve eeuw geleden nog door duizenden schoolkinderen van groep 3 dagelijks werd nagesproken terwijl de onderwijzer met een stok de bijbehorende plaatjes aan de muur aanwees.)
Die spraakopname duurde 6,35 seconde en werd opgedeeld in 1400 (elkaar overlappende) fragmenten. Met WaveWizard (zie paragraaf 1.6) hebben we 1400 sets vermenigvuldigingsfactoren: a_1, a_2, ... a_M aangemaakt (er bestaat een computermethode waarmee je uit een digitale opname van spraak razendsnel de factoren: a_1, a_2, ... a_M kunt berekenen). Daarmee kunnen we de klankopeenvolging "Maan, zaag, Fien..." representeren in 1400 verschillend klinkende filters.

Precies zoals je de vele verschillende klanken die het spraakkanaal produceert hoorbaar kunt maken door tikjes tegen je wang te geven terwijl je je mond in steeds weer andere standen zet (probeer even!), kun je ook "tikjes" tegen een discreet filter geven en daarmee de klank ervan hoorbaar maken. Dat is wat je in dit voorbeeld hoort. Voor de grap (en voor de duidelijkheid) hebben we de tijd tussen elke twee tikjes steeds korter gemaakt (dit gebeurt ***eenparig versneld***: denk aan het geluid van een wasknijper tegen een spaak terwijl je met je fiets een berg afrijdt). Daarom hoor je eerst afzonderlijke tikjes met elk een iets andere klankkleur; geleidelijk gaan die over in een "ratel" die je op een zeker moment gaat waarnemen als een toon (toonhoogte waarneming begint bij ongeveer 20 Hz). Daarbij vloeien de klanken van de afzonderlijke filters geleidelijk samen tot verstaanbare spraak. Om het geluid een beetje op te poetsen hebben we bovendien een galmeffect toegevoegd.
Luister naar ►H1_Maan_zaag.mp3. Je hoort duidelijk de tijd-strek- en toonhoogteverandering terwijl er geen Donald Duck-effect optreedt.

(b) "Desi, Desi..."
Dat je met de computer de klanken die mond- en keelholte produceren prima kunt nadoen, wil nog niet zeggen dat computerzang een eitje is. Hoe maak je vibrato en andere toonhoogtevariaties? Hoe maak je abrupte overgangen in toonhoogte, zoals bij jodelen? Dat niet alleen echte zangers maar ook sound designers veel moeten oefenen, illustreert Perry Cook in deze grappige "étude".
Luister naar ►H1_zangoefening_jodelen_Desi.mp3.

(c) Limericks
Hoe maak je computerzangstemmen waarin je duidelijk een verschil kunt horen tussen hoge en lage stemmen (sopraan, alt, tenor, bas)? Kun je een basstem geleidelijk hoger laten zingen op een zodanige manier dat ook het karakter van de stem geleidelijk verandert van een mannen- in een vrouwenstem? Kun je verschillende computerstemmen ook samen een stukje laten zingen, zo dat je toch het verschil in stemklank hoort? Dat onderzoekt een sound designer in ►H1_Limericks_zangsynthese.mp3.

1.5 "Is er nog leven na MIDI?"

Mp3 en wav

Het zal je niet zijn ontgaan dat je bij de geluidsbestanden tot nu toe twee verschillende extensies bent tegen gekomen: ***mp3*** en ***mid***.
De eerste daarvan, mp3, is een geluidsbestand dat geluidstrillingen representeert, en wel op een heel efficiënte, ***gecomprimeerde*** ("ingepakte") manier. Je zult in deze module ook nog te maken krijgen met een ander soort geluidsbestand, dat je waarschijnlijk ook wel kent, namelijk ***wav***. Dit laatste type is ongecomprimeerd, dat wil zeggen dat de computer niet eerst, zoals bij mp3, berekeningen moet uitvoeren voordat

de eigenlijke trillingen door de geluidskaart kunnen worden omgezet in hoorbaar geluid. Het wav-bestandstype kost dus maar heel weinig rekentijd, maar heeft als nadeel dat het minstens tien keer zo groot is als het mp3-bestandstype. Je vindt op internet emmers vol gratis software waarmee je een wav kunt omzetten (***converteren***) naar mp3 en omgekeerd. Een dergelijk programmaatje tref je ook aan op ►vaklokaal NLT ; we zullen het vanaf het hoofdstuk 2 een paar keer nodig hebben.

MIDI

Geluidsbestanden zoals mp3 en wav zijn universeel; je kunt er elk geluid mee weergeven: spraak, muziek, verkeerslawaai, ruisende bomen, zwemmende zaadcellen. Daarnaast hebben we ook nog het mid-bestandstype; daarmee kun je uitsluitend muziek representeren!!

De bestandsextensie mid staat voor ***MIDI*** en MIDI is ook weer een afkorting: ***Musical Instruments Digital Interface***: digitale koppeling tussen muziekinstrumenten - *elektronische* muziekinstrumenten, wel te verstaan. Om beter te kunnen uitleggen wat MIDI precies inhoudt, moeten we eerst iets vertellen over het ontstaan ervan.

MIDI dateert uit begin jaren 80, toen de digitale techniek langzaam zijn intrede deed. Op dat moment werden alle typen synthesizers nog gebouwd met analoge elektronische componenten (zoals weerstanden, transistors en spoeltjes); ze waren heel verschillend van opzet en van klankmogelijkheden. Daarom stonden er bij popconcerten een heleboel verschillende types synthesizers op en naast elkaar gestapeld en zag je de toetsenman wild heen en weer springen tussen diverse toetsenborden. Je kon in sommige gevallen synthesizers ook met elkaar doorverbinden en ze vanaf één toetsenbord bespelen en op die manier waanzinnig ingewikkelde klanken produceren. Maar dat was niet zonder risico, want vaak ontstond er kortsluiting en regelmatig bezorgde het aanraken van een knopje de toetsenist een flinke opdonder.
De MIDI-standaard werd (in 1983) door fabrikanten van synthesizers opgesteld om aan dat soort hachelijke situaties een einde te maken. Het meest sensationele aan MIDI was, in die dagen, dat het volledig digitaal was: er worden via kabels alleen maar getalcodes (vergelijkbaar met "streepjescodes") tussen apparaten heen en weer gestuurd. Alle bedieningselementen van de synthesizer (toetsen, pedalen, volumeregelaars, klankselectieknoppen) hebben een eigen getal. Alle verschillende merken synthesizers "begrijpen" die getalcodes en kunnen ze ook direct uitvoeren. Bovendien maakten de fabrikanten heldere afspraken over de elektronische techniek die voor de MIDI-communicatie zou worden gebruikt; de kans op kortsluiting werd tot nul gereduceerd.

MIDI was onmiddellijk een groot succes. Toch besefte op dat moment niemand nog dat MIDI een ware revolutie teweeg zou gaan brengen in de muziek en in de bouw van muziekinstrumenten. Maar het duurde niet lang voordat fabrikanten synthesizers op de markt begonnen te brengen waar helemaal geen toetsen meer op zaten, alleen maar een paar MIDI-ingangen en -uitgangen. Dat was niet alleen veel goedkoper, maar ook de aanblik van het concertpodium werd er strakker door. Je zag voortaan de toetsenman op een vaste plek achter één toetsenbord staan, het zogeheten ***masterkeyboard***, dat via de MIDI-kabel een aantal toetsloze synthesizermodules aanstuurde, die keurig opgestapeld achter het podium stonden. (N.B. een masterkeyboard moet je niet verwarren met het keyboard uit paragraaf 1.3!!).
Dat arsenaal aan synthesizers leverde bovendien een nog veel indrukwekkender klank op dan ooit tevoren mogelijk was geweest.

MIDI en computer

Maar dat alles was nog niets vergeleken met de enorme doorbraak die in de muziek ontstond toen mensen op het (eigenlijk erg voor de hand liggende) idee kwamen om een MIDI-synthesizer aan te sluiten op een computer. Rond 1985 begon dat grootschalig te gebeuren. Om de gevolgen daarvan goed te snappen, moeten we kijken naar wat MIDI nu eigenlijk wel en niet doet. Vaak lees je in muziekbladen dat MIDI "muzieknoten verstuurt" van de ene naar de andere synthesizer. Dat is niet waar. MIDI kan geen noten verzenden en dat heeft ook helemaal geen zin.
MIDI is een realtime systeem. Het centrale begrip in MIDI is *event* (gebeurtenis). MIDI geeft informatie over gebeurtenissen die plaats vinden op het instrument, dat wil zeggen over alles wat een muzikant doet op een instrument: toetsen aanslaan, toetsen loslaten, pedaal intrappen, pedaal loslaten, etc. Voor al die gebeurtenissen bestaat een code. Op het moment dat een gebeurtenis plaats vindt - niet eerder en niet later - verstuurt MIDI de corresponderende getalcode. Bijvoorbeeld als een toets wordt aangeslagen, geeft MIDI een getalcode af met de boodschap: "er wordt een toets aangeslagen, en wel toets nummer zoveel, en met kracht zoveel.”
Als even later de toets weer wordt losgelaten, klikt MIDI onmiddellijk: "er wordt een toets losgelaten, en wel toets nummer zoveel".
Dus MIDI verzendt geen noten, want, zoals we in hoofdstuk 6 (voortaan korten we af: H6) nader gaan bekijken, een noot bevat allerlei informatie over een toon, onder andere over de nootlengte. En nootlengte is geen gebeurtenis, maar het tijdsverschil tussen twee gebeurtenissen, namelijk het aanslaan en loslaten van een toets.

Wanneer je nu de MIDI-kabel van een synthesizer aansluit op een computer, kun je niet alleen de ontvangen MIDI-boodschappen noteren, maar ook de tijdstippen waarop die boodschappen binnenkomen. Zo maakt de computer een lijst getallen waarin het muziekstuk zoals het door de

muzikant wordt gespeeld, dus inclusief de voordracht, op een heel exacte manier wordt geregistreerd. Hierbij gaat het dus niet om een geluidsopname, maar om een "toetsen"-opname! Ze noemen dat wel eens ***MIDI-recording*** en een lijst met MIDI-commando's en bijbehorende tijdstippen een ***MIDI-bestand***. Het heeft de extensie mid. Een voorbeeld van een MIDI-bestand met z'n getalcodes zie je in figuur 42.

Computer bespeelt synthesizer

De reden waarom MIDI zo'n enorme revolutie in de muziek teweeg heeft gebracht is dat je dit proces ook kunt omkeren!! De lijst getallen waarmee de computer een stuk muziek heeft vastgelegd dat op een synthesizer werd gespeeld, kan ook door de computer zelf worden gebruikt om precies op de juiste tijdstippen toetsinformatie via de MIDI-kabel naar een synthesizer te sturen. Daarmee produceert de synthesizer weer exact dezelfde klanken. Kortom: de computer kan een synthesizer bespelen, net zoals een muzikant dat doet, maar kan bovendien hetzelfde stuk ook sneller of langzamer, hoger of lager laten klinken.

Componerende computer

Een andere consequentie is dat je met de computer uitstekend kunt componeren: in plaats van noten neer te schrijven op de notenbalk, voer je ze in de computer in en brengt ze direct via MIDI en synthesizer tot klinken. Zo kun je het klinkende resultaat al in het vroegste ontwikkelstadium beluisteren. Dat brengt je vaak op nieuwe ideeën voor het stuk dat je aan het maken bent.
De koppeling van de computer aan synthesizers heeft ook tot een heel nieuwe ontwikkeling geleid: niet alleen componisten kunnen nieuwe muzikale klanken creëren, ook de computer kan dat! Daarvoor is nodig dat je compositieregels opstelt en uitwerkt in de vorm software. De computer kiest door te “dobbelen” een volstrekt willekeurige toon en kijkt dan of volgens de opgestelde regels die toon op dit punt, in deze context, "geoorloofd" is. Zo ja, dan wordt de toon geselecteerd in de compositie, zo nee, dan werpt de computer simpelweg opnieuw met de dobbelsteen om een nieuwe toon te selecteren, enz. Het resultaat is een wonderlijk mengsel van onnavolgbare, maar toch op een of andere manier "logische" melodieën en samenklanken. In hoofdstuk 6 ga je de computer willekeurige toonhoogten bij elkaar laten dobbelen en nog andere effecten laten maken. Daarmee illustreer je hoe onmisbaar dobbelsteengetallen (***random getallen***) zijn om de klank en de voordracht juist minder "computerachtig" of "mechanisch" te laten klinken.

Tel je dat allemaal bij elkaar op, dan mag je gerust stellen dat de komst van MIDI een belangrijke wending in de geschiedenis van de muziek markeert. Al tegen het eind van de jaren 80 verzuchtte een columnist van een bekend muziekblad: "Is er nog leven na MIDI?" Met andere woorden:

wat valt er aan muziek nog meer te sleutelen dan wat er al niet door MIDI kan worden gerealiseerd?
We hebben het antwoord daarop eigenlijk al gegeven. De opkomst van natuurkundige modellen en alle fabelachtige klankbewerkingen die je tot nu toe in de voorbeelden al bent tegengekomen, vormen samen dat leven na MIDI dat sinds begin jaren 90 in ontwikkeling is. Toch heeft die columnist wel gelijk dat MIDI in de jaren 80 werkelijk insloeg als een bom en de muziek radicaal heeft veranderd. Dat proces is eigenlijk nog steeds gaande.

1.6 Sound design en NLT

Lees deze paragraaf extra aandachtig door: hier vind je het "spoorboekje" van de module. In deze paragraaf staat wat je in de volgende hoofdstukken leert en waarom je dat leert. Ook in het geval dat je je in latere hoofdstukken mocht afvragen: "waarom moet ik dit leren of kunnen?", is het goed deze paragraaf nog eens door te lezen!

Je hebt inmiddels een globaal idee van wat sound design inhoudt. Nu wordt het tijd om duidelijk te maken wat we in de volgende hoofdstukken wel en niet gaan doen. Want realiseer je goed: over sound design kun je heel uiteenlopende soorten lesmateriaal schrijven! Een sound design cursus die bedoeld is voor het vak muziek zal heel andere doelen nastreven, zal er heel anders uitzien en zal van heel andere software gebruik maken dan een NLT-module over sound design!
Je hebt deze tekst waarschijnlijk al eens doorgebladerd en een boel grafieken, schema's, diagrammen en formules zien staan. Daaraan kun je dat NLT- karakter al afmeten. De sound design-lessen worden dan ook niet gegeven door de muziekdocent, maar door de docent wis- of natuurkunde!

Aan het slot van paragraaf 1.1 las je: "We zullen in de rest van deze module zien dat alle bewerkingen die je bij sound design verricht niets anders zijn dan sonificaties van wiskundige objecten en wiskundige bewerkingen." Dat is dan ook precies wat we in de volgende hoofdstukken gaan doen: je gaat geluidsbewerkingen uitvoeren door de computer rechtstreeks te instrueren met wiskundige formules. In de laatste twee hoofdstukken laat je de computer ook hele muziekstukken doorrekenen, ook weer op basis van formules en programmeertaalinstructies (en MIDI natuurlijk).

Nogmaals de klankwet

Toen je de geluidsvoorbeelden van al de vorige paragrafen beluisterde, heb je al kunnen ontdekken dat zelfs de meest alledaagse geluiden veel complexer, veel "listiger" in elkaar blijken te zitten dan je altijd had gedacht. Vraag bijvoorbeeld eens aan mensen wat je nu eigenlijk precies

hoort als je een snaar van een gitaar tokkelt. De meesten (ook veel gitaristen!) zullen antwoorden: "een toon die steeds zachter wordt". En dat klopt natuurlijk ook. Maar toch, als je met de computer een toon maakt die inderdaad alleen maar zachter wordt, dan krijg je een geluid te horen waarvan niemand zal zeggen: "ik hoor het geluid van een getokkelde snaar." Een van de belangrijkste redenen daarvan is dat een toon die alleen maar zachter wordt, niet voldoet aan de klankwet (zie paragraaf 1.3) en getokkelde snaren doen dat nu eenmaal wèl. We hebben van de klankwet al enkele voorbeelden beluisterd. Daar voegen we in de volgende opdracht nog het voorbeeld van de getokkelde snaar aan toe.

6. Opdracht: zachter en doffer
Luister naar ►H1_Gedempt_en_GedemptDoffer.mp3.
Eerst hoor je een toon die alleen maar zachter wordt.
Na 5 seconden hoor je een toon die zachter en doffer wordt op de manier van een getokkelde snaar.

Misschien vind je de eerste toon heel "gewoon" klinken, maar zodra je de tweede toon hebt gehoord - die verbluffend veel lijkt op die van een echte, getokkelde snaar - besef je pas hoe onnatuurlijk en "elektronisch" de eerste toon klinkt. Toch is ook die tweede toon geen gesamplede opname of zoiets, maar voor 100% gemaakt met de computer!

Rode draad van de module: "op zoek naar realisaties van de Klankwet"

Dit voorbeeld van de getokkelde snaar, die niet alleen steeds zachter maar ook steeds doffer wordt, vormt de rode draad van deze module. Het is het simpelste voorbeeld van de realisatie van de klankwet. En de klankwet is geldig voor alle vormen van akoestische en mechanische trillingen en daarmee ook voor de klankopwekking in muziekinstrumenten.
Kort samengevat kun je zeggen dat we in deze module op zoek gaan naar het antwoord op de volgende vragen:

1. Hoe kunnen we de klankwet zodanig wiskundig formuleren dat we (met de computer) het geluid van een getokkelde snaar kunnen realiseren?
2. Hoe kunnen we die snaarsimulatie gebruiken om er een heel snaarmuziekstuk mee te maken?
3. Hoe kunnen we de klankwet ook nog op een meer universele manier formuleren, zodat we, naast die getokkelde snaar, een veel grotere klasse van geluidsverschijnselen die voldoen aan de klankwet kunnen realiseren?

De antwoorden op deze vragen vind je pas in de laatste twee hoofdstukken van de module! Misschien vind je die "suspense" wel erg lang duren, maar

je zult snel ontdekken dat de weg naar dat einddoel ons voert langs fundamentele geluidsbewerkingen zoals:

- mixen — het gelijktijdig laten samenklinken van verschillende geluiden
- versterken — geluid harder maken (en het omgekeerde: verzwakken)
- in- en outfaden — geluid geleidelijk steeds harder of zachter maken
- echo's toevoegen — ook eindeloos repeterende echo's, zoals in trappenhuizen en viaducten
- filteren — de allerbelangrijkste bewerking in alle vormen van sound design.

Voordat we uiteindelijk aankomen bij die snaarsimulatie en een universeel bruikbaar "klankwetfilter", weet je dus al hoe je al die elementaire geluidsbewerkingen - die onmisbaar zijn in elke vorm van sound design - wiskundig kunt formuleren.
Sterker nog: je zult in hoofdstuk 7 ontdekken dat je eigenlijk niets nieuws hoeft te leren om dat realistische tokkelgeluid te maken. Je hoeft alleen maar die meer elementaire bewerkingen uit de eerdere hoofdstukken op een heel uitgekookte manier met elkaar te combineren!

Audiosoftware WaveWizard

Het programma dat we gaan gebruiken is vooral ontworpen voor het onderwijs en heet WaveWizard. Het is geschreven door een van de ontwikkelaars van deze module. Geen wonder dus dat WaveWizard de laatste jaren zeer sterk werd beïnvloed door de eisen die NLT stelt aan een module over sound design! Het belangrijkste kenmerk is dat WaveWizard de wiskunde niet wegstopt achter gelikte menufuncties, maar jou juist in staat stelt om geluid heel direct "aan te sturen" door middel van wiskundige formules, en wel op zo'n manier dat je op een gebruiksvriendelijke manier snel een geluid of zelfs een heel muziekstuk kunt maken.
Hierboven heb je al een aantal korte geluidsvoorbeelden gehoord die met WaveWizard zijn gemaakt en er volgt zo dadelijk nog een langer.

Eindtermen (leerdoelen) van de afzonderlijke hoofdstukken, toetsing en beoordeling

Hieronder vind je een gedetailleerde beschrijving van wat je na elk hoofdstuk geacht wordt te kennen en te kunnen. Ook de vier toetsen en hun gewicht [in %] zijn aangegeven.

Hoofdstuk 1 Intro sound design

Eindtermen

1. Je kunt in grote lijnen aangeven wat er onder sound design wordt verstaan, en een aantal typische technieken kort beschrijven.

2. Je kunt de inhoud van de "klankwet" aangeven en er voorbeelden van geven..
3. Je kunt de verschillen uitleggen tussen veel gebruikte typen audio- en muziekbestanden zoals WAV, MP3 en MIDI.
4. Je kunt aangeven dat een eigenschap van een geluid op te merken is door dat geluid te vergelijken met een soortgelijk geluid waarin die eigenschap ontbreekt.
5. Je kunt het specifieke NLT- karakter van deze sound design cursus onder woorden brengen.

Toets H1: schriftelijk, zonder computer, duur 20 minuten. [15%]

Hoofdstuk 2 Horen en zien

Vanaf dit hoofdstuk gaan we intensief gebruik maken van de computer. In hoeverre kun je door plaatjes en animaties een betere voorstelling krijgen van geluid? Met andere woorden: hoe kan je oog je oor te hulp komen om trillingsverschijnselen beter te snappen? Naast het bekende $u(t)$-diagram leer je dat je geluid ook nog op een heel andere manier kunt afbeelden, namelijk door het ***spectrum***, dat de boventonen van een toon veel duidelijker laat zien dan het $u(t)$-diagram. Dat is enorm belangrijk, want vooral klankkleur is iets dat veel beter tot uitdrukking komt in het spectrum. Speciale aandacht zullen we schenken aan tonen met harmonische boventonen.
We maken in dit hoofdstuk ook kennis met het ***sonogram***. Een sonogram is een afbeelding die heel geschikt is om veranderingen in toonhoogte en klankkleur weer te geven.

Eindtermen

1. Je bent in staat om geluidseigenschappen zoals toonhoogte, volume en klankkleur niet alleen auditief (dus met je oren), maar ook visueel te herkennen, en wel in twee verschillende grafische representaties van geluid, namelijk de golfvorm ($u(t)$-diagram) en het spectrum (frequentiediagram).
2. Je kunt begrippen als grondtoon en boventonen uitleggen en aangeven onder welke voorwaarden boventonen harmonisch genoemd worden.
3. Je bent in staat om een zogeheten sonogram te interpreteren, dat wil zeggen in een plaatje aan te wijzen hoe een geluid in de loop van de tijd bijvoorbeeld helderder of doffer wordt.

Toets H2: schriftelijk, zonder computer, duur 20 minuten. [15%]

Hoofdstuk 3 Sampling en de wiskundige representatie van digitaal geluid

Om geluid met een computer te kunnen bewerken moet je het eerst omzetten in een reeks getallen (sampling). Pas als je weet hoe dat in elkaar zit, kun je geluid gaan bewerken en tonen opwekken.

Eindtermen

1. Je kunt uitleggen aan welke voorwaarden moet worden voldaan om een geluid "digitaal" te maken, dat wil zeggen te representeren door een reeks getallen.
2. Je bent vertrouwd met de begrippen Sampling frequentie F_s en Sampling tijd T_s, de Nyquist-frequentie.
3. Je kunt de betekenis van DA- en AD-conversie weergeven.

Hoofdstuk 4 Versterken, mixen en echo's toevoegen

Elke geluidsbewerking blijkt niets anders te zijn dan een getalbewerking die je beschrijft in de vorm van een wiskundige formule.

Eindtermen

1. Je kunt een (digitaal) geluid niet alleen harder of zachter maken, maar ook geleidelijk van volume veranderen ("in"- en "outfaden"). Je doet dat door middel van discrete, lineaire functies waarvan je zelf de formule opstelt.
2. Je kunt het volumeverloop van tonen regelen met een zogeheten contourgenerator die in synthesizers wordt gebruikt.
3. Je kunt twee of meerdere geluiden met elkaar mixen.
4. Je kunt uitleggen wat verstaan wordt onder fouriersynthese ("mix van sinustonen") en kunt enkele zeer veel voorkomende synthesizer-golfvormen, zoals de zaagtand en de blokgolf een fouriersynthese maken.
5. Je bent in staat om aan een geluid twee soorten echo-effecten toe te voegen, namelijk een eenmalige echo en een eindeloos repeterende echo. Beide effecten kun je:
   - in wiskundige vorm beschrijven (niet-recursieve en recursieve differentievergelijkingen)
   - grafisch weergeven door middel van systeemdiagrammen (feedforward- en feedback).

Hoofdstuk 5 Filters

Een filter is een bewerking die de klank(kleur) van een geluid wijzigt.

Eindtermen

1. Je kunt uitleggen waarom een filtereffect altijd tevens een echo-effect is en dus op dezelfde manier kan worden weergegeven in systeemdiagrammen (feedforward- en feedback) als elk echo-effect (zie hoofdstuk 4).
2. Je kunt de van volgende typen filters aangeven wat hun invloed is op het geluid:
   - laagdoorlatend (maakt geluid doffer, donkerder)
   - hoogdoorlatend (maakt geluid scherper, briljanter)

- banddoorlatend (maakt geluid "resonerender" of "toniger", denk aan het klankverschil tussen ruisende bomen en "huilende" wind).

3. Je kunt (met de computer) het klankveranderende effect van een filter zichtbaar maken in een frequentiegrafiek (zogeheten amplitudekarakteristiek). En aan de hand daarvan kun je vertellen om wat voor type filter (zie punt 2) het gaat.

Toets H3 t/m H5: schriftelijk, met WaveWizard, duur 50 minuten. [35%]

Hoofdstuk 6 Sequencing en MIDI

Hoe kun je een hele reeks tonen na elkaar en/of tegelijkertijd laten klinken (= muziekstuk)? Dat doe je door slechts één instructie die je in elke programmeertaal tegenkomt en die je misschien wel eens hebt gebruikt op je grafische rekenmachine, namelijk de **`FOR...TO...NEXT`** instructie, waardoor je een willekeurig aantal commando's een willekeurig aantal keer kunt laten uitvoeren (zie figuur 42). Je maakt van die instructie gebruik om een MIDI-bestand (waarin een voordracht van een bepaalde compositie is vastgelegd) tot klinken te brengen.

Eindtermen

1. Je bent in staat om met de computer een reeks tonen met willekeurige frequenties, volumes, lengten en begintijdstippen op te wekken door middel van de lus-instructie **`FOR...TO...NEXT`**.
2. Je kunt aangeven wat bedoeld wordt met de evenredig zwevende stemming (meest gebruikte stemming in de muziek) en kunt de wiskundige formulering daarvan afleiden.
3. Je bent vertrouwd met al die elementen van het MIDI-protocol waarmee een muzikale noot wordt beschreven, en de wijze waarop de MIDI-data van een willekeurig muziekstuk in software worden gebruikt om toongeneratoren mee aan te sturen. Je kunt dat ook zelf toepassen in een muzikaal voorbeeld.

Hoofdstuk 7 Tonen waarvan de klankkleur verandert

Dit hoofdstuk beschrijft twee methoden waarmee de belangrijkste eis wordt gerealiseerd die we aan muzikale tonen stellen, namelijk dat ze voldoen aan de klankwet.
De snaarplukvergelijking is een zeer bekend algoritme (rekenmethode) waarmee het geluid van een getokkelde snaar zeer realistisch kan worden nagebootst.
Een tijdveranderlijk filter (= een filter waarvan de klankeigenschappen geleidelijk veranderen; voorbeeld: zing een toon terwijl je je mondstand geleidelijk verandert van "IE" naar "OE").

Eindtermen

1. Je kunt de ***snaarplukvergelijking*** (een formule waarmee de klank van een getokkelde snaar realistisch wordt nagebootst)
   a. verklaren aan de hand van systeemdiagrammen
   b. wiskundig afleiden
   c. toepassen als toongenerator die door een MIDI-bestand wordt aangestuurd.
2. Tijdveranderlijk filter. Je kunt van een banddoorlaatfilter zowel de centrumfrequentie als de bandbreedte instellen als lineaire functie van de tijd.
3. Je kunt uitleggen wat een amplitude-afhankelijk laagdoorlaatfilter is en waarom je daarmee de klankwet kunt implementeren.

Muzikale toepassing

In de laatste paragraaf van hoofdstuk 7 komen alle technieken die je hebt geleerd samen: toonopwekkers aansturen met een MIDI-file, waarbij je gebruik maakt van het snaarpluk-effect om het geluid van eerst een klavecimbel (opdracht 66) en vervolgens dat van een piano te imiteren. De resultaten daarvan hangen natuurlijk van jou af, maar kunnen klinken zoals in de volgende MP3-bestanden:

►H7_Chopin_Etude_3_Clavecimbel.mp3
►H7_Rachmaninov_Rhaps_Var18.mp3

Toets H6 en H7: schriftelijk, met WaveWizard, duur 50 minuten. [35%]

Slotopdracht

In de laatste opdracht ga je zelf een geluid maken (stukje muziek of een geluidseffect) waarin je gebruik maakt van de technieken die je in deze module hebt leren kennen en toepassen. Je docent bepaalt in hoeverre de slotopdracht meeweegt in je eindbeoordeling.

Betekenis van het sterretje *

In deze module zie je bij een aantal paragrafen en opdrachten een sterretje *. Ze hebben de volgende betekenissen:

- Paragrafen met * kun je zonder verlies van continuïteit overslaan; ze zijn bedoeld als verdieping of verbreding en worden niet getoetst.
- Opdrachten met * zijn pittiger dan het toetsniveau en liggen wat meer in de interessesfeer van het NT- profiel. We dagen je uit om er je tanden eens in te zetten, maar je mag ze ook overslaan.

Website

We hopen dat je door deze module de smaak te pakken zult krijgen. Je komt dan voor de vraag te staan: hoe nu verder? Lees daarover meer op de speciale website voor sound design projecten ten behoeve van het voortgezet onderwijs: ►URL1.
Je vindt daar:

- enkele hoofdstukken die wellicht in een later stadium verder zullen worden uitgewerkt tot een tweede module of die geschikt zijn voor het profielwerkstuk of voor praktische opdrachten
- literatuurlijst over computermuziek en signaalverwerking
- een aantal uitgewerkte opgaven die ook goed van pas komen bij de voorbereiding op de toetsen voor deze module.

Vakopleidingen

Sonologie, Koninklijk Conservatorium in Den Haag, ►URL4
Sound design aan de Hogeschool voor de Kunsten Utrecht, ►URL5

Woordenlijst

aanslaggevoelig §3
bewerking §1
componist §1, §3
comprimeren §5
converteren §5
cross fading §2
differentiëren §1
discreet filter §4
Donald Duck-effect §2
doorklinkpedaal §3
eenparig versneld §4
fader §2
filter §1, §4
formantverschuiving §2
geluidsfictie §1
keyboard §3
klankkleur §1, §4
"klankwet" §3
masterkeyboard §5
meerstemmig §6
mid §5
MIDI §5
MIDI-bestand §5
MIDI-recording §5
mix §2
morphing §2
mp3 §5
multi-timbraal §6
natuurkundig model §3
opwekken §1
organised sound §2
percussie-instrument §3
periodiek §1
random getal §5
realisme §1
realtime §4
sampler §3
sinustoon §1
slagklank §3
snaarplukvergelijking §6
sonificatie §1
sonogram §6
sound design §1
spectrum §6
spraaksynthese §4
synthesizer §3
synthetiseren §2
toongenerator §1
visualisatie §1
vocoding §2
waterlabyrint-effect §1
wav §5
WaveWizard §3, §6

2 Horen en zien

Naast geluidsvoorbeelden vind je vanaf dit hoofdstuk ook presets. Dit zijn teksten waarin menucommando's beschreven zijn voor het programma WaveWizard.
Al deze bestanden, geluiden en presets, worden in de tekst aangegeven door dit symbool ► en zijn wederom te vinden op vaklokaal NLT.

2.1 Inleiding

In dit hoofdstuk wordt het allemaal wat praktischer. Je gaat zelf aan de slag, vooral met de computer.
We beginnen met een rondleiding langs de belangrijkste functies van WaveWizard's hoofdpagina. Zo kunnen we aan de hand van een paar geluidsvoorbeelden enkele basisbegrippen over geluid herhalen waarmee je kennis maakte bij natuurkunde in de onderbouw. Je ontdekte toen hoe je ogen je te hulp komen om beter te snappen wat je oren nu eigenlijk horen! Je maakte kennis met het zogeheten ***uitwijking-tijd diagram (u(t)-diagram)***, een afbeelding van geluidstrillingen waarin je informatie over onder andere het volume en de toonhoogte kunt aflezen. In paragraaf 2.3 zetten we dat allemaal nog eens op een rij.

Boventonen, spectrum en sonogram
Naast het *u*(*t*)-diagram bestaat er nog een ander soort afbeelding van geluid waarop die eigenschappen vaak nog veel duidelijker worden weergegeven: het ***spectrum***. Zoals we in paragraaf 1.1 al opmerkten: je kunt elk geluid maken door een aantal sinustonen met verschillende toonhoogten en verschillende volumes gelijktijdig te laten klinken. Omgekeerd blijkt dan ook elk geluid, zoals spraak en muziek, te zijn opgebouwd uit sinustonen, die meestal ***boventonen*** worden genoemd. De sterkteverhoudingen van de boventonen zijn bepalend voor de klankkleur ("scherp", "dof", "nasaal"). In het spectrum kun je zien hoe sterk alle boventonen van een geluid zijn; dat informeert je veel beter over de ***klankkleur*** dan het *u*(*t*)-diagram.
Naast het spectrum maak je ook kennis met het ***sonogram***, een afbeelding waarop je veranderingen van toonhoogte, volume en klankkleur goed kunt zien.

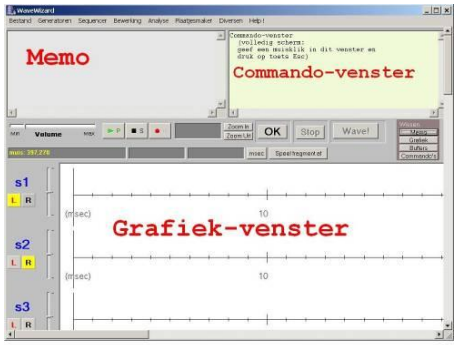


Fig. 6

Figuur 6: WaveWizard, hoofdscherm.

2.2 De hoofdpagina van WaveWizard

Start het programma. Je krijgt nu het hoofdscherm in beeld.

Memo, Commando-venster en Grafiek-venster

Het grijze vak linksboven heet **Memo**. Daarin praat WaveWizard met jou over het verloop van allerlei processen. Rechts daarvan, met een lichtgele achtergrond, zie je het **Commando-venster**. Dat is de plaats waar jij praat met WaveWizard. Meer daarover in hoofdstuk 3. Het grote witte vak onder Memo en Commando-venster heet **Grafiek-venster**. Daar tekent WaveWizard grafieken of animaties. Op dit moment zie je daar drie horizontale lijnen: de tijd-assen van drie uitwijking-tijd-diagrammen.

Sporen S1, S2 en S3

De drie tijd-assen horen bij de drie geluidssporen van WaveWizard, spoor S1, spoor S2 en spoor S3. Links in beeld zie je ze afgebeeld met daaronder steeds twee drukknopjes **L** en **R**. Daarmee kun je het geluid van een spoor naar een luidspreker sturen. Op dit moment wordt spoor S1 naar de linker box (of linker oor van je koptelefoon) gestuurd (knopje **L** is geel), maar niet naar de rechter (knopje **R** is grijs). Bij spoor S2 is het net omgekeerd.

7. Opdracht: geluid afspelen en bekijken
Ga met de muis naar menu **Bestand** en selecteer **Openen .WAV.**
Ga naar submap **WAV** in map Sound Design en open ►Raaf.wav.
Je ziet nu niet één, maar twee golfvormen verschijnen, op sporen S1 en S2. Dat komt omdat dit geluid ***stereo*** is; de twee luidsprekers geven elk een iets andere geluidstrilling weer die samen een ruimtelijke indruk geven. (Onder andere doordat de golven van een geluidsbron je linker oor op een iets eerder of later tijdstip bereiken, ervaar je de ruimtelijkheid van het geluid; daarom heb je ook minstens twee boxen nodig om die weer te geven).

Afspelen, opnemen en stoppen

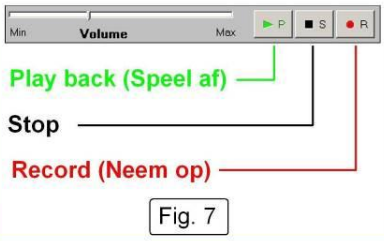


Fig. 7

Figuur 7: afspelen, opnemen en stoppen.

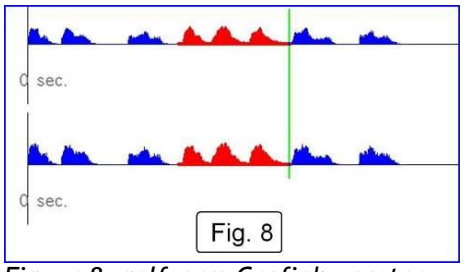
We gaan nu het geluid afspelen. Onder Memo vind je rechts naast de **Volume**-slider (waarmee je het geluid harder of zachter kunt zetten), de bedieningsknoppen voor het afspelen, opnemen en stoppen. Zie figuur 7.
Klik met de muis op **Play-back**. Nu worden de sporen S1, S2 en S3 naar de geselecteerde boxen gestuurd.
Let op: de "digitale band" blijft lopen tot het einde van de sporen (8 minuten). Klik eerst op de **Stop-knop**, als je het geluid opnieuw wilt beluisteren. Maak er een gewoonte van om altijd op de Stop-knop te klikken als je klaar bent met luisteren.

TijdNaald

Je ziet dat tijdens het afspelen een groene, verticale lijn van links naar rechts door het beeld loopt. We noemen dat de TijdNaald. Die wijst aan welk gedeelte van de trilling je op dat moment hoort.
Speel het geluid nog eens af en klik dan ondertussen eens wat op de drukknopjes **L** en **R** van sporen S1 en S2. Je hoort dan op je koptelefoon (of de boxen) links of rechts niets als een knopje grijs is.

Zoom in en Zoom Uit

Het zal je wel zijn opgevallen dat het geroep van de raaf in het grafiek-venster wordt weergegeven door een enkele grillige, blauwe vlakjes. Die blauwe kleur is om aan te geven dat niet de geluidstrilling wordt aangegeven, maar het volumeverloop (vandaar dat je ook geen negatieve uitwijking ziet, want geluidsvolume kan niet negatief zijn). De reden daarvan is dat de tijdschaal van de grafiek zo is ingesteld dat je maar liefst bijna 20 seconde geluid in één blik kunt overzien. Dat overzicht is vaak het handigst als je een nieuw geluid opent. Als je van zo'n lang fragment de geluidstrilling zou willen afbeelden, moest je beeldscherm honderden of zelfs duizenden malen zo lang zijn!
Wil je meer of minder details zien, klik dan op de knoppen **Zoom in** en **Zoom Uit**. Hiermee wordt respectievelijk een kleiner of een groter geluidsfragment afgebeeld. Hoe ver je hebt in- of uitgezoomd wordt aangegeven in het vakje links van de zoomknoppen. Je leest daar bijvoorbeeld: 1024 Ts / pix. Merk op dat, wanneer je steeds verder inzoomt, na 1 Ts / pix de meeteenheid "omklapt": 2 pix / Ts. Wat dat precies betekent wordt in hoofdstuk 3 verklaard.
Als je voldoende inzoomt, wordt de kleur van de grafiek zwart en zie je ook negatieve uitwijkingen: dat is de afbeelding van de trilling zelf!
Experimenteer eens wat met de zoomknoppen. Merk op dat, als je op **Play Back** klikt, de TijdNaald ook sneller of langzamer gaat lopen.



Figuur 8: golfvorm Grafiek-venster.

Speel fragment af

Als je op **Play Back** klikt, wordt het geluid altijd vanaf het begin (tijdstip 0) afgespeeld. Maar het komt vaak voor dat je alleen maar een bepaald fragment wilt horen. Daarom is het belangrijk dat je zo'n fragment snel en eenvoudig kunt aanwijzen en afspelen. Je doet dat door muisklikken in het Grafiek-venster: met de linker muisknop wijs je het begintijdstip van het fragment aan. Dat begintijdstip noemen we **Links**. Met de rechter muisknop wijs je het eindtijdstip aan. We noemen dat **Rechts**. Van het fragment dat je hebt geselecteerd, wordt de golfvorm op het Grafiek-venster aangegeven in rood. Zie figuur 8. Wil je het alleen het rode fragment horen, dan klik je op de knop **Speel fragment af**.

msec, sec, Ts

Zodra je op de linker en rechter muisknop hebt geklikt, wordt de waarde van **Links** of **Rechts** ook in getalvorm aangegeven, in de vakjes onder de zoomknoppen. Die getallen kunnen de tijd uitdrukken ofwel in seconden, ofwel in milliseconden ofwel in sample-tijd (zie hoofdstuk 3). Klik op het knopje rechts van de vakjes **Links** en **Rechts** om een andere tijdeenheid te selecteren.

Y-Sliders

Je hebt er inmiddels waarschijnlijk al even wat mee op en neer geschoven: de Y-sliders, links in de marge van het Grafiek-venster, direct rechts naast de sporen S1, S2 en S3. Je kunt er de $u(t)$-diagrammen van S1, S2 en S3 afzonderlijk mee vergroten of verkleinen in verticale richting.
Merk op dat schuiven met de Y-sliders geen enkel effect op het geluid heeft, evenmin als de zoomknoppen; het zijn allemaal puur grafische instellingen.
Ook de **Volume-**slider bij Play Back doet niks met het geluid zelf. Als je daarmee schuift, klinkt het wel harder of zachter, net zoals wanneer je aan de volumeknop van een versterker draait. Maar die verandering heeft alleen maar betrekking op een instelling van de geluidskaart in je PC, niet op de geluidstrillingen zelf. Wat je moet doen om het geluid zelf te veranderen of te bewerken zien we in hoofdstuk 3.

Wisknoppen

Rechts boven het Grafiek-venster vind je een paneeltje met de naam **Wissen**. Daaronder zie je vier knoppen: **Memo**, **Grafiek**, **Buffers**, **Commando's**. Met de eerste twee wis je respectievelijk de tekst in **Memo** en de afbeelding in het Grafiek-venster.
Het woord ***buffer*** is nogal een computer-begrip. Het betekent: rij van bij elkaar horende getallen. Wat die getallen betekenen of voorstellen, kan van alles zijn. In hoofdstuk 3 lees je hoe je ook geluid kunt voorstellen door een rij getallen. De sporen S1, S2 en S3 zijn in feite niets anders dan buffers. Met de wisknop kun je alle buffers in één keer leeg maken. Er is ook een menufunctie waarmee je zelf kunt aangeven welke buffers je wilt wissen: zie menu **Bewerken** → **Wis geselecteerde buffer(s)**.
Als je bij de opdrachten hieronder eerst de sporen leeg moet maken voordat je er nieuwe geluiden op gaat zetten, lees je:
Klik op **Wissen Buffers**.

Toetscommando's

In menu **Help** vind je een overzicht van alle toetscommando's. Let er even op dat de toetscommando's niet werken als je **Memo** of het **Commando-venster** door een muisklik hebt geactiveerd. Dat is logisch, want dat zijn alle twee tekstvensters, waar je ook in moet kunnen typen.
Klik daarom eerst op de **OK**-knop om de toetscommando's te activeren!

Bedenk ook dat de toetscommando's alleen maar snelle methodes zijn om bepaalde menufuncties uit te voeren. Welke menufuncties dat zijn, vertelt **Help**.

'g': **Menu → Grafiek → Grafiek van floatbuffers**
'h': **Menu → Grafiek → Grafiek van S1, S2 en S3.**

h-toets
Met name de **'h'**-toets zul je vaak gebruiken om WaveWizard op te dragen de trillingen op de sporen S1, S2 en S3 te tekenen in het Grafiek-venster. Als je gebruik maakt van het Commando-venster en van **Start** (zie paragraaf 2.3) dan worden vrijwel altijd automatisch de grafiek van S1, S2 en S3 getekend. In enkele gevallen gebeurt dat niet en verschijnt er geen grafiek. Dan maak je gebruik van de **h-toets** of van menufunctie **Grafiek → Grafiek van S1, S2 en S3.**

2.3 Periode, frequentie, amplitude

Nu je een klein beetje de weg weet met WaveWizard is het een goed moment om een paar dingen die je in de onderbouw bij natuurkunde leerde over geluid nog eens op een rijtje te zetten.

Geluid

Dat geluid een trillingsverschijnsel is, kun je vaststellen door een vinger op je strottenhoofd te leggen en een toon te zingen. Je voelt duidelijk een trilling in de top van je vinger. Dat die trilling zich ook door de lucht voortplant, blijkt als je een velletje papier op een paar centimeter afstand voor je mond houdt en begint te praten: ook dan registreren je vingertoppen duidelijk trillingspatronen in het papier. Kennelijk brengen je stembanden de lucht in trilling en vervolgens de lucht het papier.
Precies dezelfde functie als dat velletje papier heeft het membraan dat in een microfoon zit. De trillingsbeweging daarvan wordt omgezet in een wisselspanning, die je vervolgens met opnameapparatuur kunt vastleggen. Wat er precies wordt vastgelegd is de steeds veranderende uitwijking van het membraan.
Het ***uitwijking-tijddiagram*** ($u(t)$-diagram) is daarvan een grafische weergave. Je zag hierboven al de $u(t)$-diagrammen van de raaf-trillingen

op sporen S1 en S2. Op het *u*(*t*)-diagram wordt de tijd afgebeeld op de horizontale as en de uitwijking op de verticale as.

Periode *T*

De golfvormen van het raaf-geluid laten goed zien dat trillingen heel ingewikkeld zijn. Als je inzoomt tot bijv. 2 Ts / pix, valt het je op dat er, min of meer, een herhalingspatroon in te zien is. Zo'n herhalingspatroon neem je waar als een ***toon***. Je kunt met de toetsen **Page Up** en **Page Down** door de golfvorm "heen wandelen" en zo het herhalingspatroon nog verder bekijken. Als je in een golfvorm geen enkel herhalingspatroon kunt ontdekken, dan hoor je ***ruis***.
De tijdsduur *T* van het fragment dat zich steeds herhaalt noemen we de ***periode***.
T wordt uitgedrukt in seconden. Of een toon een korte of een lange periode heeft, kun je eveneens horen: bij lange periodes hoor je een lage toon, bij korte een hoge.

Definitie periodieke trilling

Een trilling noemen we ***periodiek*** als de golfvorm ervan is opgebouwd uit identieke periodes.
In het *u*(*t*)-diagram van een periodieke trilling kun je tussen de verschillende periodes geen enkel verschil onderscheiden!
In de wiskunde wordt een functie *f(t)* periodiek genoemd als voor elke waarde van *t* geldt dat

$$f(t) = f(t+T) \qquad (1)$$

Bijvoorbeeld als $f(t) = \sin(t)$, dan is de periode $T = 2\pi$, zodat $f(t+T) = \sin(t+2\pi)$ en je weet dat $\sin(t) = \sin(t+2\pi)$.

Met de computer is het kinderspel om periodieke tonen te maken. Maar dat tonen, zoals van de raaf of tonen die je zingt of op een muziekinstrument speelt, zo levendig klinken, komt juist omdat ze niet helemaal zuiver periodiek zijn! Want je zult zo dadelijk snel vaststellen: periodiek = saai!
Daarom spreken we bij tonen van muziekinstrumenten of zang van ***quasi-periodieke*** trillingen. Tonen waarvan de toonhoogte schommelt, of tonen die van volume veranderen zijn quasi-periodiek. Ook tonen met een beetje achtergrondruis zijn quasi-periodiek.

Frequentie *f*

Heel vaak zullen we ook een andere grootheid hanteren om de toonhoogte van een toon uit te drukken: de ***frequentie***: het aantal periodes per tijdseenheid. We duiden de frequentie aan met een kleine letter *f*. De

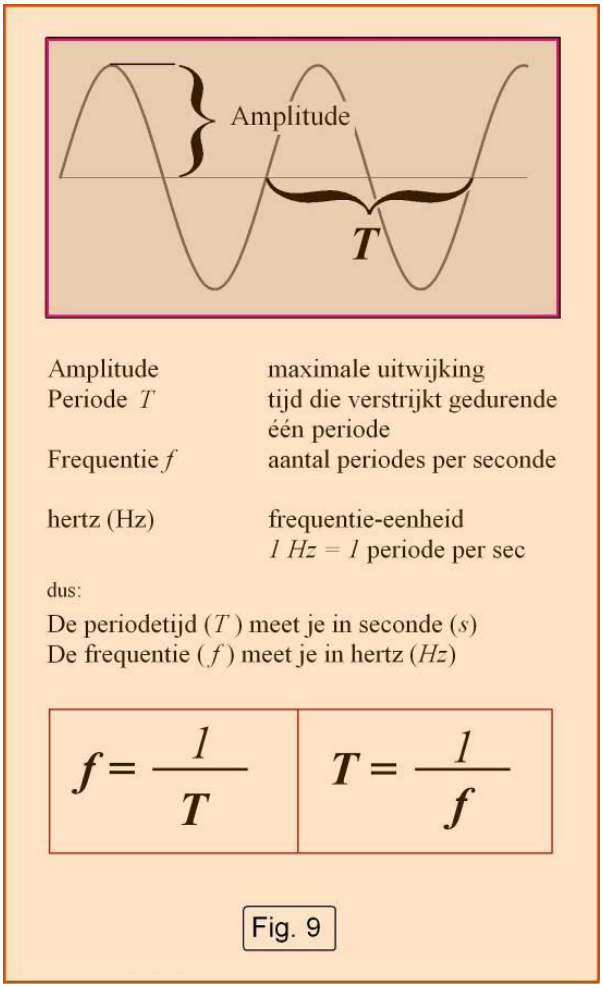
eenheid voor frequentie is de ***Hertz***, afgekort Hz, het aantal periodes per seconde.
Bijvoorbeeld: de lage bromtoon die je soms hoort in een elektrisch apparaat heeft een frequentie van 50 Hz. De poten van een stemvork trillen 440 keer per seconde heen en weer; de frequentie is dus 440 Hz. De toon van een stemvork klinkt veel hoger dan de bromtoon van 50 Hz.

Periode en frequentie

Merk op dat de periode *T* en de frequentie *f* elkaars inversen zijn: als een toon een frequentie heeft van 50 Hz, dan passen er 50 periodes in 1 seconde. Elke periode *T* duurt dan 1 / 50 seconde.
Dus

$T = 1/f$ (2)

In figuur 9 vatten we een paar belangrijke begrippen nog eens samen. Daarbij voegen we ook de term ***amplitude*** die je, net als de andere, al kent uit de natuurkundeles.



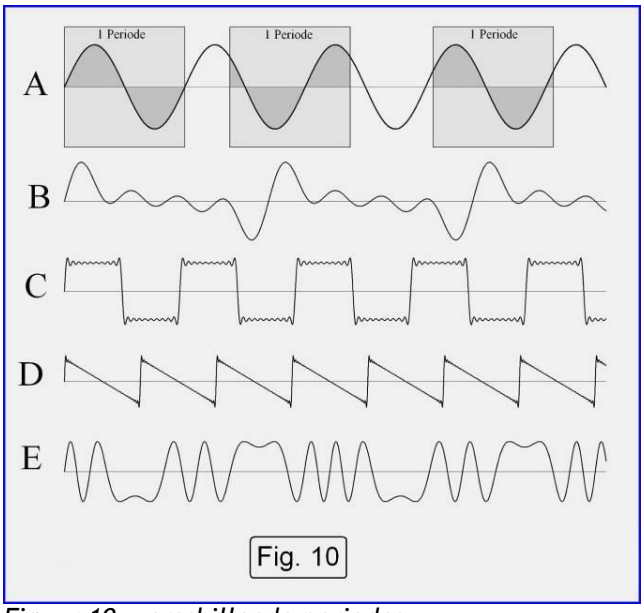
Figuur 9: samenvatting periode en frequentie.

8. Opdracht: raaf-trilling

a. Is de roep van de raaf een periodieke of een quasi-periodieke trilling?
b. Heeft de roep een vaste toonhoogte? Vind je het antwoord daarop ook terug in het $u(t)$-diagram?

9. Opdracht: periode

a. Markeer op drie manieren in figuur 10 met blokken de periode in de golfvormen B t/m E. (zoals in het voorbeeld bij A). Bij welke golfvormen lukt dat niet, als de blokken elkaar niet mogen overlappen?
b. Noteer B t/m E in volgorde van toenemende periodetijd.
c. Noteer B t/m E in volgorde van toenemende frequentie.
d. Leg uit waarom een toon die steeds zachter wordt alleen maar quasi-periodiek kan zijn.

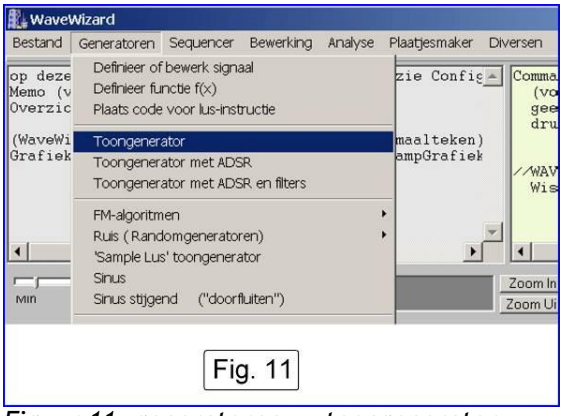


Figuur 10: verschillende periodes.

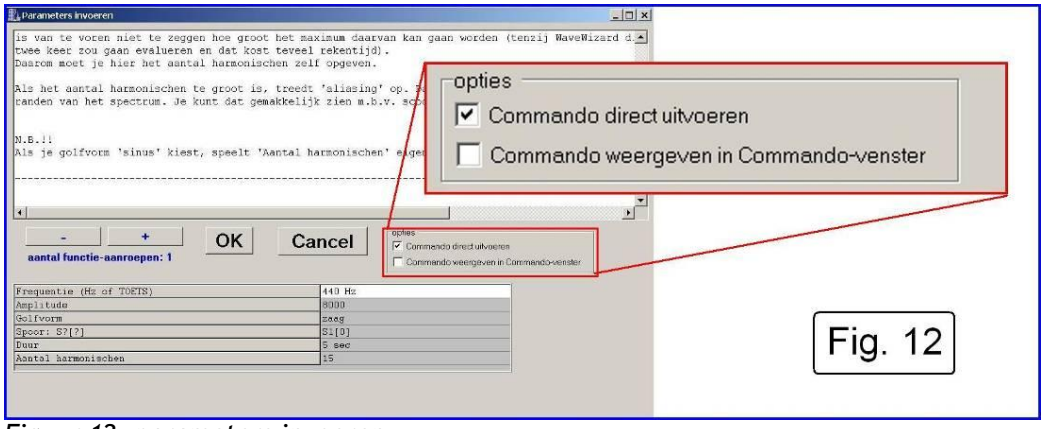
10. Opdracht: toongenerator

We gaan nu een ***sinustoon*** opwekken. Een sinustoon is de toon die je hoort als de periodes van een trilling de vorm hebben van de sinusfunctie. We plaatsen de toon op spoor S1.

a. Klik op **Wissen Buffers**.
Ga met de muis naar menu **Generatoren → Toongenerator**. Deze notatie gebruiken we voortaan voor wat je ziet in figuur 11. Je komt nu op het scherm dat is afgebeeld in figuur 12. Controleer dat in het vak **opties** de vinkjes staan zoals in figuur 12.



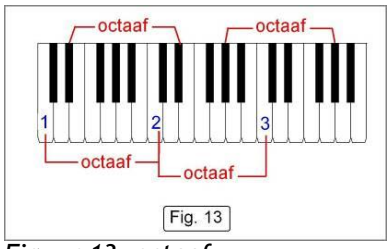
Figuur 11: generatoren → toongenerator.



Figuur 12: parameters invoeren.

Het grote witte vak boven geeft een uitleg van de functie die je hebt aangeroepen.
Daaronder zie je een soort invulformulier. Want om een toon te kunnen opwekken moet WaveWizard een aantal dingen weten, zoals: Op welk spoor moet de toon komen? Hoe lang duurt de toon? Wat is de toonhoogte? Wat is het volume? Daarom moet je eerst dat formulier invullen.
In bijna alle gevallen geeft WaveWizard je gelijk een voorbeeld. Je kunt alle ingevulde instellingen laten staan op één na: er moet een sinustoon gemaakt worden, daarom vul je achter golfvorm in: `sinus`.
Tip 1: gebruik de cursortoetsen Neer (↓) en Op (↑) om snel naar het volgende of vorige veld te gaan.
OK-knop. Je komt dan weer terug op het hoofdscherm.

b. Speel de sinustoon op spoor S1 verschillende keren af. Laat het geluid alleen uit de rechter box komen en daarna uit beide boxen. Zet het geluid harder en zachter met de **Volume**-slider.
c. Hoeveel (hele) periodes van de toon zijn afgebeeld in het grafiek-venster als je inzoomt tot 1 Ts / pix?
d. De streepjes op de tijd-as geven milliseconden aan (1 msec = 1/1000 seconde). Lees hiermee zo nauwkeurig mogelijk de periodetijd *T* af.
e. Druk *T* uit in seconde en bereken daarmee de frequentie *f*. Hoe groot is de afwijking van de werkelijke frequentie?
f. Maak nogmaals een sinustoon aan, maar nu voer je in:
   `Frequentie (Hz)  880 Hz`
   `Spoor: S?[?]  S2[0]`
   De rest van de invoervelden blijft hetzelfde.
   Probeer vóórdat je op **OK** klikt te vertellen wat er gaat gebeuren.
   Speel alleen spoor S2 af op beide boxen.
   Wat is er gebeurd met de periodetijd *T*?
g. Hoe verhouden zich de frequenties van de tonen op spoor S1 en spoor S2?
h. Speel nu spoor S1 af op de linker box en spoor S2 op de rechter. Luister naar de samenklank.



Figuur 13: octaaf.

De frequentieverhouding van twee tonen, f1 : f2, noemen we een interval. Bij het interval dat je hier hoort is f1 : f2 = 1 : 2 . Dit interval wordt octaaf genoemd. (Het octaaf omspant op de piano 8 witte toetsen; octavus = achtste. De begin- en eindtoon van onze toonladder vormen samen een octaaf (do re mi fa sol la si do). Zie figuur 13.

11. Opdracht: boven- en ondergrens toonhoogte

Klik op **Wissen Buffers**.

a. Zet op spoor S1 een sinustoon van 20 sec met Amplitude 8000. Stel de frequentie zo in dat je een zo laag mogelijke bromtoon hoort. Op welke frequentie ben je uitgekomen?
b. Zet op spoor S2 een sinustoon van 20 sec met Amplitude 8000. Stel de frequentie zo in dat je een zo hoog mogelijke pieptoon hoort. Op welke frequentie ben je uitgekomen?
c. Noem redenen waarom je uit de resultaten van vraag (a) en (b) geen al te betrouwbare conclusies kunt trekken over de kwaliteit van je gehoor.

12. Opdracht: amplitude en volume

Klik op **Wissen Buffers** en op **Wissen Commando's**.

Zet op spoor S1 een sinustoon van 20 sec met frequentie 1000 Hz en Amplitude 8000.

Zet op spoor S2 een sinustoon van 20 sec met frequentie 1000 Hz en Amplitude 4000.

Stuur spoor S1 naar alleen de linkerbox en spoor S2 naar alleen de rechter. Beluister ze beurtelings.

a. Waarin verschillen de tonen op spoor S1en S2 qua klank?
b. Waarin verschillen de grafieken van spoor S1en S2?

13. Opdracht: ruis

"Als je in een golfvorm geen herhalingspatroon kunt ontdekken, dan hoor je ruis", las je in paragraaf 2.3 (onder het kopje periode). Dat gaan we nu onderzoeken.

Klik op **Wissen Buffers**.

Ga naar menu:

Generatoren → Ruis (Randomgeneratoren) → Ruis (wit) uniforme verdeling

a. Voer de volgende waarden in:

Buffer	S2
Min	-5000
Max	5000
Vanaf samplenummer (ook 'sec')	0
Aantal samples (ook 'sec')	20 sec
Startgetal Toevalgenerator	0

Maak de grafiek van de ruis op spoor S2 zichtbaar.

b. De ruis duurt 20 seconde. Als je inzoomt tot 1 Ts / pix, wordt daarvan op het scherm slechts de eerste 20 milliseconde (msec) afgebeeld. Hoe ziet het vervolg eruit? Druk op toets **Page Down**. Je ziet nu het tijd-interval van 10 - 30 msec. Druk nog een aantal malen op **Page Down** en probeer regelmaat te ontdekken... (succes!...)
c. Speel het geluid af en beschrijf wat je hoort. Kun je een toonhoogte ontdekken?
d. Nu gaan we ook op spoor S3 ruis zetten, maar wel met een verschil. Voer nu in:

Buffer	**S3**
Min	**-1000**
Max	**1000**
Vanaf samplenummer (ook 'sec')	**0**
Aantal samples (ook 'sec')	**20 sec**
Startgetal Toevalgenerator	**0**

Speel spoor S2 en S3 afwisselend af op resp. de linker en de rechter box. Wat is het verschil in klank? Wat is het verschil in grafiek?

1. Achtergrondinformatie: natuurwetenschappelijke en perceptieve begrippen

In de opdrachten 8 t/m 13 heb je vastgesteld dat de toonhoogte toeneemt met de frequentie en dat het volume toeneemt met de amplitude. Misschien ben je daarom geneigd om te zeggen: "frequentie is toonhoogte en amplitude is volume". In grote lijnen klopt dat ook wel. Maar toch moet je even oppassen. Frequentie en amplitude zeggen iets over het geluid zelf. Het zijn objectieve, natuurwetenschappelijke, meetbare grootheden. Toonhoogte en volume daarentegen zeggen uitsluitend iets over de manier waarop we geluiden horen. Het zijn subjectieve, ***perceptieve*** begrippen: ze hebben betrekking op onze ***waarneming***. Je kunt ze alleen maar "meten" door vragen te stellen aan proefpersonen, vooral aan de proefpersoon die altijd aanwezig is: jezelf.
Zoals gezegd, meestal kloppen onze waarnemingen heel redelijk met de objectieve grootheden. Maar niet altijd. Onderzoek leert bijvoorbeeld dat proefpersonen een toon soms hoger vinden worden als de amplitude toeneemt, terwijl de frequentie toch dezelfde blijft. Dat gebeurt vooral bij zeer luide tonen.
Verderop zullen we nog zo'n derde koppel tegenkomen: spectrum (objectief) en klankkleur of timbre (perceptief).

2.4 Scoop en Spectrum

Beperkingen van statisch $u(t)$-diagram

Tot nu toe hebben we een aantal verschillende tonen gemaakt die één ding gemeen hadden: het waren periodieke tonen. Als je naar het $u(t)$-diagram van een periodieke toon kijkt, dan heb je aan één enkele periode genoeg. De rest is "meer van hetzelfde". In de klank komt dat periodieke karakter

tot uitdrukking doordat de tonen volkomen statisch zijn. De eigenschappen ervan, zoals volume en toonhoogte, zijn constant; er "gebeurt niks". Daarom klinken periodieke tonen nogal saai en onmuzikaal.

In de opdrachten 14 t/m 17 gaan we tonen opwekken, waarvan bijvoorbeeld de toonhoogte of de klankkleur geleidelijk veranderen. Natuurlijk kan het $u(t)$-diagram je daarbij wel van dienst zijn, maar als je een heel klein beetje detail wilt zien, moet je enorm gaan inzoomen en dan vliegt de TijdNaald over het scherm, zodat wat je hoort al binnen een fractie van een seconde buiten beeld valt. Zoom je dan wat meer uit, dan zie je wel een langer fragment in beeld, maar het is niet gedetailleerd genoeg om je echt goed te vertellen met welke zichtbare golfvormverandering de hoorbare klankveranderingen gepaard gaan.
In zulke gevallen heb je meer aan een $u(t)$-diagram dat is uitgevoerd in de vorm van een oscilloscoop. Je vindt die in het **Scoop- en Spectrumvenster**.

Je komt op het Scoop- en Spectrumvenster (figuur 14) door op de **m-toets** te drukken (of, wat precies hetzelfde is, menu **Analyse→ Scoop en Spectrum**).

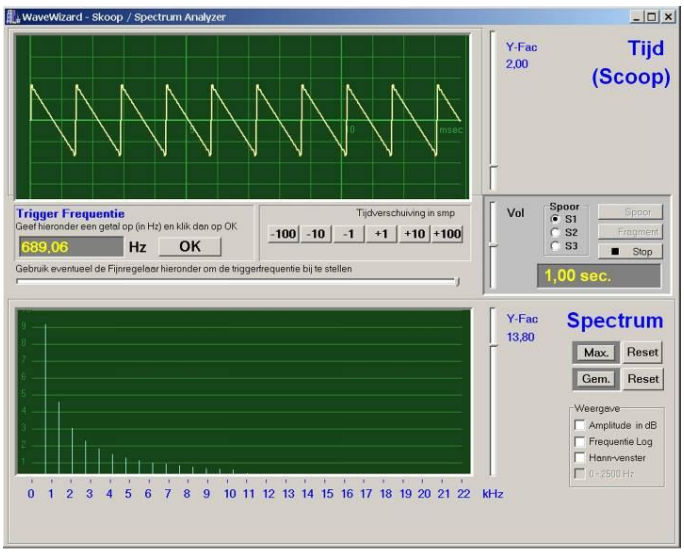


Fig. 14

Figuur 14: scoop- en spectrumvenster.

Scoop

Net zoals op het hoofdscherm van WaveWizard tref je ook hier twee knoppen voor het afspelen van het geluid: het hele spoor (**Spoor**) of een selectie (**Fragment**). Klik je op één van beide, dan stuurt WaveWizard het geluid niet alleen naar de geluidskaart, maar plaatst ook 20 "filmbeelden" per seconde van het $u(t)$-diagram en geeft die beelden razendsnel na elkaar weer in het bovenste venster (Tijd). Terwijl je luistert naar het geluid, zie je tegelijkertijd in het Scoop-venster een bewegend beeld van

het $u(t)$-diagram van het geluid dat je op dat moment hoort. Vandaar de naam Scoop: het is een simulatie van de werking van de ***oscilloscoop*** die je ongetwijfeld in het natuurkundelokaal wel eens aan het werk hebt gezien.

Spectrum

Het spectrum is een grafiek waarbij op de horizontale as de frequentie wordt afgebeeld en op de verticale as de amplitude. Een sinustoon wordt er op weergegeven als een "paaltje": een ***spectraallijn***. Hoe hoger de sinustoon des te verder die rechts op de frequentie-as staat. Hoe harder de sinustoon des te langer de spectraallijn. (Je zou het spectrum ook kunnen noemen: het uitwijking-frequentie-diagram of *u.f.*-diagram, maar spectrum is meer ingeburgerd.)

Commando-venster

Van nu af gaan we ook gebruik maken van het commandovenster. Je zult snel merken dat dat veel sneller en prettiger werkt dan via de menu's.

14. Opdracht: commando-venster / preset openen / spectrum sinustoon
Ga naar menu **Bestand** → **Open Preset** en open ►Preset_H2_TweeSinustonen.txt. Er verschijnt nu in het commandovenster een tekst. Deze tekst noemen we een preset. Daarin staan menucommando's beschreven, zoals je die zelf in vorige opdrachten met de hand invoerde. Nu je eenmaal weet hoe die menufuncties werken, maak je verder uitsluitend nog gebruik van presets, dat scheelt je veel tijd.
Wel zal in die preset-teksten meestal op sommige plaatsen puntjes of een vraagteken staan dat je eerst moet vervangen door een getal of een formule voordat je de preset kunt uitvoeren.
Let op! WaveWizard geeft een foutmelding zolang er puntjes of vraagtekens staan in de commando-tekst!
Dat uitvoeren doe je door met de muis op de knop **Start** te klikken. In deze preset bestaat de tekst uit drie alinea's die elk een commando bevatten. Het eerste commando heet **`Wis buffers`**; het vertelt WaveWizard dat de sporen S1, S2 en S3 moeten worden gewist. De twee daarop volgende commando's maken elk een toon aan met de toongenerator die je in opdracht 10 (toongenerator) hebt leren kennen.

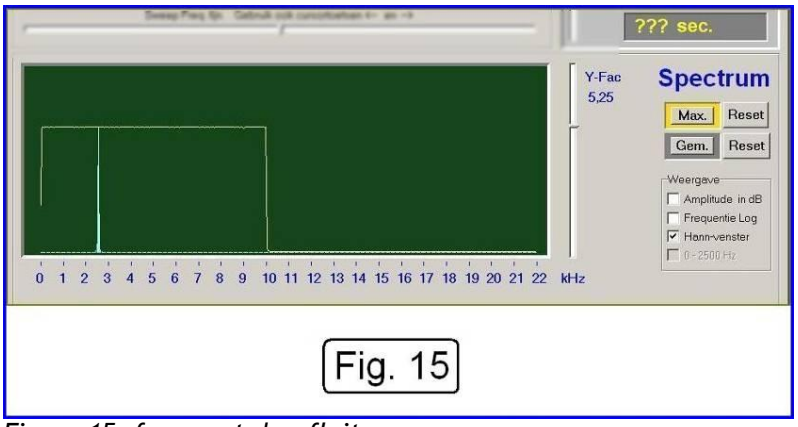
a. Vervang bij de eerste toongenerator de vraagtekens zodat je een sinustoon krijgt met een frequentie van 1000 Hz en een amplitude 8000.
b. Vervang bij de tweede toongenerator de vraagtekens zodat je een sinustoon krijgt met een frequentie van 3000 Hz en een amplitude 4000.
c. Als je klaar bent met invullen, klik je op de knop **Start**

d. Druk op de **m-toets** (of, wat precies hetzelfde is, ga naar menu **Analyse→ Scoop en Spectrum**). Je komt nu op de Scoop-pagina. Klik op **Speel af** en selecteer afwisselend spoor S1 en spoor S2. Wat zie je in het Scoop- en het Spectrum-venster (scoop: periode en amplitude; spectrum: frequentie en amplitude?

15. Opdracht: "doorfluiten"

We gaan nu een sinustoon maken waarvan de frequentie heel langzaam lineair toeneemt. In audiostudio's gebruiken ze zo'n stijgende toon om apparatuur mee "door te fluiten": testen hoe een apparaat reageert op "elke" sinus. We komen daar in hoofdstuk 5 op terug, waar je zelf filters gaat doorfluiten.

a. Ga naar menu **Bestand** → **Open Preset** en open ►Preset_H2_Doorfluiten.txt. Klik op **Start.**
b. Ga naar het Scoop-Spectrumvenster. Speel de toon af. Klik op **Max** en speel nogmaals de stijgende toon af. **Max** houdt de maximumwaarden van de spectraallijnen bij. Hoe kun je zien dat de frequentie lineair toeneemt?
c. Op welk moment is figuur 15 “gefotografeerd”? Met andere woorden: wat moet je invullen in het vakje van de tijdsaanduiding rechtsboven in de figuur (op 1 seconde nauwkeurig)?



Figuur 15: fragment doorfluiten.

16. Opdracht: toon met veranderende golfvorm

Nu gaan we een toon maken waarvan de golfvorm begint als sinus en dan langzaam steeds complexer wordt.

a. Ga naar menu **Bestand** → **Open Preset** en open ►Preset_H2_ToonSteedsHelderder.txt. Klik op **Start.**
b. Speel de toon op S2 af en beschrijf wat je hoort. Let daarbij op de klankkleur (dof? scherp?).
c. Beschrijf de beelden in het scoop- en het spectrum-venster.
d. Vervang rechts van **`Afsnijfrequentie (Hz)`** de expressie **`a*n + b`** door de expressie
   `2400 + 2000*sin(pi*n*Ts).`

Let goed op de schrijfwijze (grote en kleine letters)!

Klik weer op **Start** en beschrijf wat je hoort en ziet in het scoop- en het spectrum-venster.

17. Opdracht: spectrum van stijgende zaagtandtoon

a. Ga naar menu **Bestand** → **Open Preset** en open ►Preset_H2_StijgendeZaagtand.txt.

b. Nadat je op **Start** hebt geklikt verschijnen er tonen op spoor S1, S2 en S3 als volgt:
S1: zaagtand-toon met frequentie 500 Hz
S2: zaagtand-toon met frequentie 1000 Hz
S3: zaagtand-toon met een frequentie die langzaam toeneemt van 500 tot 1000 Hz.
Kijk en luister eerst alleen naar S1 en S2. Wat verwacht je te zien op S3?

c. Beschrijf wat je ziet in het scoop- en het spectrum-venster als je S3 selecteert.

d. Kun je verklaren waar dat `Aantal harmonischen` op slaat? (Zo niet, maak de toon dan eens met 10 of met 3 harmonischen en kijk naar het spectrum.) Het kan zijn dat je **Y-fac** moet aanpassen om de spectraallijnen verticaal voldoende op te schalen.

2.5 Grondtoon, boventonen, harmonischen

In opdrachten 16 en 17 heb je een merkwaardig verschijnsel gezien in het spectrum: die zaagtandtoon blijkt in het spectrum meerdere pieken te vertonen, terwijl je toch maar één toon hoort....
Omdat één piek in het spectrum een enkele sinustoon weergeeft, moet de zaagtandtoon zijn opgebouwd uit een aantal gelijktijdig klinkende sinustonen.
Als je afwisselend een sinustoon en een zaagtandtoon met dezelfde toonhoogte hoort en je kijkt in het spectrum, dan zie je dat de meest linkse spectraallijn op z'n plaats blijft liggen; zodra de zaagtand klinkt, komt er een aantal pieken bij en tegelijkertijd wordt de klank scherper. Ga dat na in opdracht 18.

18. Opdracht: sinus-zaagtand afgewisseld

a. Ga naar menu **Bestand** → **Open Preset** en open ►Preset_H2_Sinus_Zaag.txt. Klik op **Start** en ga naar het **Scoop- en Spectrumvenster**, beluister de toonafwisseling en bekijk de bijbehorende golfvormen en spectra.

b. In de commando-tekst staan twee Toongenerator-codeblokken, de eerste voor de sinustoon en de tweede voor de zaagtandtoon. De onderste instelling, `Aantal harmonischen`, is ingesteld op 50. Maak daar achtereenvolgens 25, 10, 5 en 1 van en klik telkens op **Start** Wat is het verschil in klank en in spectrum?

Die meest linkse toon in het spectrum, die zowel in het sinusspectrum als in het zaagtandspectrum altijd op z'n plek blijft, correspondeert met de toonhoogte die je waarneemt en meezingt. We noemen dat de ***grondtoon***.

Bij de zaagtand zag je er een aantal spectraallijnen bijkomen. Die noemen we de ***boventonen***. Boventonen zijn (evenals de grondtoon trouwens) altijd sinustonen, want elke spectraallijn vertegenwoordigt precies één sinustoon.
Wat verder opvalt, is dat de spectraallijnen op regelmatige afstanden van elkaar staan. Het blijkt dat dit altijd het geval is als een toon periodiek is. Onthoud de volgende regel:

Als een toon periodiek is, dan zijn de frequenties van de boventonen veelvouden van die van de grondtoon. We noemen dan de boventonen ***harmonischen***.

Anders gezegd: dat een toon periodiek is, kun je zowel zien aan de tijdgolfvorm (scoop) als aan het spectrum:

1. De golfvorm is een perfecte herhaling van één periode (volgens definitie van periodiciteit, zie paragraaf 2.3);
2. In het spectrum liggen de spectraallijnen op gelijke afstanden van elkaar. In opdracht 17 bekeek je het spectrum van een zaagtandtoon waarvan de toonhoogte toenam. Je zag de spectraallijnen "uitwaaieren" over het spectrum, maar op elk tijdstip bleven hun onderlinge afstanden gelijk, net zoals onderlinge afstanden tussen de vouwen van een harmonica-deur tijdens het openschuiven ook steeds gelijk blijven.

Voorbeeld
Een zaagtandtoon met een grondtoon van 300 Hz is periodiek en heeft dus harmonische boventonen met frequenties 300, 600, 900,... Hz.
Harmonische nummer 17 vind je in het spectrum als de 17de spectraallijn en heeft een frequentie van $17 \cdot 300 = 5100$ Hz.
Als de grondtoon toeneemt tot 400 Hz, dan waaieren de harmonischen uit tot 400, 800, 1200, ... Hz.

Verder lezen

Vanwaar die samenhang tussen periodiek-zijn en harmonisch-zijn? Waarom overheerst in periodieke, harmonische tonen zo duidelijk de grondtoon? Waarom is het voor muziek zo belangrijk dat tonen harmonisch zijn? Daarover kun je meer te weten komen op►URL6 (behoort niet tot de stof van deze module en wordt niet getoetst).

2.6 Sonogram

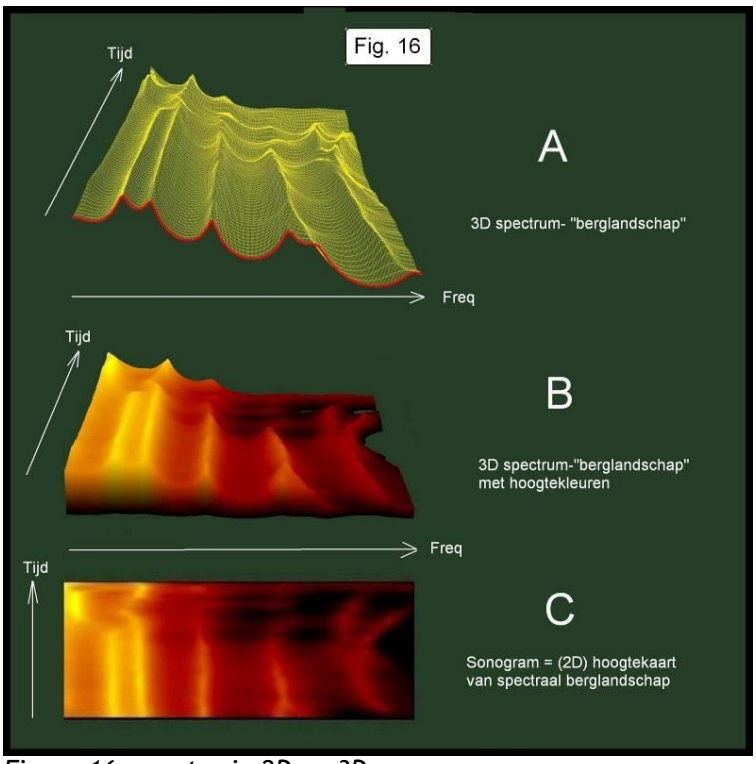
In de opdrachten van paragraaf 2.4 en 2.5 heb je tonen bestudeerd waarvan hetzij de toonhoogte, hetzij de klankkleur veranderde. Die veranderingen waren mooi te zien in het scoop- en het spectrumvenster. Maar dan alleen als je eerst op **Play** drukte en alleen zolang je iets hoorde. Scoop en spectrum zijn animaties, "filmpjes". Toch kun je veranderingen in de toon ook afbeelden in één enkel plaatje. Natuurlijk doet het $u(t)$-diagram dat ook al, maar er is nog een veel duidelijker afbeelding mogelijk die een combinatie is van tijd en spectrum en die je een lang geluidsfragment van bijvoorbeeld 10 of 20 sec (of langer) in één oogopslag laat zien. Zo'n afbeelding noemen we een ***sonogram***.

Laten we eerst even resumeren:

1. de afbeelding in het scoopvenster is een (steeds veranderend) ***u(t)-diagram***: horizontaal de tijd en verticaal de uitwijking
2. de afbeelding in het spectrumvenster toont het (steeds veranderend) ***spectrum***: horizontaal de frequentie en verticaal de uitwijking.

Wat is een sonogram?

In het spectrumvenster wordt 20 keer per seconde een spectrum afgebeeld. Steeds wordt eerst het vorige plaatje gewist, voordat een nieuw wordt getekend. Maar je zou ook een 3D figuur kunnen maken door al de spectrumplaatjes niet uit te vegen, maar "achter elkaar" te leggen, zodat je een soort piek- of berglandschap krijgt.
Dit is de zien bij A in figuur 16. De spectrum-as loopt van links naar rechts, net zoals in het spectrumvenster, en de tijd-as loopt van voor na achter "het papier in". (Het eerste spectrum (op tijdstip 0) is vooraan in de figuur te zien en in rood aangegeven.)
Bij B is datzelfde spectrale berglandschap te zien, maar dit is geen



Figuur 16: spectra in 2D en 3D.

"ijzerdraad-"model, zoals A, maar een gekleurd "kaarsvet-"model, waarvan de kleur verloopt met de hoogte: de pieken zijn helder en geel, de dalen zijn donker en roodachtig.
In C kijk je pal bovenop dit kaarsvetlandschap van B; de dieptedimensie is weggevallen, zodat je een 2D hoogtekaart overhoudt, waarbij kleur de hoogte in elk punt aangeeft, precies zoals je dat van atlassen gewend bent. Zo'n hoogtekaart van een spectraal landschap noemen we een ***sonogram***: het is een 2D-figuur, waarvan tijd en frequentie de dimensies zijn en waarvan de amplitude ("hoogte") van elk punt wordt aangegeven door de helderheid of kleur.

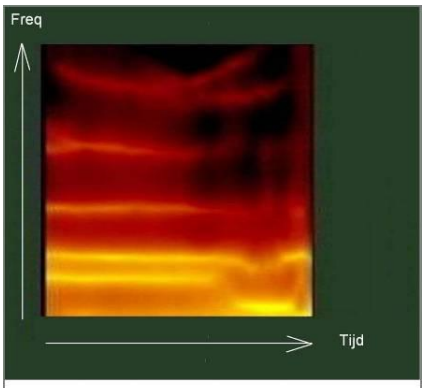


Fig. 17

Figuur 17: sonogram, tijd uitgezet tegen frequentie.

Afspraak: oriëntatie sonogram

Merk op dat in C de horizontale as de frequentie weergeeft en de verticale as de tijd. Zo wordt de overeenkomst met de plaatjes van het spectrumvenster benadrukt. Toch zie je vaak liever de tijd op de horizontale as, en de frequentie dan maar op de verticale. Zie figuur 17. In dit plaatje loopt de tijd van links naar rechts en de frequentie loopt op van beneden naar boven. Net zoals in het notenschrift; langs de notenbalk staan opeenvolgende noten van links naar rechts; hoe hoger een noot des te hoger die op de balk wordt weergegeven.
We spreken af dat we voortaan in sonogrammen de tijd horizontaal afbeelden en de frequentie verticaal, zoals in Figuur 17.

19. Opdracht: sonogram van een sinustoon

a. Ga naar menu **Bestand** → **Open Preset** en open
►Preset_H2_Sinustoon_Sonogram.txt.
Klik op **Start**. Beluister en bekijk de toon in het spectrumvenster. Hoe verwacht je dat het sonogram eruit zal zien?

b. De tekst in het commandovenster eindigt met een instructie voor het sonogram. Maar die instructie werd niet uitgevoerd, omdat er een uitroepteken voor het commando `Sonogram` staat (daarover meer in hoofdstuk 3). Het enige wat je nu moet doen, is dat uitroepteken verwijderen en opnieuw op **Start** klikken. Nu wordt het sonogram wèl getekend. Klopte je antwoord bij (a)?

c. De vierde tekstregel in het commandovenster luidt: `f = 1000`.
Als je daarvan maakt: `f = 10000`, hoe zal het sonogram er dan uitzien?

d. Verander de tekstregel:
```
Bewerking   S1[n] = 8000 * sin(w*n)
```
in:
```
Bewerking   S1[n] = (b * a^n) * sin(w*n)
```
Vergeet de haakjes ( ) niet!

Zet weer een uitroepteken voor **`Sonogram`**.
Klik op **Start**. Kijk naar de golfvorm van S1. Beluister en bekijk de toon ook in het spectrumvenster. Hoe verwacht je dat het sonogram eruit zal zien?

e. Verwijder het uitroepteken en klik weer op **Start**. Had je gelijk?

20. Opdracht: sonogram van een zaagtand- en een parabooltoon

a. Ga naar menu **Bestand** → **Open Preset** en open
►Preset_H2_Zaagtand_Parabool_Sonogram.txt.
Klik op **Start**. Er worden achter elkaar twee tonen met dezelfde toonhoogte maar met verschillende golfvorm gezet, eerst een zaagtandgolf en vervolgens een paraboolgolf.
Beluister en bekijk de tonen in het Spectrumvenster.
Hoe verwacht je dat het sonogram eruit zal zien?

b. Hoeveel lijnen tel je in het spectrum en het sonogram?

c. Vul bij de instelling van de **`Toongenerator`** in:
`Aantal harmonischen  5.`
Klik op **Start** en bekijk het sonogram opnieuw. Hoeveel lijnen tel je nu?

d. Is de klank veranderd? Zo ja, hoe omschrijf je die verandering?

e. Plaats de muis over de onderste horizontale lijn in het sonogram. Geef een linker muisklik: welke frequentie lees je af?

f. Vul bij de instelling van de **`Toongenerator`** in:
`Frequentie (Hz of TOETS)  1000 Hz`
Klik op **Start** en bekijk het sonogram opnieuw. Wat is er veranderd?

g. Vul bij de instelling van de **`Toongenerator`** in:
`Amplitude  b * a ^ n`
Klik op **Start** en bekijk het sonogram opnieuw. Wat is er veranderd?

21. Opdracht: sonogram van stijgende tonen

a. Ga naar menu **Bestand** → **Open Preset** en open
►Preset_H2_Doorfluiten_Sonogram.txt.
Klik op **Start**. Dit is de "doorfluit"-toon van opdracht 15. Beluister en bekijk de toon in het Spectrumvenster. Hoe verwacht je dat het sonogram eruit zal zien?

b. Beantwoord dezelfde vraag voor
►Preset_H2_StijgendeZaagtand_Parabool_Sonogram.txt. Let op dat "uitwaaieren" van de boventonen, waarbij de onderlinge frequentie-afstanden op elk tijdstip gelijk zijn.

c.* Door slechts één cijfer in de commandocode te veranderen, verdwijnen die knikken op het tijdstip t = 2,5 sec en blijven de spectraallijnen in dezelfde richting lopen als in het tijdinterval 0 - 2,5 sec. Heb je enig idee? (We lopen nu vooruit op de stof van hoofdstuk 4! Je hoeft dit nu nog niet te kunnen!)

d.* Je ziet in het sonogram twee verticale lijnen lopen. Met welk audioverschijnsel corresponderen die, denk je? (We hebben ook dit niet behandeld! Zie paragraaf 5.2)

22. Opdracht: sonogram van (galmende) raaf

a. Ga naar menu **Bestand** → **Openen .WAV** en open ►Raaf.wav.
b. Ga naar menu **Bestand** → **Open Preset** en open ►Preset_H2_Raaf_Sonogram.txt. Beluister het geluid en bekijk het verloop van het spectrum in het Spectrumvenster.
c. Waaruit blijkt dat de toonhoogte van de raaf varieert?
d. Waaruit blijkt dat de raafstem een harmonische toon produceert?
e. Je hoort tussen sommige roepen even een zachte galm. Hoe uit zich die in het sonogram?

23. Opdracht: sonogram van een toon die voldoet aan de klankwet

a. Ga naar menu **Bestand** → **Open Prese**t en open ►Preset_H2_Klankwet_Sonogram.txt.
Klik op **Start**. Dit is de toon van opdracht 5b (klankwet demo) uit paragraaf 1.3. Beluister de toon en bekijk het sonogram.
b. Waaruit blijkt dat de toon een constante toonhoogte heeft?
c. Waaruit blijkt dat de toon zacht begint, steeds harder wordt en daarna steeds zachter?
d. Hoe blijkt dat de klankkleur verandert met het volume?

24. Opdracht: sonogram van toon akoestische gitaar

a. Ga naar menu **Bestand** → **Openen .WAV** en open ►Gitaar_hoge_E.wav.
b. Ga naar menu **Bestand** → **Open Preset** en open ►Preset_H2_Gitaar_hoge_E_Sonogram.txt.
c. Je hoort de toon aanvankelijk enkele malen zachter en daarna weer iets harder worden, maar op de lange duur geleidelijk volledig uitsterven. Hoe vind je dat terug in het sonogram?
d. Waaruit blijkt dat de toon steeds doffer wordt?
e. Voldoet de toon aan de klankwet?
f. Klik met de muis op de grondtoon. Welke frequentie lees je af?
g. Klik met de muis op tiende harmonische (N.B. de grondtoon is de eerste harmonische). Welke frequentie lees je af? Hoe hoog moet de frequentie van de grondtoon zijn als die tiende lijn inderdaad een harmonische is?

De opdrachten 23 en 24 hebben gedemonstreerd dat het sonogram heel geschikt is om te kijken of een toon voldoet aan de klankwet. Dat is een belangrijke eerste stap gezet op de weg naar ons einddoel!

Samenvatting

- Geluid kun je op drie manieren grafisch voorstellen, namelijk door het *u(t)*-diagram, door het spectrum, dat trillingen afbeeldt als een mix van sinustonen en door het sonogram.
- *u(t)*-diagram en spectrum kun je met de computer weergeven als bewegende beelden (scoop- en spectrumvenster) zodat beeld en geluid, horen en zien, veel beter aan elkaar gekoppeld worden en meer inzicht geven in trillingseigenschappen.
- Elke geluidstrilling waarvan de golfvorm niet sinusvormig is, is opgebouwd uit een mix van sinustonen met verschillende frequenties en amplitudes (en beginfases). Deze sinustonen noemen we boventonen.
- Een periodieke trilling is harmonisch, dat wil zeggen: heeft boventonen waarvan de frequenties veelvouden zijn van die van de grondtoon. De boventonen van een periodieke trilling noemen we harmonischen. Ze liggen in het spectrum op gelijke afstanden van elkaar.
- Door middel van het sonogram kunnen we zien of een toon voldoet aan de Klankwet.

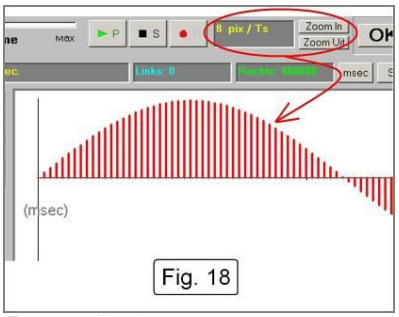
Woordenlijst

amplitude	§4
boventoon	§1. §5
buffer	§2
commando-venster	§2
doorfluiten	§4
frequentie	§3
grafiek-venster	§2
grondtoon	§5
harmonische	§5
Hertz	§3
Hz	§3
interval	§3
klankkleur	§1
links	§2
memo	§2
octaaf	§3
oscilloscoop	§4
paraboolgolf	§6
perceptief	§3
periode	§3
periodiek	§3
quasi-periodiek	§3
rechts	§2
ruis	§3
scoop	§4
sinustoon	§3
spectraallijn	§4
spectrum	§1, §4, §6
spectrumvenster	§4
speel fragment af	§2
sonogram	§1, §6
spoor	§2
stereo	§2
timbre (klankkleur)	§1
toetscommando	§2
toon	§3
uitwijking-tijddiagram	§1, §3, §4, §6
volume-slider	§2
waarneming	§3
Y-slider	§2
zaagtandgolf	§4, §5, §6
zoom in	§2
zoom uit	§2

3 Sampling en de wiskundige representatie van digitaal geluid

3.1 Inleiding

We gaan ons nu bezig houden met het bewerken van geluid. In paragraaf 1.6 las je al dat we drie bewerkingen, versterken, mixen en echo's toevoegen, kunnen beschouwen als fundamenteel, omdat ze in combinatie een filter opleveren. Je hebt voorbeelden gehoord die illustreerden hoe onmisbaar filters zijn voor sound design.



Figuur 18: inzoomen op golfvorm.

3.2 Sampling

Misschien heb je inmiddels al ontdekt dat wanneer je, op de hoofdpagina van WaveWizard, een aantal keer hebt geklikt op **Zoom In**, de golfvorm plotseling niet meer wordt weergegeven als een ononderbroken kromme, maar als een aantal verticale lijntjes van verschillende lengten. Zie figuur 18. De toppen daarvan geven de contour van de golfvorm aan. De lengte van elk lijntje representeert een getal dat we voortaan een ***sample*** noemen. Bij alle geluiden die je tot nu toe met WaveWizard maakte en hoorde, deed de computer niets anders dan een lange rij samples in het geheugen plaatsen. Toen je het bestand ►Raaf.wav opende, las de computer een lange rij getallen die op de harddisk staan en plaatste die in het geheugen. Toen je met de toongenerator een sinustoon maakte, rekende de processor een aantal sinussamples uit en plaatste die eveneens in het geheugen. Verder gebeurde er niets. Klik je vervolgens op de afspeelknop, dan krijg je niettemin een echt geluid te horen, een echte raaf of een echte sinustoon!

DA-converter

Hoe kunnen getallen een geluid representeren? Zou je niet op z'n minst een heel ander geluid moeten krijgen? Immers, als je kijkt naar die lijntjesgrafiek van de samples, dan zie je dat tussen elke twee lijntjes de uitwijking nul is; kennelijk is het daar heel eventjes "stil". De lijntjesgrafiek suggereert dat bij elke sample de conus van de luidspreker zeer abrupt naar voren (de + uitwijking) of naar achteren (de − uitwijking) schiet en vervolgens weer razendsnel terug springt naar de evenwichtstand. Dat is heel iets anders dan wanneer de uitwijking van de conus de weg volgt die de ononderbroken golf aangeeft. Dus de vraag is:

waarom hoor je computerklanken eigenlijk als gewoon geluid en niet een soort afgrijselijk geratel of geknetter?
Dat komt omdat de computer die samplegetallen, als je klikt op afspelen, domweg opstuurt naar de geluidskaart! Daarop vind je een chip, de ***Digitaal-Analoog-converter (DA-converter)***, die de stiltes tussen de samples zodanig weet "op te vullen" dat je weer de oorspronkelijke, ononderbroken golfvorm terug krijgt! Dus je hoort bij afspelen niet de samples zèlf, maar gewoon het "echte" geluid dat op basis van de samples wordt samengesteld.

AD-converter

Als je met de toongenerator een toon opwekt, slaat de computer aan het rekenen. Het resultaat daarvan is een lange rij getallen, samples, die in het RAM-geheugen zijn opgeslagen. Maar hoe zit 't nu met een willekeurig microfoongeluid, zoals de raaf, hoe zet je dat om in zo'n reeks samples? Dat wordt gerealiseerd door een andere chip op de geluidskaart, de ***Analoog-Digitaal converter (AD-converter)***. Een microfoon die op de "Mic"-ingang van de geluidskaart wordt aangesloten levert een ononderbroken wisselspanning. De AD-converter registreert op regelmatige tijdstippen de uitwijking daarvan en slaat die in getalvorm op in het geheugen. Dat omzettingsproces heet ***sampling*** (to sample = bemonsteren, meetwaarden verzamelen).

Bij sampling wordt een ***continue*** trilling omgezet in een ***discrete*** trilling. Een discrete trilling is niets anders dan een (lange) reeks samples (meetwaarden).

Daarbij is de uitwijking dus slechts bekend voor een aantal afzonderlijke tijdstippen (de oorspronkelijke betekenis van het woord discreet is ‘afzonderlijk’). Hoe de uitwijking verloopt gedurende de tijd tussen twee meetwaarden wéét je simpelweg niet! Wat we hierboven nogal losjes opmerkten, namelijk "dat tussen elke twee lijntjes de uitwijking nul is", is dan ook eigenlijk niet juist: je behoort te zeggen dat de uitwijking daar onbekend, niet-gedefinieerd is!

Hoe weet de DA-converter die ontbrekende informatie over de uitwijking tussen twee samples dan te vinden? Hoe kan uit die samplereeks weer een perfect klinkende, continue golf tevoorschijn worden getoverd?
Bij sampling - het verzamelen van meetwaarden van de uitwijking - kan de geluidskwaliteit slechts afhangen van twee dingen:

1. Hoe vaak meet een AD-converter de uitwijking, dus hoeveel samples worden er per seconde geproduceerd?
2. Hoe nauwkeurig is de meting?

Merk op dat deze twee dingen betrekking hebben op respectievelijk de horizontale (tijd) en de verticale (uitwijking) dimensie van het $u(t)$-

diagram. Je kunt dus wat korter spreken van de "horizontale en verticale nauwkeurigheid". Het blijkt dat beide hun eigen effect hebben op de geluidskwaliteit. We bespreken ze hieronder apart.

(1) Hoe vaak? Samplingfrequentie F_s

Het aantal metingen dat je per seconde verricht, drukken we uit in de ***samplingfrequentie F_s***. Het aantal samples dat je per seconde neemt is dus gelijk aan F_s.

Met behulp van de samples is een perfecte reconstructie mogelijk van de oorspronkelijke, continue golfvorm ***als de tonen die gesampled worden geen boventonen bevatten waarvan de frequenties hoger zijn dan de helft van de samplingfrequentie F_s.***

Dit is een fundamentele stelling in de digitale signaalverwerking die bekend staat als de ***stelling van Nyquist*** (we zullen die hier niet bewijzen, ►URL1). De AD-converter van je geluidskaart zorgt er voor dat automatisch alle tonen van het op te nemen geluid die hoger zijn dan 0,5 F_s *eerst* worden weggefilterd. Pas daarna vindt het eigenlijke samplingproces plaats.

Voorbeeld 1

Je wilt een sinustoon van 1000 Hz samplen. Dan moet de sampling frequentie F_s minstens 2000 Hz zijn. Het aantal samples van een periode is dan gelijk aan F_s / 1000 = 2.

Voorbeeld 2

Je hebt een zaagtandgolf van 800 Hz en beschikt over een AD-converter met een sampling frequentie F_s = 10.000 Hz. De zaagtandgolf is een harmonische toon met frequenties 800, 1600, 2400 ....; de reeks gaat in principe oneindig door. De AD-converter beschikt over een ingebouwd filter dat ervoor zorgt dat alle frequenties boven 0,5 F_s = 5000 Hz worden weggefilterd. Je houdt dan een zaagtandgolf over van 5000 / 800 = 6 harmonischen: 800, 1600, 2400, 3200, 4000 en 4800 Hz. Die gefilterde toon wordt vervolgens gesampled.

Aangezien onze oren tonen kunnen horen tot zo'n 20.000 Hz, moet je, om te voldoen aan de stelling van Nyquist, een samplingfrequentie F_s kiezen die minstens het dubbele daarvan is. Daarom is voor audio-cd een standaard samplingfrequentie afgesproken van 44100 Hz. Ook in WaveWizard is F_s daar op ingesteld.

Tenzij anders wordt vermeld, gaan we er in het vervolg van uit dat de samplingfrequentie F_s = 44100 Hz.

F_s en T_s

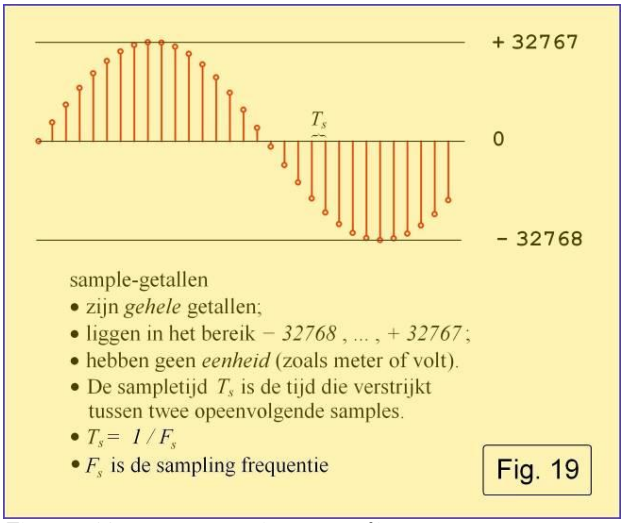
De samplingfrequentie F_s is dus het aantal samples dat per seconde van een geluid wordt genomen. In hoofdstuk 2 zagen we dat de frequentie en periode elkaars inverse zijn: $f = 1/T$ en $T = 1/f$.
Zo ook kunnen we spreken van de ***sampletijd*** T_s als de inverse van de samplingfrequentie F_s.

De sampletijd T_s is de tijd die verstrijkt tussen twee opeenvolgende samples.

Als je op de hoofdpagina van WaveWizard zo ver inzoomt op de golfvorm dat je losse sample-paaltjes ziet, dan ga je bijna automatisch een tijdsduur uitdrukken in een aantal samples. Je zegt dan bijvoorbeeld: "de periode van deze zaagtandgolf is 19 sample-paaltjes". Dat wil dan zeggen dat de periode een duur heeft van 19 T_s seconde. Als
F_s = 44100 Hz, dan is dat 19 / 44100 sec. Daarom zullen we T_s heel vaak tegenkomen.

(2) Hoe nauwkeurig? Het aantal meetniveaus

Met een liniaal waarop een millimeterverdeling is aangebracht, kun je nauwkeuriger meten dan met een liniaal met een centimeterverdeling. Zo is het ook met de metingen die AD-converters uitvoeren. De allereerste converters in de jaren '70 sampleden de uitwijking in 64 meetniveaus. Dus de uitwijking was dan ofwel 0, ofwel 1, ofwel 2, 3, ... 63. En dat leverde een erg ruis-achtige weergave van het geluid op, net zoals bij een oude grammofoon of een radio met een slechte ontvangst. Tegenwoordig heeft een AD-converter, zoals op je geluidskaart, maar liefst 65536 (= 2^{16} , "16 bits") meetniveaus en dat geeft een prima geluidskwaliteit.
Evenals over de samplingfrequentie is er ook veel wetenswaardigs te vermelden over deze "verticale" nauwkeurigheid van het sampling proces, maar toch zullen we het hier bij laten. In figuur 19 vatten we samen wat



Figuur 19: samenvatting sampling.

op dit moment voor jou van belang is om aan de slag te kunnen met geluidsbewerkingen.

25. Opdracht: geluidsbestand nader bekeken
Ga met de muis naar menufunctie **Bestand** en klik op **Openen .WAV**.
Open het bestand ►Brekend glas.wav in de map **Geluiden**.
In Memo komt het volgende te staan (de pad-naam achter `Geopend: wav-bestand` zal er bij jou waarschijnlijk iets anders uitzien!):

```
Geopend: wav-bestand 'C:\Sound
Design\WAV\Weerbericht.wav'
  Kanalen:        2
  Fs:             44100
  Geladen:        45845 Ts (16-bits op sporen S1 en S2).
  Duur:           1,040 sec.
```

Deze gegevens maken deel uit van het bestand dat je zojuist hebt geopend. Het is de zogeheten *header* die informatie verstrekt over de samples die er op volgen.

- Het aantal kanalen is 2; dat betekent dat het geluid stereo is en dat er zowel op spoor S1 als op spoor S2 geluidssamples zijn geplaatst.
- De samplingfrequentie F_s is 44100 Hz: er werden van elke seconde geluid 44100 samples genomen. Je kunt uitrekenen dat 1 T_s = 1 / F_s sec = 1 / 44100 sec.
- Geladen: er zijn op elk spoor 45845 getallen - geluidssamples - geplaatst.
- Duur: 45845 T_s = *1,040* sec.

a. Speel het geluid af via het Scoop-Spectrumvenster. Schat zo goed mogelijk hoe lang het geluid duurt.
b. Heb je door het geluid af te spelen gebruik gemaakt van de AD-converter? En van de DA-converter?
c. Stel eens dat een hacker in de header van dit bestand het getal van de samplingfrequentie verandert van 44100 Hz in 22050 Hz. Dan zal WaveWizard, niets vermoedend, dat getal overnemen en intern F_s afstellen op 22050 Hz en dus per seconde niet 44100, maar slechts 22050 samples naar de koptelefoon en de Line-uitgang sturen. Hoe denk je dat het brekend glas nu zal klinken?
d. Ga met de muis naar menu **Diversen→Systeem**. Het bovenste invoervakje is de samplingfrequentie. Stel die in op 22050 Hz. Speel het geluid af en verifieer daarmee je antwoord op (c). Kies ook nog lagere frequenties voor F_s en beschrijf het klankeffect.

Vergeet niet om daarna F_s weer terug te zetten op 44100 !!

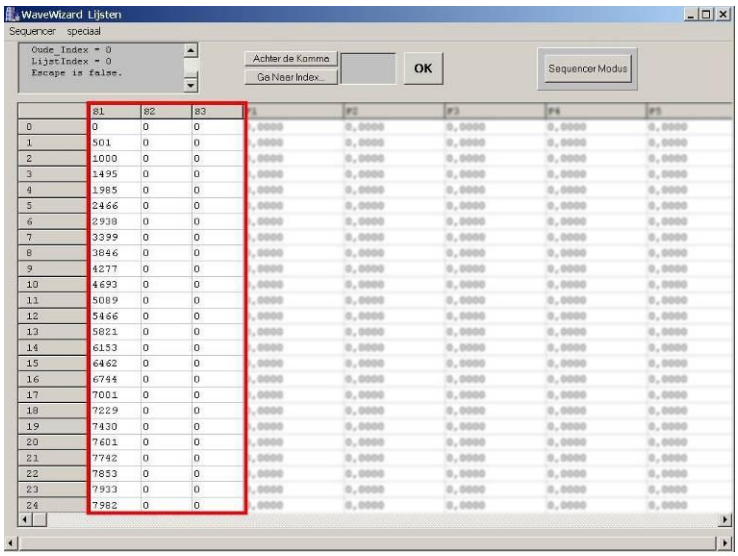
We weten nu dat geluid kan worden gerepresenteerd door een reeks getallen. Voor elke seconde geluid op een van de sporen van WaveWizard

bevat het RAM-geheugen om precies te zijn 44100 samples. Tot nu toe hebben we die getallen grafisch weergegeven in de $u(t)$-diagrammen van de sporen S1, S2 en S3. Nu we geluid gaan bewerken, moeten we de samples ook daadwerkelijk als "harde cijfers" kunnen weergeven, want van nu af aan is geluidsbewerking niets anders dan getalbewerking. Hoe dat gaat, zie je in opdracht 26.

26. Opdracht: de samples van een sinustoon bekijken en handmatig veranderen

a. Klik op **Wissen Buffers** en maak met toongenerator een sinustoon van 440 Hz, amplitude 8000, 20 sec. Klik net zo lang op **Zoom In** totdat in het veldje ernaast 32 pix / Ts verschijnt. Je ziet nu de afzonderlijke samples van de sinustoon.

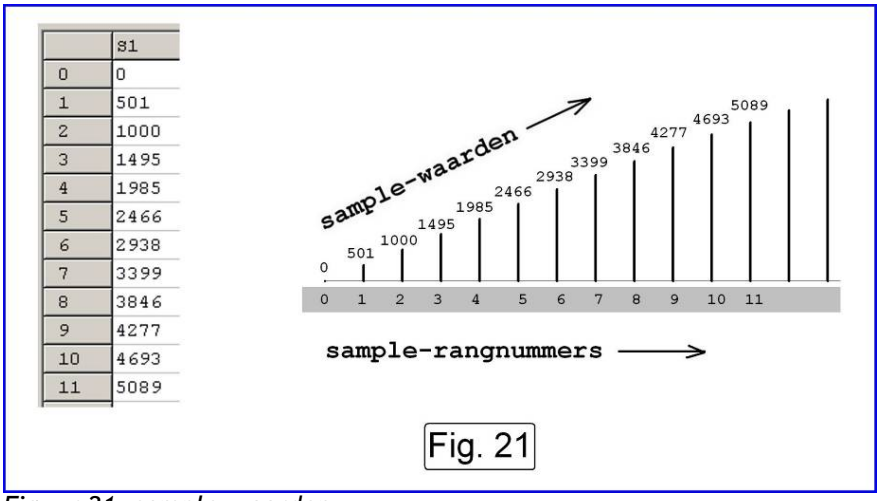
b. Ga naar menu **Diversen** → **Lijst** (je kunt ook op de **L**-toets drukken, dat gaat meestal sneller!) Je krijgt nu een lijst met getallen te zien. Zie figuur 20.
In de meest linkse (grijze) kolom vind je het nummer van de sample. Dat noemen we het ***sample-rangnummer (index)***. De eerste drie witte kolommen geven de ***samplewaarden*** (uitwijkingen) van resp. S1, S2 en S3 aan, zoals in de bovenste rij staat aangegeven. In de kolom van S1 vind je de samplewaarden van de sinustoon van 440 Hz. We spreken af dat we de samplewaarden voortaan als volgt in de tekst weergeven (vergelijk ook figuur 21):
S1[0] = 0
S1[1] = 501
...
S1[9] = 4277



	S1	S2	S3
0	0	0	0
1	501	0	0
2	1000	0	0
3	1495	0	0
4	1985	0	0
5	2466	0	0
6	2938	0	0
7	3399	0	0
8	3846	0	0
9	4277	0	0
10	4693	0	0
11	5089	0	0
12	5466	0	0
13	5821	0	0
14	6153	0	0
15	6462	0	0
16	6744	0	0
17	7001	0	0
18	7229	0	0
19	7430	0	0
20	7601	0	0
21	7742	0	0
22	7853	0	0
23	7933	0	0
24	7982	0	0

Fig. 20

Figuur 20: lijst met getallen.



Figuur 21: sample-waarden.

c. Ga met de cursor naar het veld S1[9] waarin nu 4277 staat. Maak er 32000 van en druk op toets **Esc** om het Lijst-venster te verlaten. Je bent nu weer terug op het hoofdscherm. Je ziet nu aan het begin van de golfvorm een scherpe piek. Als je op **Play** drukt dan hoor je aan het begin van de toon dan ook een tikje, als je goed luistert!
d. Hoeveel tijd (in seconde) verstrijkt er tussen sample S1[0] en sample S1[1000]?
e. Hoeveel tijd (in seconde) verstrijkt er tussen sample S1[0] en sample 2S1[1000]?
f. Hoeveel tijd (in seconde) verstrijkt er tussen sample S1[0] en sample S2[1000]?

Notatie
Let op de volgende afspraken over de notatie:

- je noteert het rangnummer altijd tussen vierkante haken [ ], bijv. S1[9]
- een rangnummer is nooit een negatief getal! Dus S1[−1] bestaat niet.

3 *S*-buffers en 32 *F*-buffers

De getalkolommen die je in het Lijst-Venster zag staan zijn de ***buffers*** van WaveWizard. Een buffer is in het algemeen een stuk geheugen dat gereserveerd wordt voor één en hetzelfde soort gegevens. De geheugenregisters van een tekstbuffer bijvoorbeeld bevatten getallen die leestekens symboliseren. De eerste drie buffers, S1, S2 en S3 zijn audiobuffers en fungeren als de geluidssporen die je op het hoofdscherm en in het Scoop-Spectrumvenster ziet. In S1, S2 en S3 kun je uitsluitend gehele getallen opslaan in het bereik van in het bereik − 32768 , ... , + 32767. Het zijn immers audiobuffers.
Verderop moeten we ook getallen met een komma erin kunnen maken. Daar heb je de zogeheten ***F-buffers*** voor nodig die ook in Lijst te vinden zijn, rechts van S1, S2 en S3. Je kunt er decimale getallen in opslaan met maximaal 14 cijfers achter de komma (hoeveel decimalen in het Lijst-

venster daadwerkelijk zichtbaar zijn, kun je instellen). Vanaf hoofdstuk 5 gaan we vaak gebruik maken van de F-buffers. In het Lijsten-venster zie je de F-buffers broederlijk naast de S-buffers staan, maar realiseer je dat ze niets met elkaar te maken hebben: alle buffers zijn volkomen onafhankelijk van elkaar! Een ander groot verschil tussen de S- en de F-buffers is hun omvang: de geluidsbuffers S1, S2 en S2 tellen elk vele miljoenen getallen (je kunt die bovendien instellen (menu **Diversen** → **Systeem**). De F-buffers bevatten elk "slechts" 70.000 decimale getallen.

3.3 Werken met het commandovenster

Je hebt in hoofdstuk 2 al te maken gehad met het commandovenster, maar dat gaan we vanaf nu veel intensiever gebruiken. De hoofdregel luidt:

Als je op **Start** klikt, worden alle instructies die in het commandovenster staan uitgevoerd, zonder onderbreking, één voor één, in volgorde van boven naar beneden.

Je kunt op drie manieren tekst plaatsen in het commandovenster, namelijk:

1. **Open Preset** (dit deed je inmiddels al in een aantal opdrachten).
2. Door te klikken op **menufunctie's**; je komt dan op het invoervenster (figuur 12) en zet een vinkje in **Commando's weergeven** in commando-venster en verwijder het vinkje uit **Commando direct uitvoeren**. Nu wordt het commando uitgeprint in het commando-venster en gebeurt er verder niets (de computer slaat pas aan het rekenen nadat je op **Start** hebt geklikt).
3. Door rechtstreeks instructies in te typen. Doe dit alleen als het gaat om slechts een enkel getal of formule of een van de eenregelige instructies die we hieronder zullen bespreken. Ga nooit hele codeblokken (zoals bijv. van Toongenerator) direct intypen! De kans is groot dat je fouten maakt en dan stopt WaveWizard en geeft foutmeldingen. Maak zoveel mogelijk gebruik van presets en menufuncties!

De "=" instructie

Je kunt de waarde van een sample direct veranderen via de "=" instructie . Typ in het commando-venster onder de instructies die er al staan:

```
S1[9] = S1[9] + 11
Print S1[9]
```

Klik op **Wissen→Buffers** en klik op **Start**. In Memo verschijnt de volgende tekst:

```
Uitvoer commando's gestart...
```

```
  S1[9] = 0
  Toongenerator
  S1[9] = 4277
      S1[9] = 4288

     Commando's uitgevoerd. OK
```

Let op de wat vreemde betekenis van het "=" teken in de instructie `S1[9] = S1[9] + 11`. Wiskundig en logisch gezien staat daar onzin. Maar in programmeertalen betekent "=" dat de uitkomst van de formule rechts van "=" wordt opgeslagen in het bufferregister dat links van "=" staat.

Foutmeldingen

Het zal ongetwijfeld wel eens voorkomen dat er in de code die je in het commando-venster schrijft instructies staan die fouten opleveren. Die zijn niet altijd direct zichtbaar. Als je intypt:

```
Print 1/0
```

dan zal het je niet verbazen dat je in Memo de melding krijgt:

```
Deling door nul
Functie Print meldt (run-)fout in expressie: 1/0
Programma-executie afgebroken: Fout geconstateerd.
```

Maar als je intypt:

```
Print 1/S2[237]
```

dan kun je precies dezelfde melding krijgen, want S2[237] kan natuurlijk ook nul zijn.

Een ander voorbeeld:. Er staat in het commando-venster:

```
n = -10
Print S1[n]
```

Als je op **Start** klikt, krijg je in Memo de melding:

```
Afgebroken: Fout in expressie buffer. ( F??[??] , S??[??]  )
  buffernummer = 1
  index = -10
Functie Print meldt (run-)fout in expressie: S1[-1]
Programma-executie afgebroken: Fout geconstateerd.
```

Nog een laatste voorbeeld. Als je intypt:

```
S1[0] = 1234567
```

dan verschijnt in Memo de melding:

```
De waarde 1234567 ligt buiten bereik [-32768, +32767] van een S-
buffer in sample-rangnummer 0
Programma-executie afgebroken: Fout geconstateerd.
```

Bij de eerste de beste fout houdt WaveWizard het voor gezien en breekt de programma-executie af.

Als je op **Start** hebt geklikt, en WaveWizard begint met de uitvoer van de instructies, houd dan een oogje op Memo; WaveWizard probeert je zo goed mogelijk te informeren als er iets fout gaat.

De Wis-instructie

Soms vergeet je wel eens een buffer eerst te wissen voordat je op **Start** klikt. Dan zal bijvoorbeeld bij de sinustoon die er al staat een identieke worden gemixt - we zullen dat verderop nader bekijken. Daarom is het handig als je de buffers ook kunt uitwissen d.m.v. een instructie in het commando-venster. Je moet uiteraard die instructie dan wel bovenaan zetten! Zet je hem onderaan, of ergens in het midden, dan zal WaveWizard eerst braaf aan het rekenen slaan en uiteindelijk alle resultaten direct weer uitwissen...
Wil je alleen de drie sporen S1, S2 en S3 wissen, ga dan naar menufunctie **Bewerking → Wis geselecteerde Buffer(s)**. Vul daar in welke buffer je wilt wissen, zorg dat er een vinkje staat bij **Commando's weergeven** in commando-venster en verwijder het vinkje uit **Commando direct uitvoeren**. Klik op **OK**. Nu wordt het onderstaande wis-commando geprint in het commando-venster:

```
Wis buffers
  Wis buffer (F1...F32, S1, S2, S3)    S1  |  S2  |  S3
```

Het uitroepteken

De wis-instructie zul je heel vaak gebruiken, want je hoeft dan nooit meer te klikken op **Wissen→Buffers**. Daarom staat de wis-instructie bij opstarten van WaveWizard al voorgedrukt in het commando-venster, maar wel met een uitroepteken ervoor.

WaveWizard is blind voor alles wat rechts van **!** staat.

Zo kun je instructies snel eventjes buiten werking zetten door er een **!** voor te plaatsen. Je kunt dat doen door alleen de regel van de instructie-naam te voorzien van een uitroepteken; je hoeft niet alle regels daaronder, waarin de invoervelden staan, van een uitroepteken te voorzien. Bijvoorbeeld: de toongenerator hieronder wordt niet uitgevoerd, als er staat:

```
!Toongenerator
  Frequentie (Hz of TOETS)   440 Hz
  Amplitude                  8000
  Golfvorm                   sinus
  Spoor: S?[?]               S1[0]
  Duur                       20 sec
      Aantal harmonischen        1
```

Je kunt ook je eigen commentaar schrijven achter de instructies die je invoert, want zodra WaveWizard in een regel een uitroepteken tegenkomt, verliest hij elke interesse en gaat direct verder met de volgende regel.

Declareren

Vaak zul je het handig vinden om zelf namen voor variabelen te bedenken en te gebruiken. Bijvoorbeeld in opdracht 39 komt in een formule de lengte van een geluid, uitgedrukt in T_s, voor, namelijk 552193. Het kan zijn dat zo'n groot en nietszeggend getal steeds weer in formules opduikt, misschien naast nog andere nietszeggende getallen. Dat is niet alleen vervelend om in te typen maar ook om te lezen. Welke van onderstaande codes vind jij begrijpelijker, (a) of (b)? (De instructie **`Bewerk signaal`** leggen we uit in hoofdstuk 4.)

```
(a)
Bewerk signaal
  n0          44100
  n1          552193 + 44100  - 1
  Bewerking   S2[n] = S2[n] + S1[n-44100]

(b)
Bewerk signaal
  n0          Echotijd
  n1          Lengte + Echotijd - 1
  Bewerking   S2[n] = S2[n] + S1[n-Echotijd]
```

Je kunt zelf namen voor variabele of constante getallen verzinnen zoals **`Lengte`** en **`Echotijd`**. Je moet ze dan wel eerst "aanmelden" bij WaveWizard en ook vertellen welke waarde die getallen hebben. Dat aanmelden noemen we ***declareren***.Het WaveWizard-commando ervoor luidt: **`Declareer`**. Achter dat woord kun je een of meerdere namen noteren, gescheiden door een puntkomma:

```
Declareer Lengte; Echotijd
```

Een belangrijke regel voor declaraties is:

Het commando **`Declareer`** mag uitsluitend bovenaan in de commando-tekst staan, dus vóór alle andere instructies.
Geef je eerst een ander commando en daarna **`Declareer`**, dan krijg je een foutmelding.

Ook moet je erop letten dat je die variabelen een waarde geeft vóórdat je ze in een formule gebruikt.
Doe je dat niet dan gaat WaveWizard ervan uit dat ze nul zijn!!
Bijvoorbeeld:

```
Lengte = 552193
Echotijd = 1 * Fs
```

Zakrekenmachine

Je kunt de printinstructie ook gebruiken als zakrekenmachine. Voer in:

```
Print sin(pi/3)
Print e
Print 2 - log(100)
```

Je krijgt dan in Memo:

```
Uitvoer commando's gestart...

  sin(pi/3) = 0,866025403784439
  e = 2,71828182845905
  2 - log(100) = 0

Commando's uitgevoerd. OK
```

- Let erop dat in computerprogrammeertalen het vermenigvuldigingsteken een sterretje is: `*`.
- Machtsverheffen geef je aan met een hoedje, bijvoorbeeld 2^3 schrijf je als `2^3`.
- Je mag spaties zetten tussen de afzonderlijke elementen van een formule. Je kunt dus invoeren: `60*n+1`, maar ook: `60 * n + 1`. Dat leest vaak wat prettiger.
- Je kunt in WaveWizard als decimaalteken een komma zetten (zoals gebruikelijk in ons land) maar ook een punt (zoals op je rekenmachine).
- In formules kun je ook `pi`, `e`, `Fs` en `Ts` noteren. De eerste twee zijn natuurlijk de bekende constanten uit de wiskunde. F_s is ingesteld, zoals je weet, op 44100 Hz, (maar die waarde kun je veranderen; zie opdracht 25d) en $T_s = 1/F_s$ sec. Al deze getallen zijn "double precision": 14 cijfers achter de komma.
- Alle bewerkingen worden trouwens altijd uitgevoerd in "double precision", als de eindbestemming een samplebuffer S1, S2 of S3 is, wordt uitsluitend het eindresultaat afgerond tot een geheel getal.
- Als absolute waarde van een getal a noteer je niet $|a|$, zoals in de wiskunde, maar abs(a).

- In de huidige versie van WaveWizard kun je de volgende wiskundige functies gebruiken: log(n), ln(n), a^x , sin(n), cos(n), tan(n) en ook een paar functies die je wellicht niet kent (en niet hoeft te gebruiken in deze module) zoals: asin(n), acos(n), atan(n).

27. Vraag

Waarom geeft de instructie `S1[0] = sin(pi/3)` de foute uitkomst 1 en niet, zoals hierboven, 0,866025403784439? Maak er eens van: `F1[0] = sin(pi/3)` en vergelijk het resultaat.

Presets opslaan en openen

De tekst in het commando-venster is dus een lijst met instructies voor WaveWizard. Die tekst kun je opslaan en naderhand weer openen. Dit doe je met resp. **Bestand→Preset Opslaan** als en **Bestand→Open Preset** (dat laatste heb je inmiddels al vaak gedaan).
Maak er een goede gewoonte van om je presets eerst op te slaan en dan pas uit te voeren via de **Start**-knop. Dat is altijd, bij alles wat je met de computer doet, verstandig! Je kunt niet vaak genoeg opslaan!

Samenvatting

Sampling

1. Omdat een computer (= rekenmachine!) alleen maar getallen kan verwerken, moet geluid eerst in getalvorm worden gerepresenteerd. Dit gebeurt door op regelmatige tijdstippen de uitwijking te registreren (sampling). Een meetgetal heet een ***sample***.
2. De ***samplingfrequentie*** F_s is het aantal samples dat per seconde wordt genomen. F_s moet minstens twee maal zo hoog zijn als de hoogste frequentie die in een signaal voorkomt. (***stelling van Nyquist***).
3. De ***sample-tijd*** T_s is de tijd die verstrijkt tussen twee samples. $T_s = 1/F_s$.
4. Samples zijn gehele getallen in het bereik van −32768 tot +32767.
5. Samples worden in de computer opgeslagen in gereserveerde geheugensegmenten: ***buffers***. WaveWizard heeft drie audiobuffers ("sporen"): S1, S2 en S3. Daarnaast zijn er ook de zg. ***F-buffers***, waarin je decimale getallen kunt opslaan en bewerken.
6. Het rangnummer (index) van een sample wordt aangegeven tussen haken [ ] achter de buffernaam, bijv. S1[100] of F7[23]. Rangnummers kunnen niet negatief zijn.

Commandovenster (WaveWizard)

1. Elke menufunctie die je aanroept produceert in het commando-venster een tekst waarin de invoer wordt beschreven.
2. In het commando-venster kunnen ook instructies staan die niet van een menufunctie afkomstig zijn, zoals: **`Print`**, **`Declareer`** en **`=`**.

3. Als je op de **Start**-knop klikt, worden alle instructies in het commando-venster in volgorde van boven naar beneden, uitgevoerd.
4. Declaraties mogen uitsluitend aan het begin van de commando-tekst staan.
5. Het is meestal verstandig om de tekst in het commandovenster te laten beginnen (althans na de declaraties!) met de wis-instructie die de sporen S1, S2 en S3 nul maakt. Uitzondering: het openen van een wav-bestand.
6. De tekst in het commandovenster kun je opslaan als Preset en in een later stadium weer openen en uitvoeren.

Woordenlijst

AD-converter	§2	sample	§2
buffer	§2	sample-rangnummer	§2
continu	§2	sample-tijd T_s	§2
DA-converter	§2	sample-waarde	§2
`Declareer`	§3	sampling	§2
declareren	§3	samplingfrequentie F_s	§2
discreet	§2	signaalverwerking	§2
F-buffer	§2	stelling van Nyquist	§2
`Print`	§3	**`Wis buffers`**	§3

4 Versterken, mixen en echo's toevoegen

4.1 Versterken

Constante versterking

Nu komen we eindelijk toe aan een beetje signaalverwerking! Het eerste wat we graag willen doen is een geluid harder of zachter maken. In hoofdstuk 2 heb je al ontdekt dat een geluid zachter wordt naarmate de amplitude kleiner is. Wil je een geluid zachter maken, dan moet je alle samples vermenigvuldigen met een getal < 1, en om een geluid harder te maken met een getal > 1.
Laten we dat eens uitproberen. Stel je voor dat er in het commandovenster de volgende code staat (je hoeft alleen maar goed te lezen en nog even niets met WaveWizard te doen):

```
Toongenerator
  Frequentie (Hz of TOETS)   440 Hz
  Amplitude                  8000
  Golfvorm                   sinus
  Spoor: S?[?]               S1[0]
  Duur                       20 sec
  Aantal harmonischen        1
```

Dit is dus een sinustoon met amplitude 8000 die op spoor S1 staat.
Als we de toon 2 keer zo hard willen horen, moeten we alle samples van de sinustoon vermenigvuldigen met 2. Als we de versterkte toon in S2 zetten, dan moeten we de volgende berekeningen uitvoeren:

```
S2[0] = 2 * S1[0] →  2 * 0    = 0
S2[1] = 2 * S1[1] →  2 * 501  = 1002
S2[2] = 2 * S1[2] →  2 * 1000 = 2000
S2[3] = 2 * S1[3] →  2 * 1495 = 2990
 ...
```

Wat links van de pijl staat zou je in principe direct in het commandovenster kunnen intypen, maar je begrijpt dat dit onbegonnen werk is, want je zou voor elke seconde geluid 44100 regels tekst krijgen (leg uit waarom).
Maar ook zie je dat de berekeningen een structuur hebben, namelijk:

```
S2[n] = 2 * S1[n], voor n = 0, 1, 2, 3 ...... 20 * Fs.
```

Voor deze berekening kun je gebruik maken van de volgende functie (zie ook menu **Generatoren → Bewerk signaal**):

```
Bewerk signaal
  n0          0
  n1          20 * Fs
  Bewerking   S2[n] = 2 * S1[n]
```

Wat WaveWizard doet bij deze instructie is exact wat we hierboven hebben uitgeschreven voor n = 0 tot en met n = 3, maar dan voortgezet tot en met n = 20 F_s = 882000. In functies zoals deze wordt gewerkt met een intern tel-register n. In dit voorbeeld wordt n ingesteld op wat je invult achter `n0`. Daarmee wordt de bewerking die je opgeeft doorgerekend en de uitkomst opgeslagen. Vervolgens wordt n met *1* verhoogd en wordt de bewerking opnieuw doorgerekend en opgeslagen. Zo gaat het proces voort tot en met de waarde die je hebt opgegeven bij `n1`. Dit cyclische proces noemen we meestal een ***lus***.

Heel veel functies in WaveWizard bestaan uit dergelijke lussen, zoals bijvoorbeeld de Toongenerator of Sinus Stijgend. Ook de ruisklanken worden zo gemaakt; daarbij gooit WaveWizard steeds weer met een dobbelsteen (met een instelbaar aantal vlakken) en noteert te uitkomsten in een S-buffer.

28. Opdracht: sinustoon versterken

a. Open ►Preset_H4_Sinustoon_Versterken.txt. en klik op **Start**.
b. Vergelijk in de Lijstpagina de kolommen van S1 en S2.
c. Verander de code zo dat de toon op S2 drie keer zo zacht is als die op S1. Vergelijk de twee kolommen in de Lijstpagina opnieuw.

De notatie van de tijdsduur

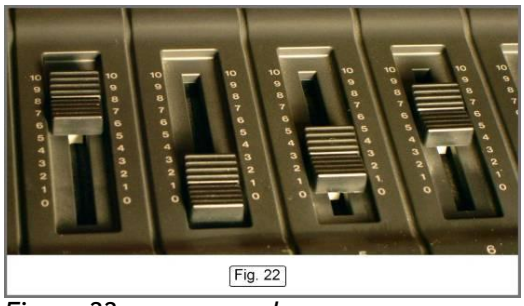
De code voor de versterking van het geluid in opdracht 28 luidde:

```
Bewerk signaal
  n0          0
  n1          20 * Fs
  Bewerking   S2[n] = 2 * S1[n]
```

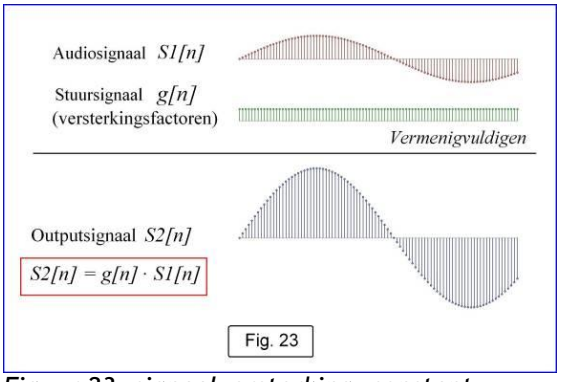
Het aantal samples dat WaveWizard hierbij uitrekent is 20 * Fs = 882000 samples (882001, om precies te zijn, leg uit!). Maak er een goede gewoonte van om tijdsduren uit te drukken in Fs , want dan zie je in de commandotekst veel gemakkelijker hoe lang het geluid in seconde duurt.
Immers 1 seconde bevat Fs samples.

"Infaden, outfaden"

Bij infaden en uitfaden maak je een geluid geleidelijk harder of zachter, zoals we in hoofdstuk 1 al hebben vermeld. Dat harder of zachter maken kun je doen door schuifpotmeters op een mengpaneel naar voren of naar achteren te schuiven, zoals te zien in figuur 22. Die op- en neergaande bewegingen die je daarbij met je hand maakt zijn fluctuaties en, zoals we in de inleiding hebben afgesproken, fluctuaties zijn signalen. Met je hand "bestuur" je het volumeverloop; daarom noemen we die handbewegingen een ***stuursignaal***. Deze term kom je met name veel tegen in de meet- en regeltechniek, waarbij elektronische apparaten andere apparaten aansturen. Ook synthesizers zijn daar voorbeelden van.



Figuur 22: mengpaneel.



Figuur 23: signaalversterking, constant stuursignaal.

We hebben zojuist een signaal versterkt door alle samples te vermenigvuldigen met hetzelfde getal.
Deze bewerking kun je je ook voorstellen op de manier van figuur 23. Hier is de constante 2 vervangen door het constante stuursignaal *g[n]* = 2 voor elke waarde van *n*. De grafiek van *g[n]* bestaat uit een horizontale, rechte lijn. In de figuur is van *g[n]* een aantal samples afgebeeld, die allemaal dezelfde waarde hebben, dus $g[0]=g[1]=g[2]=\ldots=2$.

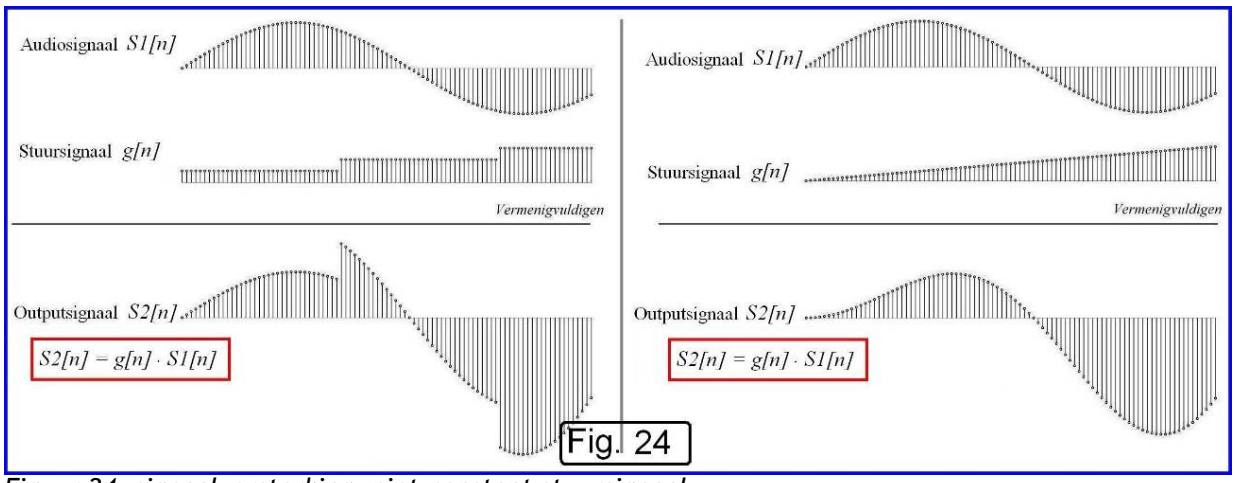
Stel, dat we het geluid op spoor S1 willen vermenigvuldigen met stuursignaal *g[n]* en het bewerkte signaal willen opslaan op spoor S2, dan kunnen we dat kort opschrijven als:

$$S2[n]=S1[n]\cdot g[n] \qquad (3)$$

Dit zou je kunnen noemen: ***"sample-gewijs vermenigvuldigen"***:

$$S2[0]=S1[0]\cdot g[0]$$
$$S2[1]=S1[1]\cdot g[1]$$
$$S2[1]=S1[2]\cdot g[2]$$

etc.



Figuur 24: signaalversterking, niet-constant stuursignaal.

Uiteraard krijgt deze manier van voorstellen pas echt zin als het stuursignaal g[n] een niet-constante functie is, zoals te zien in figuur 24.

Links in de figuur is het stuursignaal een "trapfunctie", die steeds gedurende een aantal samples constant is en dan plotseling wordt verhoogd. Het resultaat van de bewerking is dat het outputsignaal schoksgewijs sterker wordt, alsof iemand steeds de volumeknop met een ruk een stuk verder opendraait.

Rechts in de figuur is het stuursignaal een lineaire functie. Hier neemt het volume van het outputsignaal heel geleidelijk toe, alsof iemand de volumeknop van de versterker heel langzaam verder opendraait.
In de opnamestudio noemen ze dat ***infaden*** . Het omgekeerde daarvan, heel geleidelijk zachter zetten, heet ***outfaden***.

Formule van een lineaire functie opstellen

We gaan nu bekijken hoe je met de computer een geluid heel geleidelijk harder maakt. Ook hier hoef je nog niets met WaveWizard te doen en alleen maar goed te lezen. Het gaat om een techniek die we ook in latere hoofdstukken voortdurend weer tegenkomen, namelijk het opstellen van de formule van een lineaire functie als twee functiewaarden van die functie bekend zijn (bijvoorbeeld in de vorm van twee geluidssamples).

We nemen als testsignaal: een sinustoon van 500 Hz, amplitude 8000 die 10 sec duurt en zetten die op spoor S1. We willen de toon in 10 sec lineair in sterkte laten toenemen. Met dat lineair-zijn willen we het effect nabootsen dat je krijgt als je op een mengpaneel een schuifpotmeter met constante snelheid naar voren beweegt. Daarbij spreken we af dat de potmeter bij de nulstand een versterkingsfactor 0 heeft en bij de maximale stand een versterkingsfactor 1.
Het resultaat zetten we op spoor S2. We maken eerst de sinus:

```
Toongenerator
  Frequentie (Hz of TOETS)   500 Hz
  Amplitude                  8000
  Golfvorm                   sinus
  Spoor: S?[?]               S1[0]
  Duur                       10 sec
  Aantal harmonischen        1
```

De sinustoon op spoor S1 heeft een amplitude 8000. De toon op spoor S2 zal in amplitude langzaam moeten gaan toenemen van 0 tot 8000. Op tijdstip nul moet de versterking ook nul zijn, en na tien seconde is de versterking 1, zodat je amplitude 8000 krijgt.
Nu moeten we een stuursignaal *g[n]* aanmaken dat 10 sec duurt en dat een lineaire functie moet zijn, dus van de algemene vorm:

$$g[n] = an + b \qquad (4)$$

Hierin is *a* het hellingsgetal en *b* is de waarde bij *n* = 0. We moeten nu *a* en *b* afleiden uit de gegevens waar we over beschikken. Voor die twee onbekenden hebben we minimaal twee vergelijkingen nodig. Dat kan alleen als we twee functiewaarden van *g[n]* al kennen. En dat is inderdaad het geval, want we kennen de eerste en de laatste versterkingsfactor; die zijn 0 en 1.
De eerste versterkingsfactor, *g[0]*, komt op tijdstip 0. Daarom kunnen we schrijven:

$g[0] = a \cdot 0 + b = 0$ dus *b* = 0.

Die laatste versterkingsfactor is bedoeld voor de sample op het tijdstip *t* = 10 sec. Welk rangnummer heeft die sample? Er gaan F_s = 44100 samples in 1 seconde, dus na 10 seconde zitten we bij samplenummer 441000 = 10 F_s. Dat is dus de versterkingsfactor $g[10F_s]$ en we weten al dat die factor gelijk is aan 1, dus $g[10F_s]$ = 1. Nu kunnen we *a* uitrekenen:

$$g[10F_s] = a \cdot (10F_s) = 1$$

Dus $a = 1/(10F_s) = 0{,}1T_s$

en het functievoorschrift voor *g[n]* wordt daarmee:

$$g[n] = 0{,}1T_s n$$

Nu vermenigvuldigen we *S1* met *g* en zetten het resultaat op spoor *S2*:

```
Bewerk signaal
  n0          0
  n1          10 * Fs
  Bewerking   S2[n] = (0,1*Ts*n) * S1[n]
```

Uiteraard mag je die (ronde) haakjes weglaten; die staan er alleen maar om je nog even te laten zien welk gedeelte van de formule *g[n]* is.

29. Opdracht: vind een lineaire functie als twee samples gegeven zijn

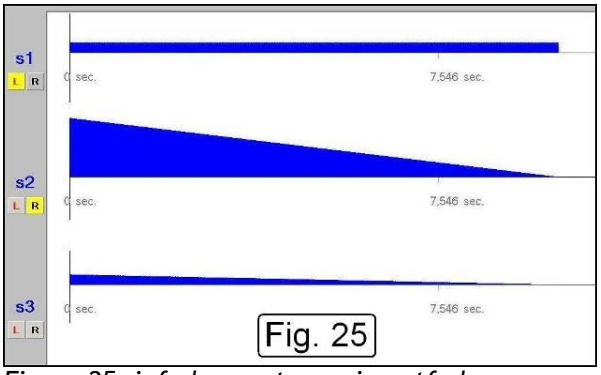
Van een lineaire functie $f(x) = ax + b$ zijn twee functiewaarden gegeven:

1. $f(x_0)$ en $f(x_1)$
2. de waarden x_0 en x_1; ze zijn beide ongelijk aan nul.

a. Druk *a* en *b* uit in x_0, x_1, $f(x_0)$ en $f(x_1)$.
b. Beantwoord dezelfde vraag voor het geval dat $x_0 = 0$.

30. Opdracht: infaden, outfaden

a. Open ►Preset_H4_Infaden.txt. Dit is de code waarmee je een geluid kunt infaden, zoals hierboven werd besproken. In de drie regels die eindigen met een boel vraagtekens moet je eerst de puntjes vervangen door de juiste code. Die vind je hierboven.
b. Wat moet je veranderen om ervoor te zorgen dat infaden overgaat in outfaden zoals te zien in figuur 25?
c. Welke veranderingen moet je aanbrengen om een sinustoon te maken van 2,5 sec, waarvan de amplitude in diezelfde tijd lineair afneemt tot nul?



Figuur 25: infaden gaat over in outfaden.

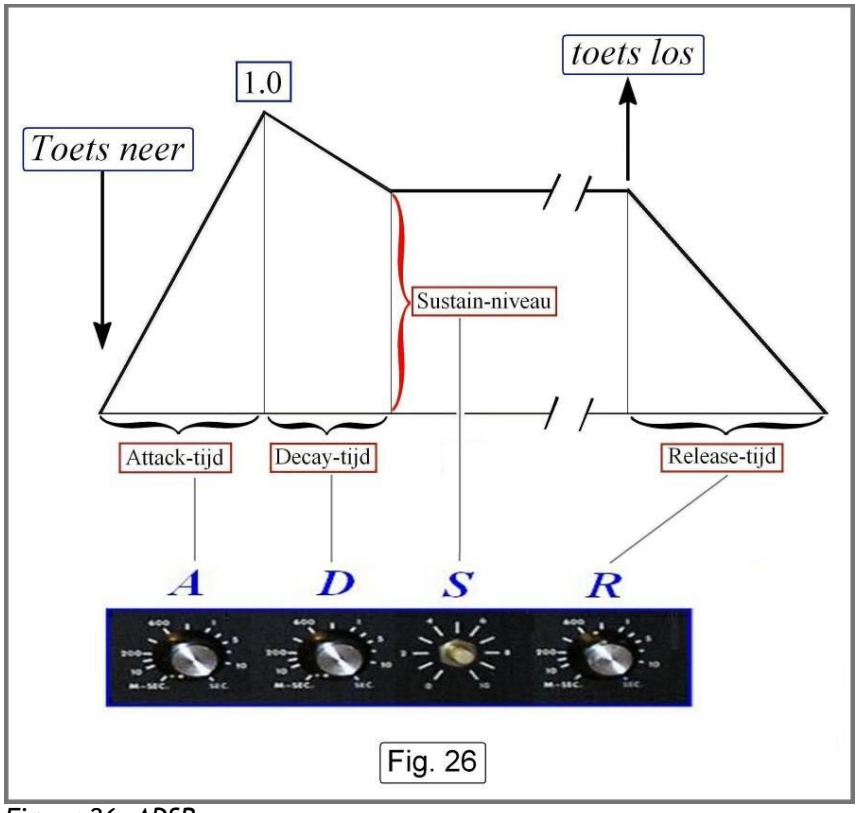
Toepassing infaden of outfaden: contourgenerator

Een van de vele redenen waarom de tonen die je met de toongenerator maakt zo onnatuurlijk klinken is dat het volumeverloop constant is. De toon komt abrupt op vol volume, blijft tot het einde op datzelfde volume voortklinken en eindigt ook weer heel plotseling. Bij echte muziekinstrumenten komt de toon geleidelijker opzetten en dempt ook geleidelijk uit.
Daarom vind je in WaveWizard nog een uitgebreidere versie van de toongenerator, met instellingen waarmee de "contour" (het verloop) van

het volume wordt geregeld precies op de manier zoals je dat hierboven zelf deed bij dat in- en outfaden. Deze extra instellingen heten gezamenlijk de ***contourgenerator*** en die uitgebreidere versie van de toongenerator noemen we de ***Toongenerator ADSR***.

ADSR

Al op de vroegste synthesizers werden in het volumeverloop van de toon vier fases onderscheiden: ***Attack*** (aanzet), ***Decay*** (terugval), ***Sustain*** (aangehouden) en ***Release*** (loslaten van toets) afgekort: ***ADSR***. Deze kon je op de synthesizer via draaiknopjes instellen, zoals te zien in figuur 26.



Figuur 26: ADSR.

Attack

Je herkent een muziekinstrument voor een belangrijk deel aan de zogeheten ***attack*** of *aanzet* van de toon. Daarmee wordt bedoeld: de tijd die de toon nodig heeft om op volle sterkte te komen. Een toon op een dwarsfluit heeft een veel snellere attack dan een toon op een contrabas. De attack-tijden van muziekinstrumenten lopen uiteen van ongeveer 10 tot 50 milliseconde. Extreem lage orgelpijpen kunnen een attack-tijd van ruim 3 seconde hebben!

Decay

Bij veel instrumenten, bijvoorbeeld koperblaasinstrumenten zakt direct na de attack het geluid een beetje in voordat de fase van het constante volume begint. Dat inzakken wordt de ***decay*** (verval) genoemd. Juist door

de decay krijgt de attack een impulsief karakter, dat wil zeggen pijlsnel stijgen en dalen achter elkaar.

Sustain

Bij blaas- en strijkinstrumenten en het orgel wordt voortdurend energie toegevoerd, zodat de toon in principe oneindig lang kan blijven voortklinken. Wel kan het zijn dat de voortklinkende toon *zachter* is dan op het moment waarop de attack overgaat in de decay. Dit regel je met de sustain.

Sustain is geen tijdsduur, zoals attack, decay en release, maar een volumeniveau!

Op het moment dat de attack-fase overgaat in de decay-fase is het niveau altijd gelijk aan 1. Daarna kan het nooit harder worden! Dus het sustain-niveau is een getal s waarvoor geldt:
$0 \leq s \leq 1$.
Bij tokkelinstrumenten (gitaar, piano, harp) wordt eenmalig een zekere hoeveelheid trillingsenergie toegevoerd. Daarna kan de toon alleen nog maar langzaam zachter worden; de toon wordt niet aangehouden (dat is wat het woord sustain betekent): het sustain-niveau is nul! Het uitklinken regel je dus volledig met decay en release.

Release

Als je de toets van een piano loslaat (***release***), of je hand op een trillende (gitaar-)snaar legt, wordt de trilling als het ware gesmoord. Dat wil zeggen dat de "vrije" trilling van de snaar, waarvan je de dempingstijd instelt met decay, extra wordt verkort. Toch duurt het dan nog heel eventjes voordat je echt niets meer hoort. Daarom klinkt een toon zonder de release-fase onrealistisch. Ook bij blaasinstrumenten is er sprake van een release-fase, want de lucht die trilt in een buis houdt daar niet abrupt mee op als je stopt met blazen, maar de dempingstijd is wel veel korter dan die van snaren.

ADSR samengevat

A	**Attack-tijd** *(in msec)*	*tijd nodig om het maximale volume (= 1) te bereiken.*
D	**Decay-tijd***(in msec)*	*tijd nodig om van het max. volume naar sustain-volume te gaan.*
S	**Sustain-niveau** *[0, 1]*	*volume van aangehouden toongedeelte. Bereik: [0, 1].*
R	**Release-tijd** *(in msec)*	*tijd nodig om van sustain-niveau te dempen naar volume nul.*

Lineaire of exponentiële ADSR?

In opdracht 30 heb je zelf het volume van een geluid lineair laten toe- en afnemen (in- en outfaden). Bij muziekinstrumenten komt exponentiële toename of afname van het geluidsvolume veel vaker voor dan lineaire. Dus als je een toon op een "natuurlijke" manier wilt laten klinken, maak je

liever gebruik van exponentiële demping. Figuur 27 toont de flanken van de Decay-fase en de Release-fase als je de ADSR instelt op `exp`.

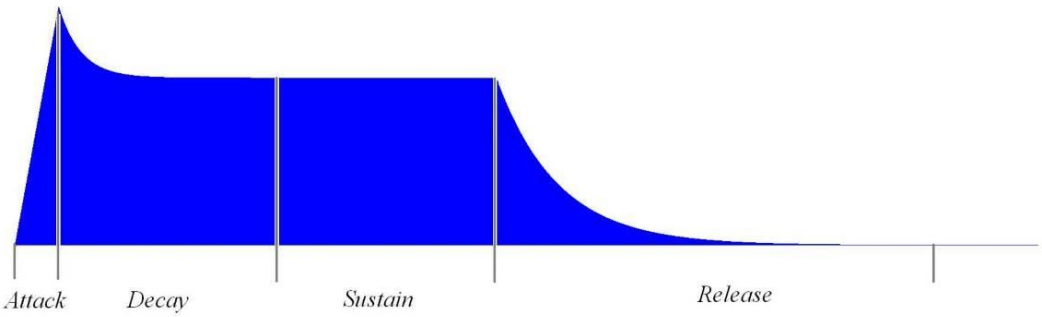


Fig. 27

Figuur 27: exponentiële demping.

31. Opdracht: drie contouren
Open ►Preset_H4_ADSR.txt. Deze code maakt op resp. S1, S2, S3 een zaagtandtoon aan. Probeer die tonen een contour te geven zoals in figuur 28.

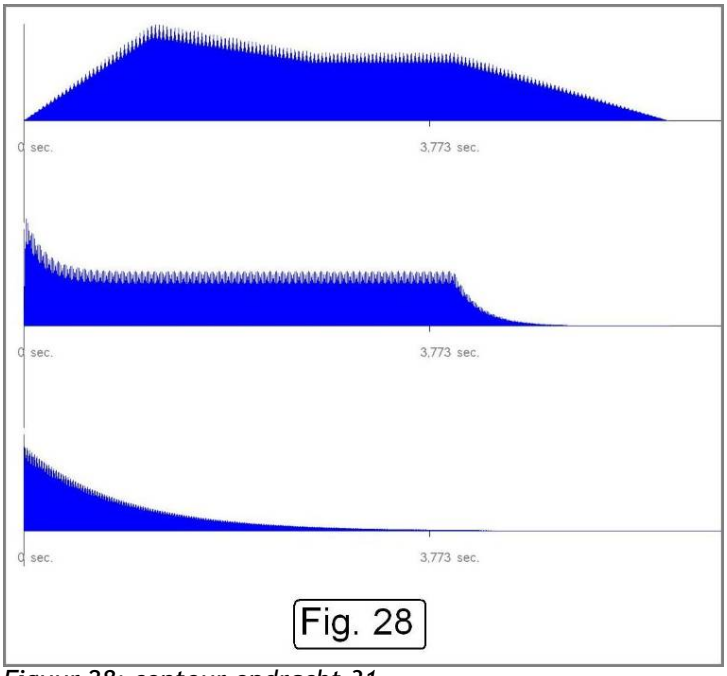


Fig. 28

Figuur 28: contour opdracht 31.

32. Opdracht: release
Het is eigenlijk een beetje moeilijk om een goede indruk te krijgen van het effect van de volume-contourgenerator door slechts naar één enkele toon te luisteren. Het gaat vooral om de totaalindruk.
Luister nog eens naar ►H1_Bach_motet_release_2000.mp3 en ►H1_Bach_motet_release_200.mp3. Breng het verschil in klank onder woorden.

4.2 Mixen

Als je op straat een brommer hoort en tegelijkertijd een blaffende hond, hoor je een mix van twee geluiden. ***Mixen*** is het samenvoegen van meerdere signalen tot één signaal. Wil je bijvoorbeeld de geluiden van spoor S1 en S2 mixen op spoor S3, dan tel je ze eenvoudigweg sample voor sample bij elkaar op. Dat gaat als volgt:

$$S3[0] = S1[0] + S2[0]$$
$$S3[1] = S1[1] + S2[1]$$
$$S3[1] = S1[2] + S2[2]$$
etc.

Net zoals we bij versterken spraken van "sample-gewijs vermenigvuldigen", kunnen we mixen aanduiden als ***"sample-gewijs optellen"***. Ook hier vinden we die cyclische structuur:

$$S3[n] = S1[n] + S2[n] \qquad (5)$$

Daarom kunnen we ook bij mixen gebruik maken van diezelfde lus-instructie **`Bewerk signaal`**.
Stel dat je de eerste 20 seconde van de sporen wilt mixen, dan krijg je de volgende code:

```
Bewerk signaal
  n0          0
  n1          20 * Fs
  Bewerking   S3[n] = S1[n] + S2[n]
```

33. Opdracht: raaf met brom

Een sound designer moet een geluid zo afleveren dat het afkomstig lijkt uit een goedkope, oude versterker waar een brom in zit. Die brom is een zogeheten ***blokgolf***-toon van 50 Hz (blokgolf: zie figuur 10C en opdracht 36).

a. Open ►Preset_H4_BlokRaaf.txt. Je ziet in de commando-tekst regels met vraagtekens erachter. In die regels moet je iets invullen voordat je op Start kunt drukken.
   Open ►Raaf.wav. Je hebt nu een stereogeluid op sporen S1 en S2. Uit hoeveel samples bestaat het geluid? (zie de tekst die in Memo verschijnt na openen.).
   Vul je antwoord in bij regel (a), waar je de **`duur`** van het geluid specificeert.
b. De toongenerator maakt de blokgolf aan. Geef bij (b) de frequentie op.

c. De bromtoon moet op spoor S2 komen en beginnen op sample 0. Noteer dat bij (c).
d. Wat is het maximum aantal harmonischen dat je hier kunt invullen? Noteer bij (d). Denk aan de stelling van Nyquist!
e. Vul bij (e) een formule in zodat op S3 een mix komt van S1 en S2.
f. Als je de brom in de mix wat overdreven hard vindt, kun je natuurlijk in de code van `Toongenerator` de amplitude ervan verlagen, maar je kunt ook de formule van vraag (e) veranderen. Hoe?

34. Opdracht: toongenerator werkt automatisch in "mix-modus"
Het mixen van twee willekeurige geluiden moet altijd gebeuren met de "mix-formule" zoals we dat hierboven hebben gedaan. Maar als je de toongenerator gebruikt, wordt een toon altijd automatisch "bijgemixt" bij het geluid dat al op een spoor staat! Dat is zo gedaan om muziekzaalpraktische redenen: als je met een muziekinstrument een toon maakt, wordt die bijgemixt bij het geluid dat er in een ruimte al is, bijvoorbeeld het geluid van samenspelende muzikanten. Dus in de toongenerator zit een "ingebouwde" mixer. Dat checken we even:
a. Open ►Weerbericht.wav. Druk op **Play** en luister naar S1 en S2.
b. Open ►Preset_H4_MixModus.txt. Klik op **Start** en luister weer naar S1 en S2.
Wat hoor je?

Een bijzondere mix: fouriersynthese

Een heel belangrijke stelling in de wiskunde, die vooral in de theorie van trillingen en golven een fundamentele rol speelt, is de ***stelling van Fourier***.
Jean-Baptiste Joseph Fourier (1768 - 1830) toonde aan dat:

Elk periodiek signaal kan worden samengesteld door een mix van sinusgolven met harmonische frequenties (***fouriersynthese***). De amplitudes en de fases van die sinusgolven kunnen berekend worden uit de golfvorm van één periode van dat gegeven signaal (***fourieranalyse***).

Wat je de Spectrum analyzer en de Sonogramfunctie steeds hebt zien doen was: van opeenvolgende korte geluidsfragmentjes de fourieranalyse berekenen en de gevonden amplitudewaarden weergeven in een grafiek. (De fases zijn voor de waarneming van de klank nauwelijks van belang.)

Een fouriersynthese is dus een mix van sinusgolven met harmonische frequenties en gegeven amplitudes en fases. Hoe meer harmonischen je toevoegt hoe beter de golfvorm van de sinus-mix gaat lijken op de oorspronkelijke golfvorm. Van dat laatste hebben we al voorbeelden gezien, want wat de toongenerator doet is niets anders dan

fouriersyntheses maken van bijvoorbeeld een zaagtand- of een parabool- of een blokgolf. Zie ook opdracht 35 en 36.

Als je de fourieranalyse toepast op een zaagtandgolf dan krijg je als resultaat een formule die je als volgt onder woorden kunt brengen:

De amplitudes van de harmonischen van een zaagtandgolf zijn omgekeerd evenredig met het rangnummer van de harmonischen. Dus als de amplitude van de grondtoon gelijk is aan A, dan is de amplitude van de *k*-de harmonische gelijk aan A/k.

Voorbeeld
Als een zaagtandgolf een grondtoon (= harmonische nummer 1) heeft waarvan de amplitude gelijk is aan 1000, dan zijn de amplitudes van de volgende harmonischen gelijk aan 1000/2 = 500, 1000/3 = 333, 1000/4 = 250... etc.

Dat gaan we nu uittesten.

35. Opdracht: fouriersynthese van een zaagtand

a. Open ►Preset_H4_FouriersyntheseZaagtand.txt. Vul de ontbrekende gegevens in.
b. Op spoor S2 verschijnt een zaagtand die bestaat uit vijf harmonischen. Vergelijk de resultaten. Je ziet dat de toon op S2 een kleinere amplitude heeft dan die van S1. Ga naar **Lijst** om vast te stellen hoe groot de amplitudes van S1 en S2 zijn.

Je ziet dus dat de mix van de sinussen een amplitude heeft die kleiner is dan de som van de amplitudes van de afzonderlijke harmonischen. (Dat komt omdat de maxima van de afzonderlijke sinusgolven op verschillende tijdstippen vallen)

36. Opdracht: fouriersynthese van een blokgolf

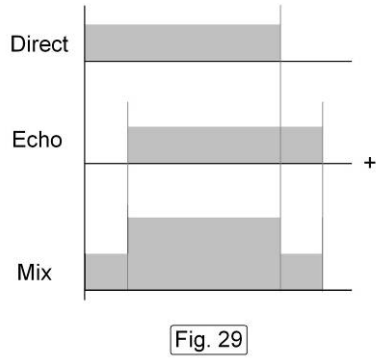
Door enkele kleine aanpassingen te maken in de code van ►Preset_H4_FouriersyntheseZaagtand.txt kun je van de zaagtandgolf een blokgolf maken! Want net zoals bij de zaagtand zijn ook bij de blokgolf de amplitudes van de harmonischen omgekeerd evenredig met het rangnummer. Het enige verschil is dat de amplitudes van alle even harmonischen van de blokgolf gelijk zijn aan 0. Een blokgolf heeft dus alleen maar oneven harmonischen!

Opmerking
Je hebt in de preset-tekst gezien dat er voor elke harmonische een apart toongenerator code-blok staat. Hadden we 100 harmonischen willen berekenen, dan hadden we 100 codeblokken moeten invullen! Dat is natuurlijk geen erg handige methode. In hoofdstuk 6 presenteren we de

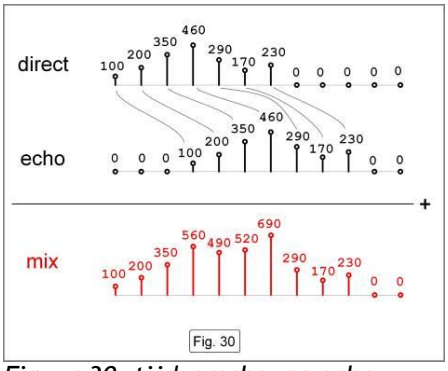
`FOR...TO...NEXT` instructie waardoor je voor fouriersynthese slechts één code-blok nodig hebt, ongeacht het aantal harmonischen.

4.3 Echo

Een ***echo*** is een geluidsreflectie. Je hoort hetzelfde geluid nog eens, maar dan ietsje later en meestal ook ietsje zachter. Op het moment dat de echo begint, gaat het directe geluid gewoon door. Je luistert dus naar een mix van het directe geluid en het echogeluid. Figuur 29 geeft dit schematisch weer.
Het enige nieuwe is dat je nu een geluid gaat mixen met een tijdverschoven kopie van datzelfde geluid. Die tijdverschuiving kun je realiseren door alle samples van een spoor te kopiëren naar een ander spoor en ze daarbij tevens een aantal plaatsen verderop neer te zetten, zoals te zien in figuur 30. Op een derde spoor kun je dan een gewone mix maken van de eerste twee sporen.



Figuur 29: echo.



Figuur 30: tijdverschoven echo.

Echoformule

Als we het echo-effect in woorden formuleren, kunnen we daaruit direct een wiskundige formule afleiden, waarmee we de mix van direct geluid en de tijdverschoven kopie daarvan kunnen berekenen. Het enige wat we moeten afspreken, is dat we de echotijd, uitgedrukt als aantal samples, *N* noemen.
Het echo-effect brengen we als volgt onder woorden:

echomix = direct geluid gemixt met *tijdverschoven kopie*

$$S2[n] = S1[n] + S1[n-N] \qquad (6)$$

Meestal zal de reflectie ook een ander volume hebben dan het directe geluid. Ook dat kunnen we eenvoudig formuleren als we *a* definiëren als versterkingsfactor:

echo-mix = direct geluid gemixt met *versterkte, tijdverschoven kopie*

$$S2[n] = S1[n] + a * S1[n-N] \qquad (7)$$

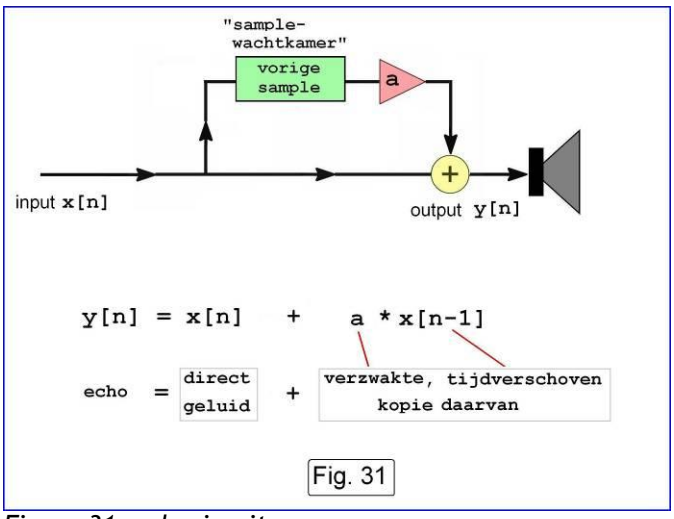
Echocircuit

Je kunt het echo-effect ook beschouwen als een circuit waar de geluidssamples langs verschillende paden doorheen wandelen en zo een echo-effect veroorzaken. In figuur 31 zie je het circuit van de echoformule waarbij we de waarde van *N* gelijk hebben gemaakt aan 1.
Links bij ***input*** betreedt het directe geluid het circuit. Rechts, bij ***output*** verlaat de echomix het circuit.
Hierboven was S1 het inputsignaal S2 het outputsignaal.

Afspraak
We noemen voortaan het inputsignaal x[n] en het outputsignaal y[n].

Nu kunnen we de volgende formule schrijven voor het echocircuit in figuur 31:

$$y[n] = x[n] + a \cdot x[n-1] \qquad (8)$$



Figuur 31: echocircuit.

Zo'n discreet systeem werkt in "arbeidsslagen". Bij elke arbeidsslag betreedt één *x*-sample het circuit en wordt één waarde van *y* berekend. Het inputsignaal bewandelt twee verschillende paden, één voor het directe geluid en één voor de reflectie. Het directe geluid gaat rechtstreeks naar de output, rechts. De reflectie gaat naar boven en moet in een "wachtkamer" verblijven. Zo'n wachtkamer is niets anders dan een geheugenregister dat één getal kan bevatten. De sample die bij de vorige arbeidsslag *n* – 1 de wachtkamer inging, verlaat die weer bij arbeidsslag *n*, wordt vermenigvuldigd met versterkingsfactor *a* en tot slot opgeteld bij sample *n*.

Op die manier bevat dit systeem z'n eigen "verleden". Dat verleden heeft de vorm van een wachtkamer die een sample gedurende 1 arbeidsslag vasthoudt. Tussen twee arbeidsslagen verstrijkt T_s seconde.

Beginconditie

Wil je met dit systeem een echo-effect toevoegen aan een geluid, dan is er maar één ding waar je vooraf op moet letten, namelijk wat voor getal er in de wachtkamer zit op tijdstip 0. We noemen dit de ***beginconditie*** van het systeem. Meestal gaan we ervan uit dat dit getal gelijk is aan nul en zeggen dan kortweg: beginconditie nul. Dit is hetzelfde als zeggen dat het stil is, "voordat het geluid begint".

Differentievergelijking

Met zo'n vergelijking als $y[n]=x[n]+a\cdot x[n-1]$ beschrijf je de relatie tussen input en output van een systeem. Links van het = teken staat de output *y* en rechts staat wat je moet uitrekenen om die output te krijgen. We noemen zo'n vergelijking een ***differentievergelijking***. De herkomst van dat woord wordt duidelijk als je substitueert $a = -1$. Dan komt er: $y[n]=x[n]-x[n-1]$. In woorden uitgedrukt: "elke sample van het signaal *y* is het verschil (de differentie) van twee opeenvolgende samples (met rangnummers *n* en *n*−1) van signaal ***x***".

2. Achtergrondinformatie: differentiequotiënt

Bij wiskunde heb je al kennis gemaakt met differentiëren en het differentiequotiënt $\frac{\Delta y}{\Delta x}$; dat is het verschil Δy tussen twee naburige functiewaarden gedeeld door hun onderlinge afstand Δx langs de horizontale as. Bij discrete signalen en functies is de afstand tussen alle samples altijd even groot en kun je dat verschil dus eenvoudig 1 stellen. Dan wordt het differentiequotiënt het verschil tussen twee naburige functiewaarden. Daarom spreek je bij discrete signalen van de differentie en niet van differentiequotiënt.

Werkt de differentie *ook* verhelderend?

Wat de afgeleide is voor continue functies, is de differentie voor discrete functies. In hoofdstuk 1 zeiden we dat "differentiëren verhelderend werkt" en dat dus de afgeleide van een functie helderder klinkt dan de functie zelf. Aangezien de differentie een benadering is van de afgeleide, kun je je de volgende vragen stellen:

1. Klinkt een geluid ook helderder als je er een "differentie-"echo aan toevoegt volgens $y[n]=x[n]-x[n-1]$?
2. Krijg je zo ook een goede benadering van de afgeleide?

Dat onderzoeken we in opdracht 37.

37, Opdracht: de differentie van een parabooltoon
Open ►Preset_H4_DifferentieParabool.txt. De code bestaat uit twee blokken a en b.

a. Maak met de toongenerator op spoor S1 een parabooltoon van 10 sec, amplitude 28000 en frequentie 100 Hz. Aantal harmonischen zet je op 200.
b. Welke golfvorm krijg je op spoor S2 als je de volgende bewerking uitvoert:
   $S2[n] = S1[n] - S1[n-1]$ voor $1 \leq n \leq 10\ F_s$
   Het geluid is vrij zacht; gebruik de Y-slider van S2 om de grafiek verticaal uit te vergroten en de golfvorm te bekijken.
c. Waarom zou je een foutmelding hebben gekregen als je bij b zou zijn uitgegaan van
   $0 \leq n \leq 10\ F_s$ in plaats van $1 \leq n \leq 10\ F_s$?

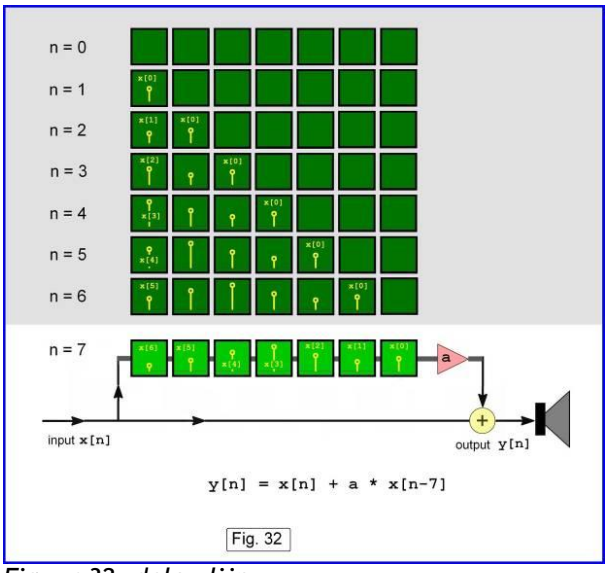
Filter

In paragraaf 1.1 las je: "Geluidsbewerkingen die de klankkleur van een geluid veranderen (bijvoorbeeld helderder of doffer maken) noemen we ***filters***."
In opdracht 37 heb je een echo-effect gebruikt om een parabooltoon te veranderen in een zaagtandtoon. De laatste klinkt veel helderder dan de eerste. Je hebt door het echo-effect de klankkleur van de toon veranderd. Kennelijk heeft dit echosysteem dus de werking van een filter. Of korter: dit systeem *is* een filter! Bewezen kan worden dat elk echo-effect altijd tevens een filtereffect is (en omgekeerd). In hoofdstuk 5 gaan we daar nog iets verder op in.

Delay-lijn

Wil je een echo maken die niet 1 maar N sampletijden duurt, dan heb je natuurlijk ook een circuit nodig met N wachtkamers! Daarin schuifelen de samples van de ene naar de volgende wachtkamer, zodat elke sample N sampletijden nodig heeft om van input naar output te wandelen. Zie figuur 32.

Zo'n lange rij wachtkamers noemen we een ***delay-lijn***. Delay-lijnen zijn de basis van alle echo-effecten, snaar- en buissimulaties, galm, etc.
In de figuur is $N = 7$; je ziet dat de samples in de delay-lijn stapje voor stapje door het circuit heen wandelen en pas 7 sampletijden T_s na hun entree bij de input zijn aangekomen bij de output.



Figuur 32: delay-lijn.

38. Opdracht: delay-lijn

a. In figuur 32 is de versterkingsfactor *a* getekend rechts van de reeks wachtkamers. Had *a* ook links mogen staan, dus vóór de wachtkamers? Of ergens in het midden?

b. Open ►Preset_H4_Delaylijn.txt. Neem eens aan dat de versterkingsfactor *a* = 1. Maak met de toongenerator een sinustoon op S1 waarvan je de frequentie zo kiest dat het outputsignaal op S2 een amplitude heeft die twee keer zo groot is als die van S1. Wat is de laagste frequentie die je daarvoor kunt kiezen? Dat wil zeggen: wat is laagst mogelijke sinustoon die door dit echocircuit een twee maal zo grote amplitude krijgt?
Hints:
1. Kijk in hoofdstuk 2 nog eens naar de definitie van een periodiek signaal.
2. Als het inputsignaal de eigenschap heeft dat $x[n] = x[n-7]$, hoe komt de differentievergelijking van dit systeem er dan uit te zien?

c. Welke waarde moet je voor *a* kiezen om ervoor zorgen dat bij diezelfde frequenties de output op S2 een amplitude nul krijgt?

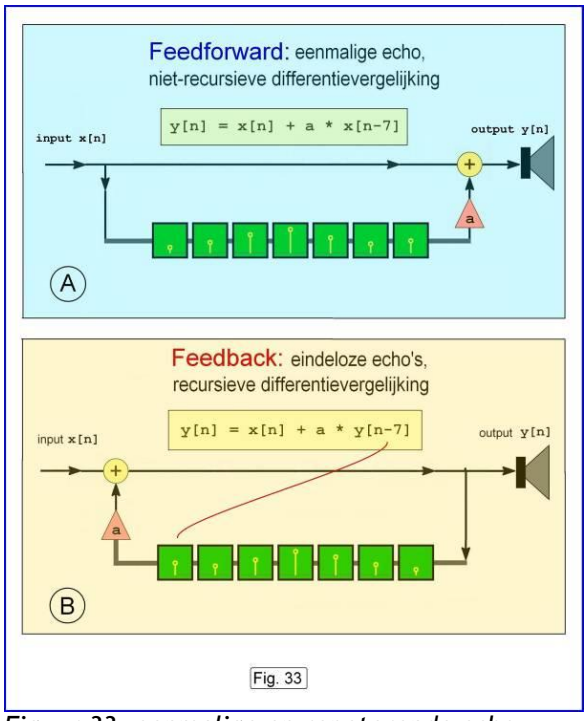
39. Opdracht: weerbericht met echo

a. Open ►Weerbericht.wav. Hoeveel seconde duurt het geluid?

b. Open ►Preset_H4_Echo.txt. Vul de ontbrekende gegevens in zodat een echo ontstaat met een echotijd van 0,5 sec, waarbij de amplitude van de reflectie de helft is van die van het directe geluid.

4.4 Echo-feedback

Met het circuit uit paragraaf 4.3 kun je eenmalige echo's realiseren die zich voordoen bij alle open objecten, zoals een bosrand langs een veld. Maar bij gesloten ruimten, zoals bijvoorbeeld een trappenhuis, ligt het

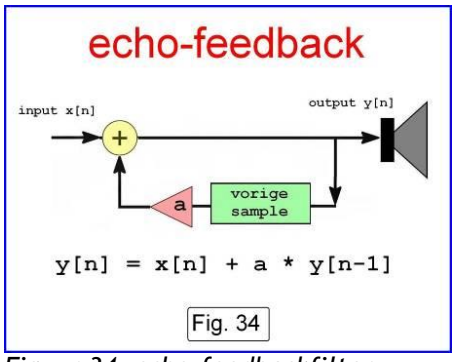


Figuur 33: eenmalige en repeterende echo.

anders. Daar blijft een geluid in principe oneindig lang heen en weer kaatsen tussen de wanden. Dat je na een tijdje niks meer hoort is het gevolg van demping en hoe sterk die demping is hangt af van de absorptie-eigenschappen van de wanden.
Ook zo'n eindeloos repeterende echo kun je nabootsen. Zie figuur 33B. Net zoals bij de eenmalige echo (A) heb je ook hier een rechtstreekse verbinding tussen input en output; en ook hier wordt gebruik gemaakt van een delay-lijn. Het verschil is dat bij B niet de input maar de output in de delay-lijn wordt geplaatst! De output wordt teruggevoerd naar en opgeteld bij de input. Daarom noemen ze dit een ***feedbackcircuit*** en een eindeloos repeterende echo een ***echo-feedback***. Het circuit van eenmalige echo wordt als tegenhanger ook wel ***feedforward*** genoemd.
Als je kijkt naar het circuit in figuur 33B, dan kun je de bijbehorende differentievergelijking afleiden. Die luidt:

$$y[n] = x[n] + ay[n-N] \qquad (9)$$

Merk op dat je om *y[n]* te kunnen uitrekenen, anders dan bij de eenmalige echo, eerst *y[n–N]* moet hebben uitgerekend. Je moet dus "teruglopen" naar een eerder resultaat, vandaar de naam ***recursieve*** differentievergelijking (Latijn: recursus = terugloop, terugkeer). Je kunt het recursieve karakter van een differentievergelijking herkennen doordat rechts van het = teken een vorige output, namelijk *y[n–N]*, te vinden is. Bij het feedforward-systeem daarentegen zie je rechts van het = teken uitsluitend inputsamples *x*, zie bijvoorbeeld formule (8).



Figuur 34: echo-feedbackfilter.

Feedbackcircuit met *N* = 1

Laten we als voorbeeld eens het echo-feedbackfilter bekijken dat wordt beschreven door de volgende recursieve differentievergelijking:

$$y[n] = x[n] + ay[n-1] \qquad (10)$$

Een afbeelding van dit circuit zie je in figuur 34.
We gaan ervan uit dat de beginconditie nul is (dus op het tijdstip *n* = 0 is de wachtkamer leeg, dat wil zeggen bevat het getal 0). Verder veronderstellen we dat het inputsignaal *x[n]* is:

x[0] = 1
x[n] = 0 voor elke waarde van *n* ≠ 0.

Dus *x* is de rij getallen 1, 0, 0, 0, 0, ... (en klinkt als een eenmalige, scherpe tik, als je *x[0]* tenminste voldoende versterkt).

Wat krijgen we dan voor de output *y[n]*?
In de tabel hieronder hebben we enkele waarden van *y[n]* uitgerekend:

n	x[n]	a * y[n-1]	y[n]
0	x[0] = 1	a * y[-1]= 0	y[0] = 1
1	x[1] = 0	a * y[0] = a	y[1] = a
2	x[2] = 0	a * y[1] = a^2	y[2] = a^2
3	x[3] = 0	a * y[2] = a^3	y[3] = a^3

Tabel met outputwaarden.

40. Vraag

Let even op dat negatieve rangnummer van `y[-1]`.We hadden de afspraak gemaakt dat buffers in WaveWizard nooit een negatief rangnummer kunnen hebben.
Wat wordt er eigenlijk bedoeld met `y[-1]`?

Je ziet dat *y[n]* een ***exponentiële reeks*** vormt: a^0, a^1, a^2, ...

Stabiliteit

Als de versterkingsfactor *a* groter is dan 1, dan zal *y[n]* exponentieel groeien; vroeger of later zal *y[n]* de maximum amplitude die we voor geluidssamples hebben afgesproken, namelijk 32767, overschrijden.

We spreken daarom van een ***instabiel*** systeem als de versterkingfactor a ≥ 1.

Instabiele systemen zijn in de meeste toepassingen, zoals audio, onrealistisch en onbruikbaar.

Voorbeeld Knars Kraak!

Stel dat in het systeem van figuur 34 de waarde van de versterkingsfactor *a* gelijk is aan 1,001. Na hoeveel samples overschrijdt de uitwijking dan de grens van 32767?

Noemen we het sample-rangnummer waarbij die grens wordt overschreden *N*, dan krijgen we de reeks a^0, a^1, a^2, ... a^{N-1}, a^N, ... Dan is $y[N] = 1{,}001^N$.

We vinden dan voor *N*:

$1{,}001^N = 32767$ → we nemen aan beide kanten van het = teken de logaritme:

$\log(1{,}001^N) = \log(32767)$ →

$N \log(1{,}001) = \log(32767)$ →

$$N = \frac{\log(32767)}{\log(1{,}001)} = 10402{,}375$$

Aangezien *N* een sample-rangnummer is, en dus een geheel getal, vinden we dat reeds na 10403 $T_s \approx 0{,}25$ sec de amplitude het maximum overschrijdt!

Het is dus enorm oppassen geblazen met feedbacksystemen! Je kunt er je oren (en je boxen) een behoorlijke opdonder mee bezorgen! Als *a* ook maar een fractie groter is dan 1, hoor je binnen de kortste keren: KNARSKRAAK!!

Kortom:

Let erop dat de waarde van de versterkingsfactor a in een audio-feedbacksysteem kleiner is dan 1!

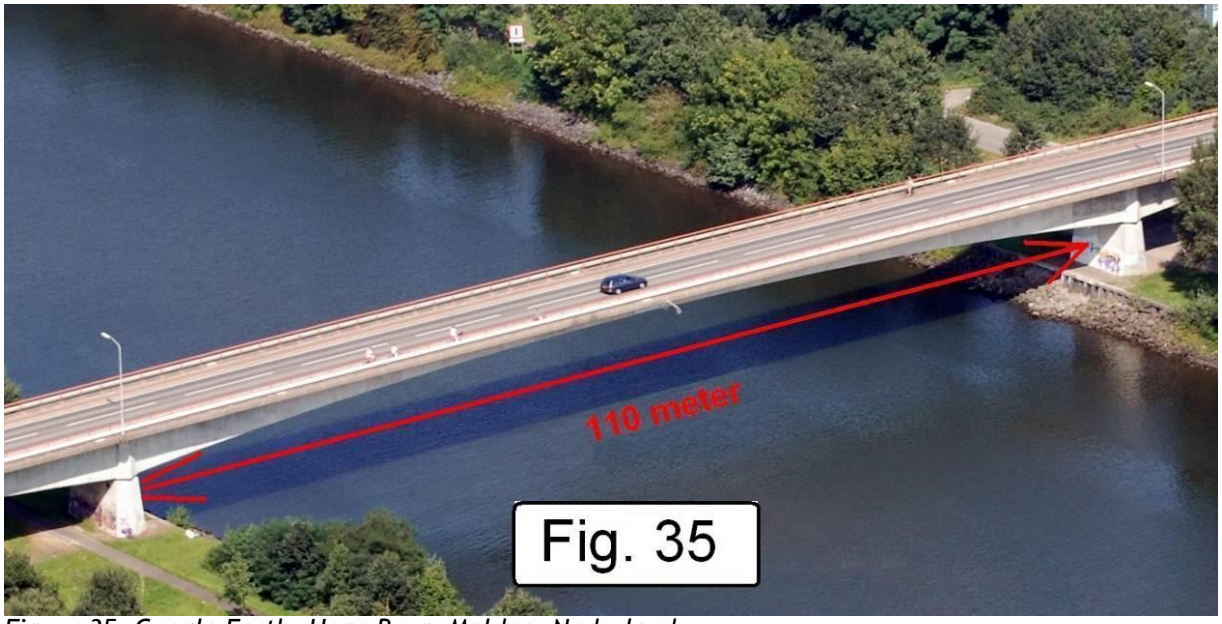
41. Opdracht: is systeem *y[n] = x[n] + y[n−1]* stabiel?

a. Als in het systeem van figuur 34 de waarde van de versterkingsfactor *a* gelijk is aan 1, dan krijg je de differentievergelijking $y[n] = x[n] + y[n-1]$. Stel eens dat het inputsignaal *x[n] = c* voor *n ≥ 0*. Vul de tabel hieronder verder in.

n	x[n]	y[n-1]	y[n]
0	x[0] =	y[-1]= 0	y[0] =
1	x[1] =	y[0] =	y[1] =
2	x[2] =	y[1] =	y[2] =
3	x[3] =	y[2] =	y[3] =

Tabel opdracht 41.

b. Wat kun je zeggen over de stabiliteit van het systeem als versterkingsfactor *a* gelijk is aan 1?



Figuur 35: Google Earth, Hoge Brug, Malden, Nederland.

42. Opdracht: Hoge Brug Malden

In figuur 35 (Google Earth, zoek op "Hoge Brug, Malden, Nederland") zie je een van de vele bruggen over het Maas-Waalkanaal bij Nijmegen. Als je onder de brug pal voor de pijler staat (bij de graffiti) en je klapt in je handen, dan hoor je enkele echo's, steeds zachter. Het zijn reflecties tussen de twee evenwijdige, betonnen (en dus weinig geluid absorberende) pijlers.

Onder de brug staan vaak scholieren akoestische proeven te doen door lege bierflessen tegen het beton te smijten en aandachtig te luisteren naar het inspirerende geluid van repeterend, brekend glaswerk.

a. Als de geluidssnelheid 340 m per sec is en de afstand tussen de pijlers 110 m, na hoeveel seconde hoor je dan de eerste echo?
b. Uit hoeveel samples moet een delay-lijn bestaan om die echotijd te krijgen? Druk dit aantal samples uit in Fs.
c. Open ►Brekend glas.wav. En open ►Preset_H4_HogeBrug.txt. Zet a = 0,5. (Een meer realistische waarde voor a is 0,1, maar voor de duidelijkheid van het effect is het beter dat je a eerst wat groter neemt. Vul de ontbrekende code in en klik op Start.
d. Kies N zodanig dat je per seconde tien echo's hoort.
e. Als je nu kiest a = 0,9, dan krijg je een foutmelding dat het geluid te hard is. Daarom ga je in de formule van echo-feedback een getal invoeren zodanig dat het inputsignaal S1 voldoende verzwakt zodat je geen foutmelding krijgt.
f. Kies N zodanig dat je per seconde 100 echo's hoort.
g. Maak nu achtereenvolgens a gelijk aan 0,99 en 0,999 (het kan zijn dat je de input nu nog wat meer moet afzwakken).
h. Wat moet je voor a invullen om een zuiver periodieke toon te krijgen?

Conclusies echo-feedback

Uit opdrachten 41 en 42 kunnen we enkele belangrijke conclusies trekken:

1. Een echo-feedbacksysteem kan elk willekeurig geluidsfragment "uitbreiden" tot een periodieke toon! Dat is het geval als de versterkingsfactor gelijk is aan 1. (Waarmee het systeem in feite onstabiel is, want de amplitude groeit dan weliswaar niet exponentieel, maar gaat ook niet naar nul. Dat laatste is alleen het geval als de versterkingsfactor < 1 is! Alleen dàn spreek je van een stabiel systeem.) Je kunt een echo-feedbacksysteem dus gebruiken als toongenerator!
2. We hebben in opdracht 30, gezien hoe een geluid lineair in amplitude afneemt. Toch komt zo'n lineaire demping eigenlijk niet zo vaak voor bij trillende voorwerpen. Veel vaker is er sprake van een ***exponentiële demping***. Een heel goed voorbeeld daarvan is een stemvork. Hoe exponentiële demping precies tot stand komt, is te verklaren aan de hand van een echo-feedbacksysteem. We hebben gezien dat zo'n systeem een exponentiële reeks oplevert en als de versterkingsfactor *a* < 1, dan neemt die steeds af en gaat naar nul. Dan heb je demping. Heel veel geluidsbronnen, zoals muziekinstrumenten, zijn goed te modelleren door middel van feedbacksystemen. Een echt spectaculair voorbeeld daarvan zien we in hoofdstuk 7, als we het gaan hebben over de "snaarplukvergelijking".

Samenvatting

Versterken

1. Een geluid versterken doe je door alle samples te vermenigvuldigen met een constante. Bijv. `s1[n] = c * s1[n]`. (Hierin betekent het = teken dat de oorspronkelijke sample wordt overschreven door de vermenigvuldigde sample).
2. Een geluid geleidelijk in volume laten toe- of afnemen (fade in of fade out) doe je door het "sample-gewijs" te vermenigvuldigen met een (al dan niet lineair) ***stuursignaal***. Bijvoorbeeld `s2[n] = g[n] * s1[n]`. Het gemakkelijkst is het om `g[n]` zo te construeren dat het bereik ervan ligt tussen 0 en 1.

Mixen

1. ***Mixen*** is het samenvoegen van geluiden op verschillende sporen tot één geluid op één spoor. Je realiseert een mix door "sample-gewijs" op te tellen. Bijvoorbeeld: een mix van spoor S1 en spoor S2 op spoor S3 maak je door middel van de formule:

   `S3[n] = S1[n] + S2[n].`
2. Mixen en versterken kun je verrichten in één enkele operatie. Voorbeeld:

   `S3[n] = a*S1[n] + b*S2[n]`, waarin *a* en *b* versterkings- en/of verzwakkingsfactoren zijn. Datzelfde geldt voor een mix waarin ook

stuursignalen voorkomen, bijvoorbeeld: `s3[n] = g[n]*S1[n] + h[n]* s2[n]`.

3. ***Fouriersynthese***: elk periodiek signaal kan worden samengesteld door een mix van sinusgolven met harmonische frequenties.
4. ***Fourieranalyse***: de amplitudes en de fases van die sinusgolven kunnen berekend worden uit de golfvorm van één periode van een gegeven (periodiek) signaal.

Echo-feedforward

1. Een echo is een mix van een geluid met een versterkte (of verzwakte), tijdverschoven kopie daarvan.
2. In formule: `y[n] = x[n] + a*x[n-N]` . Hierin stelt *N* de echotijd voor, uitgedrukt in sampletijden T_s en *a* de versterkingsfactor van de reflectie.
3. De echovergelijking noemen we een ***niet-recursieve differentievergelijking*** : links van het = teken staat het ***output***signaal, *y[n],* en rechts van het = teken staat welke berekening je moet uitvoeren op samples van uitsluitend het ***input***signaal *x[n].*
4. Met een niet-recursieve differentievergelijking kun je alleen maar een eenmalige reflectie realiseren, zoals de echo van een bosrand (***feedforward***).
5. Elk echo-effect is tevens een filtereffect dat wil zeggen dat het de klankkleur van een geluid verandert, bijvoorbeeld doffer of scherper maakt.
6. De term ***differentievergelijking*** wordt goed verklaard door het bijzondere geval `y[n] = x[n] - x[n-1]`. Bijvoorbeeld de differentie van een parabooltoon is een zaagtandtoon .
7. Niet alleen differentiëren (continue signalen en functies) maar ook de differentie (discrete signalen en functies) maakt een geluid helderder van klank. (Een zaagtand bijvoorbeeld klinkt helderder dan parabool).
8. Een echo-effect is een voorbeeld van een signaalverwerkend systeem dat, in tegenstelling tot bewerkingen zoals versterken en mixen, moet beschikken over geheugenregisters ("wachtkamers").

Echo-feedback

1. ***Echo-feedback*** geeft een eindeloos repeterende echo en realiseer je door bij de berekening van het outputsignaal *y* niet alleen de input *x*, maar ook een vorige output *y* te gebruiken. Zo krijg je een ***recursieve*** differentievergelijking, bijv. `y[n] = x[n] + a*y[n-N]`.
2. In tegenstelling tot feedforward kan een feedbacksysteem ***instabiel*** zijn. Dat is het geval als de versterkingsfactor *a* groter is dan of gelijk is aan 1. Van stabiliteit is alleen sprake als $a < 1$.
3. Net zoals echo-feedforward heeft ook echo-feedback een filterwerking. Die is zelfs nog veel krachtiger en duidelijker waarneembaar dan die van feedforward.

4. In tegenstelling tot een feedforwardsysteem kun je een feedbacksysteem gebruiken om tonen mee te maken.
5. Stabiele feedbacksystemen vertonen een ***exponentiële demping***. Die vorm van demping komt veel vaker voor dan de lineaire en klinkt (dus!?) veel "natuurlijker".

Woordenlijst

attack	§1	infaden	§1
beginconditie	§3	input	§3
`Bewerk signaal`	§1	lus	§1
blokgolf	§2	mixen	§2
contourgenerator	§1	niet-recursieve differentievergelijking	§4
decay	§1	outfaden	§1
delay-lijn	§3	output	§3
differentievergelijking	§3	recursieve differentievergelijking	§4
echo	§3	release	§1
exponentiële reeks	§4	"sample-gewijs vermenigvuldigen"	§1
exponentiële demping	§4	"sample-gewijs optellen"	§2
feedback	§4	stabiliteit	§4
feedforward	§4	stelling van Fourier	§2
filter	§3	stuursignaal	§1
fourieranalyse	§2	sustain	§1
fouriersynthese	§2	toongenerator (ASDR)	§1

5 Filters

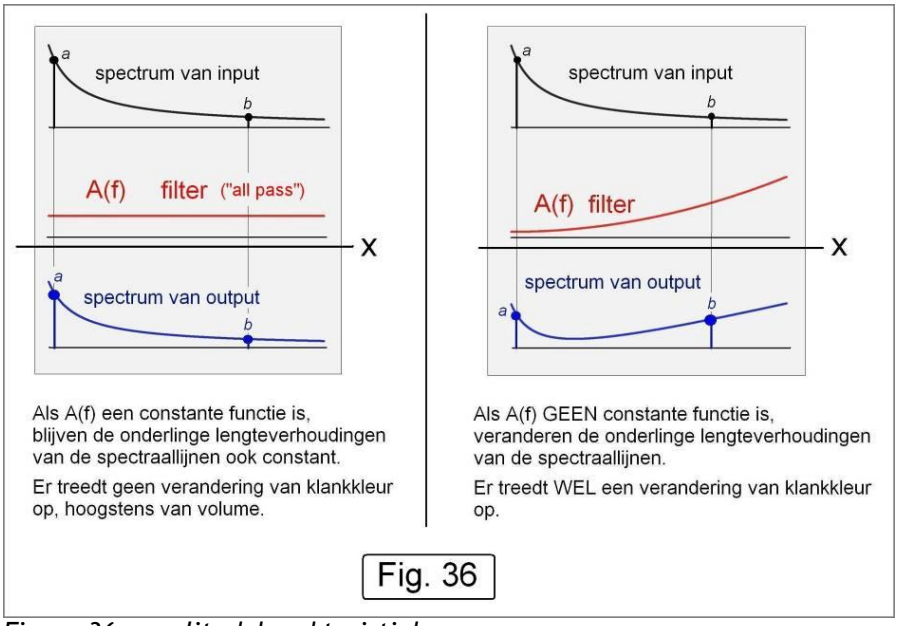
We hebben het inmiddels al vaak opgemerkt: als je een geluid filtert, verandert de klankkleur ervan. Daarom zijn filters voor alle vormen van sound design erg belangrijk. In dit hoofdstuk leer je verschillende soorten krachtige filters kennen en gebruiken. In de laatste paragraaf krijg je wat meer kijk op de werking van een filter.

5.1 De amplitudekarakteristiek A(f)

Klankkleurverandering

In hoofdstuk 2 heb je kunnen vaststellen dat de overgang van een zaagtand naar een parabool zichtbaar is in het spectrum (en in het sonogram). Blijkbaar geldt de volgende regel:

Een verandering in de lengteverhoudingen van de spectraallijnen hoor je als een verandering van klankkleur. Let op het woord verhoudingen! In figuur 38 wordt dat toegelicht.



Figuur 36: amplitudekarakteristiek.

Zowel in de linker als in de rechter afbeelding wordt het spectrum van een signaal (input) vermenigvuldigd met een functie *A(f)*. Die functie *A(f)* is gewoon een stuursignaal zoals we dat in H4 al tegenkwamen, maar er zijn twee belangrijke verschillen:

1. *A(f)* is geen functie van de tijd, maar van de frequentie;
2. *A(f)* is niet discreet maar continu.
   (Immers elke frequentie die kleiner dan of gelijk is aan de

Nyquistfrequentie kan in een signaal voorkomen! Dus het domein van f is: $0 \leq f \leq 0{,}5\ F_s$.)

De resultaten zijn onder in figuur 36 in blauw afgebeeld. Dit is het productspectrum (output).
Links is *A(f)* een constante functie. Daarom blijven de lengteverhoudingen van de spectraallijnen *a* en *b* ongewijzigd en is de klankkleur van het input- en van het outputsignaal dezelfde. Een dergelijk filter heet een ***"all pass" filter***: het "laat alle frequenties ongewijzigd door".
Rechts is *A(f)* een of andere niet-constante functie (het doet er niet toe wat voor functie; behalve dat *A(f)* voor geen enkele waarde van *f* negatief mag (en kàn) zijn). Hier zijn de lengteverhoudingen van de spectraallijnen *a* en *b* wèl gewijzigd. Rechts is dus wèl sprake van een filterwerking die de klankkleur wijzigt.

Dubbele beschrijving van filter

Wat er in figuur 36 gebeurt, is niets meer en niets minder dan een vermenigvuldiging van twee functies. Dat het spectrum van de output wordt gewijzigd als *A(f)* geen constante functie is en dat daardoor dan eveneens de klank verandert, daarmee hebben we nog niet bewezen dat de werking van echte filters ook inderdaad zo beschreven kan worden. In hoofdstuk 4 zagen we dat filterwerking ontstaat door het directe geluid te mixen met (een of meerdere) echo's. Wat heeft dat te maken met zo'n spectrale vermenigvuldiging?
We zullen in paragraaf 5.4 zien dat die twee zo verschillende dingen - echo's mixen en spectra vermenigvuldigen - op hetzelfde neerkomen! Je kunt niet het één doen en het andere laten! Het zijn twee verschillende beschrijvingen van hetzelfde proces. Al in hoofdstuk 2 zijn we begonnen met die dubbele beschrijving van geluiden, namelijk zowel in termen van tijd als van frequentie. Denk bijvoorbeeld aan Scoop en Spectrum / Analyzer die respectievelijk het tijdplaatje en het frequentieplaatje van dezelfde trilling weergeven. Als je aan een geluid een echo toevoegt (= tijd-effect), dan creëer je automatisch een functie *A(f)* (= frequentie-effect). Omgekeerd, als we het spectrum van een of ander geluid vermenigvuldigen met een niet-constante functie *A(f)*, dan voegen we altijd automatisch een echo-effect toe aan het geluid.
We zullen dat in paragraaf 5.4 nader bekijken. Maar nu eerst gaan we praktisch aan de slag met een paar krachtige filters, zodat je een beter idee krijgt hoe ingrijpend je er een klank mee kunt veranderen. We beginnen met een wat duidelijkere omschrijving van die "filterfunctie" *A(f)*.

Definitie amplitudekarakteristiek *A(f)*

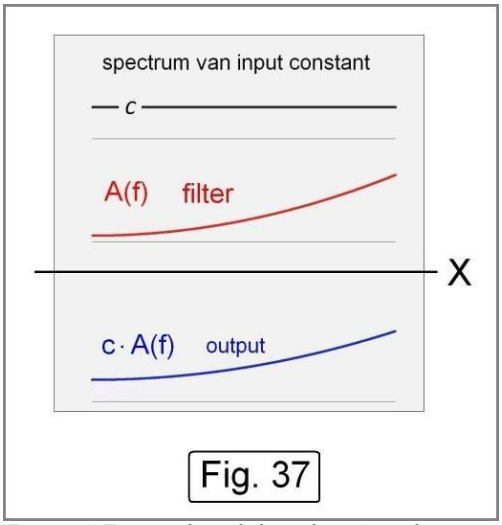
We noemen de rode grafieken in figuur 36 de ***amplitudekarakteristiek*** *A(f)* van een filter en definiëren die als volgt:

De amplitudekarakteristiek *A(f)* van een filter is een continue functie van de frequentie *f* die aangeeft hoeveel de amplitude van een sinustoon met frequentie f wordt versterkt/verzwakt door het filter.

De amplitudekarakteristiek geeft je in één oogopslag een idee van de klankeigenschappen van het filter. Dat gaan we na in de volgende opdracht.

43. Opdracht: weerbericht gefilterd
Open ►Weerbericht.wav. Open ►Preset_H5_Weerbericht_Laagdoorlaat.txt. Klik op **Start** en ga naar het Scoop / Spectrumvenster. Beluister eerst S1 en dan S2.

a. Hoe omschrijf je het geluid van S2?
b. Wat is het verschil in de spectra van S1 en S2?



Figuur 37: amplitudekarakteristiek, constante input.

5.2 "Witte" geluiden

Geschaalde versie van *A(f)* afbeelden

In opdracht 43 hierboven heb je een indruk gekregen hoe ingrijpend de werking van een filter kan zijn. Maar wat is nu de amplitudekarakteristiek *A(f)* van dit filter?
We passen een trucje toe waarmee we van elk willekeurig filter direct de grafiek van *A(f)* kunnen afbeelden, zonder dat we daarbij iets moeten berekenen. Wel is het zo dat *A(f)* daarbij in verticale richting kan zijn opgeschaald (vergroot of verkleind) met een onbekende factor.
Kijk naar figuur 37. Stel je hebt een geluid waarvan het spectrum een constante functie $I(f) = c$ is (boven in figuur, zwart). Halen we dat geluid door een filter met amplitudekarakteristiek *A(f)*, dan krijgen we als outputspectrum $I(f) \cdot A(f) = c \cdot A(f)$.
Dus dat outputspectrum is een "opgeschaalde" versie van *A(f)*, met *c* als schaalfactor (zo'n schaalfactor is alleen maar een vergrotings- of verkleiningsfactor voor de grafiek, precies zoals de sliders **Y-fac** doen in het Scoop / Spectrum-venster).
Let op! Deze opgeschaalde versie van *A(f)* vertelt alleen hoe het filter de lengteverhoudingen van de spectraallijnen wijzigt. Dat is voldoende om de filterwerking aan af te lezen. Wat het plaatje niet vertelt, is hoeveel elke sinus van het inputsignaal door het filter wordt versterkt of verzwakt! In paragraaf 5.4 bij opdracht 51 vind je de werkelijke schaal van *A(f)* voor een gegeven filter.

We moeten dus gaan werken met geluiden die een constant spectrum hebben. Maar wat ìs eigenlijk een geluid met een constant spectrum? We zullen daar verschillende voorbeelden van zien. Allereerst de pulstoon.

Pulstoon

Een ***pulstoon*** is een reeks pulsen op regelmatige tijdsafstanden van elkaar. Een pulstoon heeft de volgende verrassende eigenschap:

Het tijdsignaal en het spectrum van een pulstoon zijn gelijkvormig.

Dat wil zeggen dat je puur aan de grafiek (dus zonder bijschriften!) op geen enkele manier kunt zien of je te maken hebt met het tijdsignaal of met het spectrum van de pulstoon! Dat gaan we in opdracht 44 na.

44. Opdracht: pulstoon
Open ►Preset_H5_Pulstoon.txt. De toongenerator maakt golfvormen door middel van fouriersynthese (zoals je weet uit opdrachten 35 en 36). Bij de pulsgolf hebben alle sinustonen een gelijke amplitude. Dus alle sinustonen worden in het spectrum weergegeven door spectraallijnen van gelijke lengte.
Klik op **Start**. Er verschijnt een toon op S1. Ga naar Scoop / Spectrum-analyzer en bekijk het spectrum (zorg dat er geen vinkje staat in Hann-venster).
Maak de toon nog eens enkele malen met verschillende waarden voor **`Aantal harmonischen`** (tot max. 64).
Merk op dat met het toenemen van het aantal harmonischen de tijdgolfvorm steeds puls-achtiger wordt en er dus steeds meer gaat uitzien als het spectrum van de pulstoon!

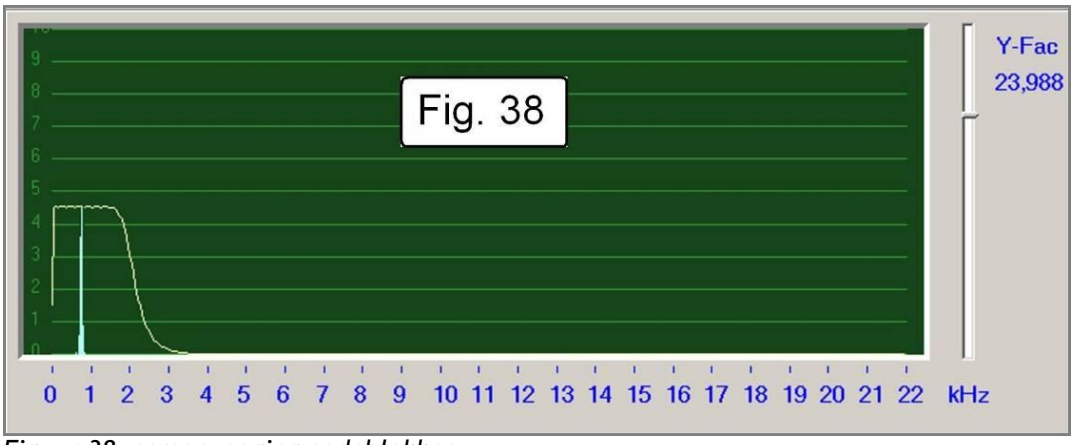
45. Opdracht: pulstoon filteren
Open ►Preset_H5_Pulstoon_Laagdoorlaat.txt, klik op **Start**.
Tijdens het runnen verschijnt in Memo de frequentie van de pulstoon: **`Fs / 256 = 172,265625`**. Kopieer dat getal (selecteer het met de muis en toets Ctrl C), ga naar het Scoop / Spectrum-venster. Drop het frequentiegetal (Ctrl V) in het donkergrijze paneel onder **Trigger Frequentie** en klik op **OK**. Nu is de scoop gesynchroniseerd met de pulstoon. Vergelijk de tonen en de spectra van S1 en S2 (je moet de **Y-fac**-slider van Spectrum wat hoger zetten). Let op de contour van het spectrum van S2. Dit is de (opgeschaalde) contour van *A(f)*!

46. Opdracht: filter doorfluiten

a. Open ►Preset_H2_Doorfluiten.txt. Deze code maakt een sinustoon aan waarvan de frequentie geleidelijk toeneemt. In hoofdstuk 2 (opdracht 15) maakte je daar al kennis mee. We kondigden aan dat je met zo'n fluittoon ook apparatuur kon testen. We gaan er nu de amplitudekarakteristiek van een filter mee vaststellen. Nu voeg je aan

deze code de preset van opdracht 45 (►Preset_H5_Pulstoon_Laagdoorlaat.txt) toe. Dat doe je met menu-functie **Open / Append Preset**. De code wordt onder die van de doorfluit-code geplaatst.

b. Smeed de twee codeblokken zo samen dat je figuur 38 te zien kunt krijgen. (Wat moet je nu invullen bij de `Duur` van het filter?) Vergeet niet om in het spectrumvenster de knop **Max** te activeren (eerst altijd ook resetten).



Figuur 38: samenvoeging codeblokken.

Het komt heel vaak voor dat je het klankeffect van een filter beter kunt testen met een geluid dat geen duidelijk waarneembare toonhoogte heeft, bijvoorbeeld witte ruis. Wat witte ruis precies is en waarom die wit genoemd wordt, zien we in opdracht 47.

47. Opdracht: witte ruis
Open ►Preset_H5_Witte_Ruis.txt, klik op **Start** en ga naar het Scoop / Spectrum-venster.
Selecteer spoor S1. Klik op **Gem**. Speel het geluid af en kijk naar het spectrum. Je ziet de blauwe spectraallijn heftig op en neer springen. Daar doorheen verschijnt een nogal bibberige, oranje grafiek. Daarin wordt bijgehouden wat van elke blauwe spectraallijn de gemiddelde waarde is sinds je voor het laatst op de **Reset** -knop drukte. Geleidelijk zie je de contour van de oranje grafiek minder bibberig worden en zich samenpersen tot een horizontale rechte.

Wat is witte ruis?

We beginnen met een eenvoudig voorbeeld. Stel je werpt een aantal keer met een dobbelsteen. Bij elke worp noteer je het aantal ogen en bepaal je het gemiddelde aantal ogen per worp. Hoe langer je doorgaat met werpen des te dichter dat gemiddelde in de buurt komt van 3,5. Bij statistiek noemde je dat de verwachting van een kansexperiment.
Nu terug naar de spectrum analyzer. De blauwe spectraallijnen kun je beschouwen als een serie dobbelstenen (met heel veel vlakjes). Elk beeldje dat op het scherm verschijnt, is een nieuwe worp met alle

dobbelstenen tegelijk. Naarmate het aantal beeldjes, worpen, toeneemt, zie je dat de oranje grafiek langzaam maar zeker convergeert (samentrekt) naar een horizontale rechte: blijkbaar hebben alle spectraallijnen dezelfde statistische verwachting. En dan spreken we van ***witte ruis*** (want in het spectrum van de kleur wit zijn alle kleuren even sterk vertegenwoordigd, met andere woorden het spectrum van wit licht is constant.)
We zeggen nu: witte ruis heeft een constant ***lange-termijnspectrum***.
Noemen we dit lange-termijnspectrum $W(\omega)$ dan geldt dus $W(\omega) = c$.
In de (oranje) grafiek van **Gem** trekt witte ruis meer en meer samen tot een horizontale rechte.

48. Opdracht: witte ruis gefilterd
Open ►Preset_H5_Ruis_Laagdoorlaat.txt, klik op **Start** en ga naar het Scoop / Spectrum-venster.
Selecteer nu spoor S2. Reset **Gem**. Nu krijg je gefilterde ruis te horen. Wat kun je zeggen over de oranje grafiek?

"Wit" geluid samengevat

Een "wit" geluid is een geluid waarvan het spectrum een contour heeft die constant is.
Die contour ziet er dus uit als een horizontale rechte lijn.

Merk op dat ook die "doorfluit"-sinus van opdracht 46 in zekere zin een "wit" geluid is!

5.3 Laagdoorlaat- en banddoorlaatfilter

Laagdoorlaatfilter

Tot nu toe bekeken we steeds hetzelfde filter. Dat type heet een ***laagdoorlaatfilter***, want het laat lage frequenties onverzwakt door (in dit geval van 0 Hz tot 2000 Hz) en hogere frequenties worden zeer sterk onderdrukt. Welk gebied wordt doorgelaten, kun je instellen met de **`Afsnijfrequentie`** (cut off frequency).

49. Opdracht: afsnijfrequentie
a. Verander de code van ►Preset_H5_Ruis_Laagdoorlaat.txt zodanig dat je op S3 een ruisklank krijgt waarbij het frequentiegebied van 0 - 5000 Hz wordt doorgelaten.
b. Vergelijk de klank met die van S1 en S2. Welke klinkt het dofst?
c. Welk effect heeft **`Flanksteilheid`** op de grafiek en het geluid? (je kunt er maximaal 50 bij invullen)

Banddoorlaatfilter

Een banddoorlaatfilter is een filter waarmee een willekeurig frequentiegebied wordt doorgelaten. In tegenstelling tot het laagdoorlaatfilter heb je bij het bandfilter te maken met twee "instelknopjes", namelijk de ***centrumfrequentie*** en de ***bandbreedte***. Wat die betekenen, leer je in de volgende opdracht.

50. Opdracht: banddoorlaatfilter

a. Open ►Preset_H5_Ruis_Banddoorlaat.txt en klik op **Start**. Bekijk de oranje grafiek van S2.
b. Wat gebeurt er met die grafiek als je de **`Centrumfrequentie`** instelt op 5000 Hz? Wat hoor je?
c. Wat gebeurt er met de grafiek als je de **`Bandbreedte`** instelt op 1500 Hz? Wat hoor je?
d. Open ►Weerbericht.wav. Verwijder uit de code in het Commando-venster alles behalve de 9 regels van **`Butterworth Banddoorlaat`**. Klik op **Start**. Is de spraak nog goed verstaanbaar? Kies ook andere waarden voor de **`Centrumfrequentie`** en de **`Bandbreedte`** en beoordeel de verstaanbaarheid.

5.4 A(f) afgeleid uit differentievergelijking

We hebben tot nu toe op twee verschillende manieren kennis gemaakt met het begrip filter. In hoofdstuk 4 zagen we dat een filterwerking ontstaat door een echo-effect, dus een tijdeffect. In dit hoofdstuk maakten we kennis met de amplitudekarakteristiek *A(f)*: een frequentie-effect.
Maar wat hebben die twee met elkaar te maken? Voordat we dat uit de doeken doen, eerst de voorbereidende opdrachten 51 en 52.

51. Opdracht: amplitudekarakteristiek van de "differentie"

In paragraaf 4.3 maakte je kennis met de vergelijking $y[n]=x[n]-x[n-1]$, de formule waaraan de differentievergelijking haar naam dankt. In deze opdracht bekijken we de amplitudekarakteristiek daarvan.

a. Open ►Preset_H5_Opdracht_51_Differentie.txt. Klik op **Start**. Ga naar het Scoop / Spectrum-venster. Stel de slider **Y-fac** van de Spectrum Analyzer op maximaal (=1000).
b. Selecteer spoor S1. Activeer en reset **Gem**. Speel S1 af. Het geluid duurt 30 sec. Zorg ervoor dat je net iets vóór het einde van het geluid op de Stopknop hebt gedrukt, anders zakt de oranje grafiek van **Gem** langzaam in ten gevolge van de stilte na 30 sec. Tot welke hoogtelijn (zie lichtgroene cijfers links in het donkergroene scherm van de Spectrum Analyzer) reikt de oranje grafiek?

c. Selecteer spoor S2. Reset **Gem**. Speel S2 af. Zorg ook nu weer dat je juist binnen die 30 seconde weer op de Stopknop drukt.
d. Ook als je op de Stopknop hebt geklikt, blijven de oranje grafieken nog zichtbaar. Klik een paar keer achter elkaar op de ronde selectieknopjes van S1 en S2 om die twee oranje grafieken goed met elkaar te kunnen vergelijken. Verloopt de stijging van de grafiek van S2 lineair? Bij welke frequentie ligt het maximum van S2? Noem deze frequentie *F*. Bij welke hoogtelijn ligt dat maximum?
e. Vul in: als je een sinustoon met frequentie *F* en amplitude *A* door dit "differentie"-filter haalt, dan is de output een sinustoon met frequentie ..... en amplitude ....
f. Kijk naar de grafiek van S2. Bij welke frequentie snijdt de grafiek de hoogtelijn die S1 bereikt? Vul in: een sinustoon van .... Hz wordt door dit filter versterkt met een factor .....
g. Het spectrum-scherm heeft hoogtelijnen 0, 1, 2 ... 9, 10. Welke cijfers zouden er bij de hoogtelijnen moeten staan als je er de amplitudekarakteristiek *A(f)* van dit "differentie"-filter mee wilt weergeven? (Kijk eerst nog even naar de definitie van de amplitudekarakteristiek *A(f)* in paragraaf 5.1.)

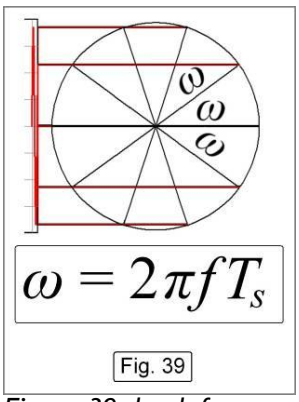
Resultaten van opdracht 51

- Straks gaan we de formule voor de amplitudekarakteristiek *A(f)* van dit filter afleiden. Die formule zal moeten bevestigen wat we in deze opdracht hebben vastgesteld, namelijk welke frequentie door dit filter het sterkst wordt doorgelaten en ook hoe groot die versterkingfactor is. Die versterkingsfactor zullen we dan ook in de formule van *A(f)* moeten kunnen terugvinden.
- Die formule voor *A(f)* zal ook de kromming van de oranje grafiek van S2 moeten beschrijven.
- Als we eenmaal over de formule voor *A(f)* beschikken kunnen we die frequentie van vraag (f) precies berekenen. Dat doen we in opdracht 52.

52. Opdracht: afleiding van hoekfrequentie $\omega = 2\pi f\ T_s$ door middel van Fasor-animatie

Open ►Preset_H5_Afleiding_2PifTs.txt en klik eerst op **Wissen Grafiek** en dan op **Start**. Rechts zie je een wijzer die ronddraait tegen de richting van de klok in. Zo'n ronddraaiende wijzer noemen we een ***fasor***. De sinus is de verticale projectie van de cirkelvormige fasorbeweging. Links is een vel ruitjespapier afgebeeld waarop de sinusgolf verschijnt die door de fasor wordt gemaakt.

a. We gaan nu de fasor langzamer laten ronddraaien en in stapjes laten lopen.
   Vul achter **`aantal beelden per seconde`** in: **`1`**.
   Vul achter **`Wissen na elke ... beelden`** in: **`50`**. Klik op **Start**.



Figuur 39: hoek fasor.

Je ziet de fasor nu stapsgewijs ronddraaien. Bij elke stap wordt een sinussample bijgemaakt op het ruitjespapier.

b. Als de fasor in precies 50 stappen 1 keer ronddraait, dan is er één periode van de sinustoon aangemaakt die uit 50 samples bestaat. Wat is dan de frequentie van de sinustoon?

c. Je kunt zelf controleren of je antwoord bij b goed is: druk op de **h-toets** om S1 af te beelden. Zoom in tot 1 Ts / pix. Meet de periode *T* van de sinustoon op S1 en bepaal *f*.
Bij elke stap legt de fasor dezelfde hoek ω af. Zie figuur 39.
In de volgende vragen ga je aantonen dat deze voor deze hoek geldt $\omega = 2\pi f T_s$ rad.

d. Stel dat je een sinustoon wilt maken met een frequentie van 1 Hz. Hoeveel stappen zal de fasor dan per omwenteling gaan zetten?

e. Hoe groot is de hoek (in rad) die de fasor bij elke stap zet, als je een toon van 1 Hz wilt maken?

f. Als je de frequentie nu *f* maal zo groot maakt, dan gaat de fasor *f* maal zo snel ronddraaien. Dat doet hij door *f* maal zo grote stappen te zetten. Hoe groot is de hoek (in rad) die de fasor bij elke stap zet, als je een toon van *f* Hz wilt maken?

g. Hoe groot is de hoek (in rad) die de fasor na *n* stappen heeft afgelegd, als je een toon van *f* Hz wilt maken?

Controleer hieronder of je de antwoorden goed hebt

Antwoord

b. $T = 50\ T_s$, dus $f = 1/T = 1/(50\ T_s) = F_s\ /\ 50$ Hz = 44100 / 50 = 882 Hz. (Herinner je dat $F_s = 1/\ T_s$).

d. F_s stappen.

e. $2\pi\ /\ F_s\ = 2\pi T_s$ rad.

f. $\omega = 2\pi\ f\ T_s$ rad.

g. $2\pi\ f\ n\ T_s\ = \omega n$ rad.

Opmerking

Als je uit de natuurkundeles de formule $\sin(2\pi\ f\ t)$ kent, dan hoef je hier alleen maar substitueren: $t = nT_s$. Je krijgt dan antwoord 52g. Want, zoals we in hoofdstuk 3 hebben gezien, discrete signalen zijn alleen gedefinieerd voor tijdstippen die een veelvoud zijn van de sampletijd T_s , dus de tijdstippen 0, T_s , $2T_s$, $3T_s$ ... nT_s ...

Definitie van de hoekfrequentie ω

De hoek ω die de fasor bij elke stap langs de eenheidscirkel aflegt noemen we de ***hoekfrequentie***. De hoekfrequentie is een lineaire functie van de frequentie:

$$\omega(f) = 2\pi f T_s \qquad (11)$$

In de nu volgende afleiding van de amplitudekarakteristiek zal blijken dat je in formules veel eenvoudiger werkt met de hoekfrequentie ω dan rechtstreeks met de frequentie f.

Amplitudekarakteristiek *A(f)* afgeleid uit differentievergelijking

We zijn nu klaar om die link te leggen tussen een differentievergelijking (= tijd-beschrijving van een filter) en de amplitudekarakteristiek *A(f)* (= frequentie-beschrijving van een filter). Dat doen we aan de hand van de differentievergelijking $y[n] = x[n] - x[n-1]$ die we in opdracht 51 bekeken.

We hoeven eigenlijk niets te doen wat we niet al eerder hebben gedaan. In opdracht 46 hebben we een filter "doorgefloten": een langzaam stijgende sinustoon stuurden we door het filter en keken hoe daardoor de amplitude werd versterkt/verzwakt. Dat doen we nu precies zo. Het enige verschil is dat we het nu op een wiskundige manier doen: we nemen een sinustoon met een willekeurige frequentie. Die noemen we f. Je mag daar in gedachten dus elk "doorfluit"-getal voor kiezen, al moet jouw keuze voor f natuurlijk wel voldoen aan de stelling van Nyquist:
kort geformuleerd: $0 \le f \le 0{,}5F_s$.

In opdracht 52 hebben we gezien dat je die f niet rechtstreeks in een formule $\sin(fn)$ kunt gebruiken; daar moet steeds iets staan als $\sin(2\pi\, f\, n\, T_s)$. Dat wordt een beetje omslachtig, daarom noteren we kort: $\sin(\omega n)$, met $\omega = 2\pi\, f\, T_s$.

We sturen nu een sinustoon $\sin(\omega n)$ door het filter. Dat doen we door $x[n] = \sin(\omega n)$ te substitueren in de differentievergelijking. We krijgen dan als output $y[n]$:

$$\begin{aligned} y[n] &= x[n] - x[n-1] \\ &= \sin(\omega n) - \sin(\omega(n-1)) \\ &= \sin(\omega n) - \sin(\omega n - \omega) \end{aligned}$$

Als we definiëren $p = \omega n$ en $q = \omega n - \omega$, dan kun je die laatste formule noteren als

$$y[n] = \sin p - \sin q$$

Als je bij wiskunde inmiddels de productformules uit de goniometrie hebt gehad, dan weet je dat we daarvoor kunnen schrijven:

$$y[n] = 2\cos\left(\frac{1}{2}(p+q)\right)\cdot\sin\left(\frac{1}{2}(p-q)\right)$$
$$y[n] = 2\cos\left(\frac{1}{2}(\omega n+\omega n-\omega)\right)\cdot\sin\left(\frac{1}{2}(\omega n-(\omega n-\omega))\right)$$

vereenvoudigd :

$$y[n] = 2\sin\left(\frac{1}{2}\omega\right)\cdot\cos\omega\left(n-\frac{1}{2}\right) \qquad (12)$$

Deze formule bevat twee factoren:
1. 2sin(½ω) en
2. cos ω(n − ½)

Laten we beginnen met die tweede term, cos ω(n − ½). Die is, in tegenstelling tot de eerste term, tijdafhankelijk. Dat kun je zien aan n, het symbool voor de discrete tijd.
Denk je dat er een verschil is te horen tussen sin(ωn) en cos ω(n − ½)? Je kunt met WaveWizard nagaan dat die precies hetzelfde klinken (afgezien van die schakelklik op tijdstip n = 0, want cos(0) = 1 en sin(0) = 0. In beide gevallen hoor je dezelfde klankkleur, dezelfde toonhoogte en hetzelfde volume.
We kunnen daaruit concluderen dat er uit het filter in elk geval een sinustoon met dezelfde toonhoogte tevoorschijn komt als die we erin stopten. (Het zou ook heel raar zijn als dat niet zo was! Want vergeet niet dat die differentievergelijking een echo-effect beschrijft; je zou raar opkijken als de echo van je stem hoger of lager zou klinken dan je stem zelf!) Dus de output heeft dezelfde "doorfluit"-frequentie als de input. Het enige wezenlijke verschil tussen input en output kan dus alleen maar zitten in de amplitude van de toon! En die wordt beschreven door die eerste term van de formule.

Die eerste term, 2sin(½ω), is puur een functie van de frequentie en verandert niet met de tijd. Het is de factor waarmee de output wordt versterkt of verzwakt, afhankelijk van ω. En dat is precies wat we sinds paragraaf 5.1 hebben verstaan onder de amplitudekarakteristiek *A(f)*!

Kortom, voor dit filter geldt:

$$A(\omega) = 2\sin\left(\frac{1}{2}\omega\right)$$

Wil je dat graag rechtstreeks uitdrukken in de frequentie f, dan krijg je:

$$A(f) = 2\sin(\pi f T_s) \qquad (13)$$

Voorbeeld

Gegeven: Een sinustoon met een frequentie van 2000 Hz en een amplitude 8000 wordt door een filter gestuurd met differentievergelijking $y[n] = x[n] - x[n-1]$.

Gevraagd: Wat is de amplitude van de sinustoon die uit het filter komt?

Oplossing: De versterkingsfactor is $A(f) = 2\sin(\pi f T_s)$. Dus $A(2000) = 2\sin(2000\pi T_s) = 2\sin(0{,}1424758) = 0{,}28399$.
De amplitude van de output is dan 8000 · 0,28399 = 2271,9.

53. Opdracht: voorbeeld narekenen met WaveWizard

a. Open ►Preset_H5_Opdracht_53.txt. Vul de ontbrekende gegevens in volgens het voorbeeld hierboven.

b. Ga naar het Scoop / Spectrum-venster. Stel de slider van **Y-fac** van Spectrum zo in dat de piek van de sinustoon op S1 reikt tot hoogtelijn 8. Tot hoe hoog verwacht je dat de piek van de sinus op S2 zal reiken? Klopt dat inderdaad?

54. Opdracht: het domein van ω

Laat zien dat uit $0 \le f \le 0{,}5F_s$ volgt dat $0 \le \omega \le \pi$ als gegeven is dat $\omega = 2\pi f T_s$.

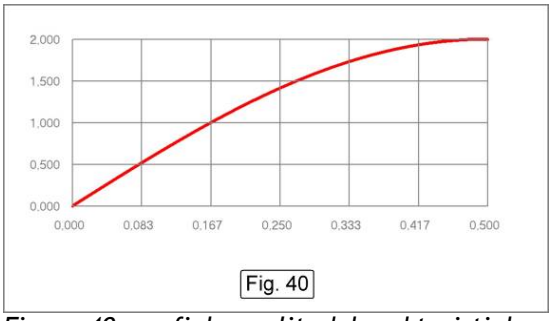
55. Opdracht: welke frequentie wordt versterkt met 1?

In opdracht 51f vonden we dat het filter een sinustoon met een frequentie van ongeveer 7000 Hz versterkt met een factor 1 (dus onversterkt, onverzwakt doorlaat). Nu we inmiddels weten dat $A(f) = 2\sin(\pi f T_s)$, kunnen we die frequentie exact uitrekenen.
Hoe? Welke uitkomst krijg je?

56. Opdracht: teken de grafiek van $A(\omega)$

Je laat WaveWizard de grafiek van $A(\omega) = 2\sin(\frac{1}{2}\omega)$ tekenen. Het domein van ω loopt van 0 tot π. De horizontale as geven we een lengte 800 beeldpunten (pixels) en berekenen dus $A(\omega)$ voor 800 waarden van ω. We definiëren een getal **`w = pi/799`** en laten de computer de functiewaarden $A(0\mathbf{w})$, $A(1\mathbf{w})$, $A(2\mathbf{w})$, $A(3\mathbf{w})$, ... $A(799\mathbf{w})$ uitrekenen. Alle uitkomsten slaan we op een decimale buffer F1.

a. Open ►Preset_H5_Grafiek_Ampkarakteristiek.txt en vul de ontbrekende code in.
HINTS: de formule wordt berekend met de functie **`Bewerk signaal`**. Die heeft **`n`** als variabele. Meestal wordt **`n`** gebruikt om de (discrete) tijd mee aan te geven, maar in dit geval is het gewoon een geheel getal dat oploopt van 0 tot 799! Gebruik dus **`n`** als de variabele in de regel **`Bewerking F1[n] = ...`**
De eerste waarde van **`n`** geef je op bij **`n0`**; de laatste bij **`n1`**.



Figuur 40: grafiek amplitudekarakteristiek.

Laat de code onder de sterretjes-regel onveranderd; die zorgt voor de lay out, zodat je figuur 40 krijgt.

b. De lay-out is zo gekozen dat de uitkomst van opdracht 55 precies op een roosterpunt ligt. Wat is de eenheid van de horizontale getallen onder de grafiek?

c. Herschrijf de differentievergelijking $y[n]=x[n]-x[n-1]$ zodanig dat je als amplitudekarakteristiek krijgt: $A(\omega) = \sin(\frac{1}{2}\omega)$.

57. Opdracht: amplitudekarakteristiek van *y[n]* = ½ *(x[n]* + *x[n−1])*

Bewijs dat je voor de amplitudekarakteristiek *A(f)* van een filter met differentievergelijking

$$y[n]=\frac{1}{2}\left(x[n]+x[n-1]\right)$$

kunt schrijven:

$$A(\omega)=\cos\left(\frac{1}{2}\omega\right)$$

Substitueer *x[n]* = sin(*ωn*) in de vergelijking en ga ook verder te werk zoals eerder in deze paragraaf gebeurde bij de afleiding van de amplitudekarakteristiek van

$$y[n]=x[n]-x[n-1].$$

N.B! Je zult een andere productformule nodig hebben:

$$\sin p+\sin q=2\sin\left(\frac{1}{2}(p+q)\right)\cdot\cos\left(\frac{1}{2}(p-q)\right).$$

Opmerking: dit filter gebruiken we in hoofdstuk 7 voor een zeer realistische imitatie van het geluid van een getokkelde snaar!

58. Opdracht: herhaald filteren

a. Stel dat je een "wit "geluid door een filter haalt met amplitudekarakteristiek *A(f)*. Nu ga je de output van dit filter weer opnieuw door datzelfde filter halen.
Wat kun je zeggen over de contour van het spectrum van het resulterende geluid?

b. En hoe ziet die contour eruit na 10 keer filteren door datzelfde filter?

c. Stel dat je een "wit "geluid door een filter haalt met amplitudekarakteristiek *A(f)*. Nu ga je de output door een filter halen

met amplitudekarakteristiek *B(f)*.Wat kun je zeggen over de contour van het spectrum van het resulterende geluid?

d. Als je bij vraag (c) eerst had gefilterd met dat *B(f)* filter en daarna met het *A(f)* filter, wat had je dan voor resultaat gekregen?

e. In de wiskunde noemen ze bewerkingen zoals vermenigvuldigen en optellen ***commutatief*** ("verwisselbaar"). Je mag de operanden van plaats verwisselen, bijvoorbeeld 2 x 7 = 7 x 2 en ook 2 + 7 = 7 + 2. Een bewerking als aftrekken is niet commutatief, want 2 – 7 is niet gelijk aan 7 – 2. En hoe zit het met filters?

Opmerking: In hoofdstuk 7 zullen we een geluidsfragment honderden of zelfs duizenden malen door dat filter van opdracht 58 gaan halen!

Amplitudekarakteristiek van willekeurig filter

Elk filter kun je beschrijven in termen van een differentievergelijking en elk filter heeft een amplitudekarakteristiek die je uit de corresponderende differentievergelijking kunt afleiden. Bijvoorbeeld ook die Butterworth filters die we bekeken. Maar voor de afleiding van zulke krachtige filters is wel wat wiskunde uit het hoger onderwijs nodig. Je weet inmiddels al dat je dat vaak helemaal niet hoeft te doen: om een indruk te krijgen van *A(f)*, stuur je er gewoon een wit geluid doorheen!

Samenvatting

- In dit hoofdstuk hebben we de krachtigste en meest veelzijdige signaalbewerking besproken, namelijk het filter. De klankveranderende eigenschap van een filter kunnen we weergeven in een afbeelding, de ***amplitudekarakteristiek A(f)***, een ***continue*** functie van de frequentie.
- Haal je een "wit" geluid door een filter, dan is het spectrum van de output een opgeschaalde versie van de amplitudekarakteristiek.
- Twee veelvoorkomende filtertypes hebben we bestudeerd, namelijk het ***laagdoorlaat-*** en het ***banddoorlaatfilter***.
- Door gebruik te maken van de ***hoekfrequentie*** $\omega = 2\pi f T_s$ worden berekeningen veel eenvoudiger.
- We hebben uit de tijdbeschrijving van een filter (differentievergelijking) de frequentiebeschrijving (amplitudekarakteristiek) afgeleid. Dit deden we door het filter "wiskundig door te fluiten", dat wil zeggen we substitueerden $\sin(\omega n)$ in de differentievergelijking en vonden zo een expressie voor $A(\omega)$.

Woordenlijst

afsnijdfrequentie	§3	banddoorlaatfilter	§3
all pass filter	§1	centrumfrequentie	§3
amplitudekarakteristiek *A(f)*	§1	commutatief	§4
bandbreedte	§3	doorfluiten	§2

fasor §4
flanksteilheid §3
hoekfrequentie ω §4
laagdoorlaatfilter §3
lange-termijnspectrum §2
pulstoon §2
witte ruis §2

6 Sequencing en MIDI

6.1 Inleiding

Het wordt zo langzamerhand tijd dat we eens wat meer muziekachtige geluiden gaan maken in plaats van alleen maar wat losse laboratoriumtoontjes te mixen of door een filter te halen!
Daarom gaan we bekijken hoe we een MIDI-bestand kunnen gebruiken om de toongenerator mee aan te sturen. Daarbij hebben we een extra commando nodig waarmee we een willekeurig aantal instructies in een lus kunnen opnemen. Deze nieuwe, universele lusinstructie kom je in vrijwel alle programmeertalen tegen (ook op je GRM), en heet **`FOR...TO...NEXT`**.
In paragraaf 6.2 gaan we daar mee aan de slag. We laten dan gelijk een paar leuke, nogal onalledaagse muzikale toepassingsvoorbeelden zien.

6.2 Een reeks tonen

Als je het puur technisch bekijkt, is een melodie niets anders dan een reeks tonen en een muziekstuk niets anders dan een aantal gelijktijdig klinkende melodieën en ritmen.
We gaan nu eerst bekijken hoe je met WaveWizard een reeks tonen maakt. Tot nu toe heb je met de toongenerator tonen opgewekt die steeds begonnen op tijdstip nul. Dat deed je door achter **`Spoor: S?[?]`** bijv. in te vullen: **`S1[0]`**. De waarde tussen **`[ ]`** is, zoals je weet, het sample-rangnummer. Wil je een toon 2 seconde na nul laten beginnen, dan vul je daar in: **`S1[88200]`** of **`S1[2*Fs]`**.

59. Opdracht: reeks tonen
Maak met de toongenerator vijf sinustonen die 1 seconde na elkaar beginnen, de eerste op tijdstip nul. De frequenties van de tonen zijn: 100, 200, 300, 400, 500 Hz;
de amplitudes zijn allemaal 2000,
de eerste toon duurt 5 sec, de tweede 4 sec, de derde 3 sec etc.
Open ►Preset_H6_Vijf_Tonen.txt, vul de ontbrekende gegevens in en klik op **Start**.
Welke eigenschap van de toongenerator wordt hiermee gedemonstreerd? (zie opdracht 34).

De **`FOR...TO...NEXT`** instructie

In opdracht 59 heb je een aantal tonen gemaakt door verschillende kopieën te maken van 1 instructie. Dat werkt prima, zolang het gaat om

slechts een paar tonen. Wil je langere reeksen opwekken, dan wordt het gauw ondoenlijk. Daarom hebben we een veel algemenere lus-instructie nodig dan die we tot nu toe hebben gebruikt.
We zagen bijvoorbeeld dat de instructie **`Bewerk signaal`** een lus uitvoert door steeds opnieuw de formule die je neerschrijft achter **`Bewerking`** te evalueren, net zo lang tot alle waarden *n* van **`n0`** tot en met **`n1`** zijn berekend. Daarom is **`Bewerk signaal`** een nogal specialistische lus: er staat in de lus altijd slechts één instructie en dat is die berekening van de formule. Je kunt per lus niet meer doen dan dat.
Maar het komt heel vaak voor dat je een lus-constructie nodig hebt waarin je elke WaveWizard-functie, zoals toongenerator kunt laten herhalen en waarin je bovendien per lus een onbeperkt aantal instructies achter elkaar kunt uitvoeren. Zo'n type lus tref je aan in elke programmeertaal, ook in WaveWizard. Het is de zg. **`FOR...TO...NEXT`** instructie.
Hieronder volgen twee voorbeelden. Doe nog even niks met WaveWizard; lees alleen de tekst goed door!

Voorbeeld 1

```
FOR n = 1 TO 5
  Print n
NEXT n
```

Als je deze code invoert en op start klikt, verschijnt in Memo de tekst:

```
Uitvoer commando's gestart...

  n = 1
  n = 2
  n = 3
  n = 4
  n = 5

Commando's uitgevoerd. OK
```

In de eerste regel van de lus vertel je de computer dat je alle instructies vanaf de regel eronder tot aan de regel die met **`NEXT n`** begint, een aantal malen wilt herhalen. Net zoals bij **`Bewerk signaal`** heb je ook hier een start- en eindwaarde. Bij **`Bewerk signaal`** heten die respectievelijk **`n0`** en **`n1`**; hier geef je ze aan door de getallen links en rechts van **`TO`**. Die getallen mogen, weer net zoals bij **`n0`** en **`n1`**, ook formules zijn en er mogen ook namen van variabelen in voorkomen die je eerst zelf hebt gedeclareerd.
Bij de eerste lus krijgt **`n`** de startwaarde, in dit geval 1. Met die waarde doorloopt de lus alle instructies - in dit geval is dat er maar één, namelijk **`Print n`**. Zodra de instructie **`NEXT n`** is bereikt, kijkt de computer eerst of alle waarden van *n* al zijn gepasseerd:
zo nee, dan wordt *n* met 1 verhoogd en wordt de lus herhaald;

zo ja, dan gaat de computer verder met de instructie die onder de regel `NEXT n` staat.

WaveWizard helpt je de lus goed te onderscheiden door alle instructies die binnen de lus worden uitgevoerd twee spaties te laten inspringen. Je hoeft dat niet zelf te doen; het gebeurt automatisch zodra je op **Start** klikt.

Pas op met de variabele *n*

Anders dan in andere programmeertalen heeft de variabele *n* bij WaveWizard een vaste betekenis.

In alle generatoren wordt `n` gebruikt om de discrete tijd mee aan te geven, precies zoals je dat al deed in `Bewerk signaal`.

Maak er daarom een gewoonte van om *n* niet te gebruiken als de variabele van een `FOR...TO...NEXT` lus!

Waarom eigenlijk een *aparte* instructie `Bewerk signaal`?

Wat je met de instructie `Bewerk signaal` kunt, kun je ook met `FOR...TO...NEXT`.
In paragraaf 5.4 kwam je tegen:

```
Bewerk signaal
  n0                                  0
  n1                               10*Fs
  Bewerking  S2[n] = (0,1*Ts*n) * S1[n]
```

Precies hetzelfde resultaat krijg je met:

```
FOR n 0 TO 10 * Fs
  S2[n] = (0,1*Ts*n) * S1[n]
NEXT n
```

Net zoals je bij `Bewerk signaal`, de waarden van `n0` en `n1` kunt aangeven door formules, kun je dat ook doen bij `FOR...TO...NEXT`; in dit voorbeeld vind je zowel vóór als achter `TO` een formule.
De reden dat WaveWizard beschikt over een aparte functie `Bewerk signaal`, als een "gespecialiseerde" `FOR...TO...NEXT` lus, is dat de rekensnelheid ervan aanzienlijk groter is.

Voorbeeld 2

Door gebruik te maken van `FOR...TO...NEXT` komt opdracht 59 er als volgt uit te zien:

```
Declareer Toonnummer           ! declaraties moeten bovenaan staan

Wis buffers
  Wis buffer (F1...F32, S1, S2, S3)    S1

FOR Toonnummer = 1 TO 5
  Print Toonnummer

  Toongenerator
    Frequentie (Hz of TOETS)   100 * Toonnummer
    Amplitude                  2000
    Golfvorm                   sinus
    Spoor: S?[?]               S1[(Toonnummer-1) * Fs]
    Duur                       (6 - Toonnummer) * Fs
    Aantal harmonischen        1

NEXT Toonnummer
```

Nu kun je in principe net zoveel tonen maken als je wilt; de code wordt er niet langer door.
Alle variabelen die je zelf declareert, kun je gebruiken om mee te rekenen. In dit voorbeeld zie je dat de toongenerator de naam **"Toonnummer"** (of elke andere naam die je declareert) in alle relevante invoervelden herkent en verwerkt.

In plaats van de code een aantal maal onder elkaar te zetten, zoals je deed in de opdracht, heb je nu maar één toongenerator-tekstblok. Dat kan natuurlijk alleen, als er structuur te ontdekken valt in de instellingen van de afzonderlijke tonen. Stel bijvoorbeeld dat de frequentiereeks had moeten zijn: 100, 122, 317, 329 en 408 Hz, dan was er geen structuur in te vinden geweest die je kunt uitdrukken in een formule waarin het nummer van de toon de onafhankelijke variabele is. Maar bij de reeks 100, 200, 300, 400, 500 is dat wel het geval. Daarvoor kun je immers schrijven:

$100\,k$ voor $k = 1$ tot en met 5

Datzelfde geldt voor de begintijdstippen van de tonen; dat zijn allemaal veelvouden van F_s.
Ook de Duur van elke toon hangt af van het toonnummer.

Random getallen maken

In opdracht 60 krijg je te maken met de instructie `random(a; b)`. Daarmee kun je een ***random*** geheel getal g opwekken waarvoor geldt: $a \leq g \leq b$.

Bijvoorbeeld, als je de computer 10 keer met een dobbelsteen wilt laten gooien, dan schrijf je:

```
FOR k = 1 TO 10
  Print random(1; 6)
NEXT k
```

Wil je de worpen bewaren in buffer F1, dan schrijf je:

```
FOR k = 1 TO 10
  F1[k] = random(1; 6)
NEXT k
```

Random getallen zijn voor sound design onmisbaar en hebben talrijke toepassingen, uiteenlopend van het opwekken van ruis tot computer-compositie (zie paragraaf 1.5). In opdracht 60 een paar voorbeeldjes.

60. Opdracht: toonreeks met random frequenties

Open ►Preset_H6_RandomToonreeks.txt en klik op **Start**.

a. Verklaar waarom elke noot een andere toonhoogte heeft.
b. Wat is de hoogste frequentie die een toon kan krijgen?
c. Wil dat ook zeggen dat je een toon zult aantreffen die inderdaad die hoogste frequentie heeft?
d. Ga naar het Scoop/Spectrum-venster. Activeer **Max**, speel S1 af en kijk naar het spectrum. Wat moet je in de code veranderen om tonen te krijgen in het frequentiegebied van 1000 tot 2000 Hz?
e. Bij de eerste lus is de waarde *k* gelijk aan nul. Als je de lus graag wilt laten beginnen met 1, en toch precies hetzelfde resultaat wilt krijgen, wat moet je dan in de code wijzigen?
f. Maak de duur van de tonen 3 keer zo lang.
g. Zorg dat de tijd tussen elke twee tonen 2 keer zo lang wordt.
h. Maak het aantal tonen 2 keer zo klein.

61. Opdracht: percussieve tonen

Open ►Preset_H6_Percussie.txt en klik op **Start** en speel het geluid af via het Scoop / Spectrum-venster.

a. Is er een verband tussen het volume van een ruistoon en de toonhoogte ervan? Zo ja welk?
b. Is er een verband tussen bandbreedte van een ruistoon en de toonhoogte ervan? Zo ja welk?

Je maakt in deze preset kennis met een nieuwe functie: de ***Feedbackgenerator***, die in hoofdstuk 7 een belangrijke rol gaat spelen. De feedbackgenerator heeft een ingebouwde ADSR-contour en kan ook ruis aanmaken. Dat is de enige reden waarom we 'm hier gebruiken: ruis opwekken en gelijk een volumecontour geven. Die ruis wordt vervolgens door het Butterworth Banddoorlaatfilter gehaald, dat je in hoofdstuk 5 leerde kennen.

c. Er wordt een ogenschijnlijk vreemde procedure gevolgd: de ruis wordt niet direct op S1 gezet, maar op **`S3[0]`**, dus steeds vooraan op S3 bij sample 0. Het filter bewerkt dan het signaal van S3 en zet de output op **`S1[BeginTijdstip]`**. Daarna worden op S3 de eerste **`Toonduur + 2*Fs`** samples op nul gezet (dus uitgewist). Waarom zo ingewikkeld? Waarom niet direct het geluid op S1 zetten en dan daar ter plekke filteren? (HINT: zie opdracht 34).
d. Wat veroorzaakt dat galm-achtige effect? Hoe kun je die galm korter maken?
e. Waarom bewerkt het filter niet gewoon **`Toonduur`** samples? Waarom rekent deze **`2*Fs`** samples extra uit? (HINT: kijk wat er gebeurt als je in de code van **`Butterworth`** en van **`Bewerk signaal`** simpelweg die term **`+ 2*Fs`** weglaat!)
f. Wat moet je in het filter veranderen om de ruistonen meer sinusvormig te maken, zodat je een soort kwetterende vogeltjes krijgt? (HINT: je weet dat een sinus er in het spectrum uitziet als een (oneindig) smal frequentiegebiedje!)

Sequencing

Een ***sequencer*** is in de elektronische muziek een automaat die een synthesizer "bespeelt" door er reeksen toongegevens naar toe te sturen, precies zoals we in de laatste twee opdrachten deden door middel van **`FOR...TO...NEXT`**.
Zo'n reeks tonen kan ook elk willekeurig muziekstuk zijn en niet alleen "dobbelsteenmuziek". De meest bekende en universele vorm van sequencing is het MIDI-bestand dat alle gegevens van een muziekstuk codeert, zelfs de manier waarop het door de muzikant wordt gespeeld (hard-zacht aanslag, vertragingen etc.) Hoe MIDI werkt, zien we in paragraaf 6.4. Eerst kijken we nog even naar de manier waarop muziekinstrumenten meestal worden gestemd, dat wil zeggen volgens welk "recept" de frequenties van de tonen van de toonladder worden bepaald.

6.3 Stemming

In de muziek wordt meestal gebruik gemaakt van de zogenaamde ***evenredig zwevende*** stemming om de frequenties van onder andere de toetsen op de piano te vinden.
Die stemming wordt gedefinieerd door twee eenvoudige regels:

1. De frequentieverhouding van elke twee opeenvolgende toetsen is constant.
2. In een octaaf passen 12 toetsen.

Je kunt de octaafafstand van 12 toetsen op de piano gemakkelijk herkennen aan het patroon van witte en zwarte toetsen. Zie bijvoorbeeld figuur 13.

Noemen we de frequenties van twee opeenvolgende toetsen f_0 en f_1, dan kunnen we regel (1) als volgt in formulevorm uitdrukken:

$$f_1 : f_0 = c$$

ofwel

$$f_1 = cf_0 \qquad (14)$$

Dit laatste laat zien dat je de frequentie f_1 kunt vinden door die van f_0 te vermenigvuldigen met *c*. De vraag is dan natuurlijk: hoe kom ik aan *c*?
Om het antwoord vinden passen we eerst regel (1) toe op alle twaalf tonen van het octaaf:

$$f_2 = cf_1 = c^2 f_0$$
$$f_3 = cf_2 = c^3 f_0$$
$$\ldots$$
$$f_{12} = cf_{11} = c^{12} f_0$$

Volgens regel (2) moet f_{12} een octaaf hoger zijn dan f_0 en, zoals je weet (paragraaf 2.2), een octaaf is een frequentieverhouding van 1 : 2. Dus je kunt aanvullen:

$$f_{12} = c^{12} f_0 = 2 f_0$$

Deel je beide kanten van het rechtse = teken door f_0, dan krijg je

$$c^{12} = 2$$

Nemen we nu de twaalfde-machtswortel:

$$c = 2^{1/12} = 1.0594631 \quad (15)$$

Als je dus de frequentie van slechts één toon weet, dan kun je de frequenties van alle andere tonen uitrekenen.
In het orkest wordt altijd een vaste toonhoogte gebruikt om de instrumenten te stemmen; dat is de toon *a'* ("a - éengestreept") met een

frequentie van 440 Hz. Kiezen we die frequentie als f_0 dan vinden we voor de *n-de* toets rechts daarvan de frequentie f_n:

$$
\begin{aligned}
f_n &= f_0 \cdot c^n \\
&= 440 \cdot \left(2^{1/12}\right)^n \\
&= 440 \cdot 2^{n/12}
\end{aligned}
$$

Voorbeelden

Het octaaf boven toets *a'* (12 toetsafstanden naar rechts) is: $f_{12} = 440 \cdot (2^{12/12}) = 880$ Hz.

Het octaaf onder toets *a'* (12 toetsafstanden naar links) is $f_{-12} = 440 \cdot (2^{-12/12}) = 440 \cdot (2^{-1}) = 220$ Hz.

62. Opdracht: bereken de frequenties van de evenredig zwevende stemming

Open ►Preset_Stemming.txt en vul een formule in waarmee de frequenties van alle 12 tonen van het octaaf worden berekend, beginnend bij de stemtoon *a'* = 440 Hz.

6.4 Toongenerator aansturen met een MIDI-bestand

Een muzieknoot kun je representeren door een groepje van 5 getallen die we ***nootparameters*** noemen. Met vier daarvan heb je bij de toongenerator al gewerkt:

`Frequentie (Hz of TOETS)`	toonhoogte
`Amplitude`	volume
`Spoor: S?[?]`	begintijdstip
`Duur`	nootlengte

Bij MIDI komt daar nog een vijfde nootparameter bij: het zogeheten ***kanaalnummer***. Dat heb je nodig omdat er in een muziekstuk vaak meerdere instrumenten of zangstemmen gelijktijdig klinken. Elk instrument en elke stem krijgt dan een eigen kanaalnummer, net zoals elk muziekinstrument in de partituur z'n eigen notenbalk heeft. Zonder kanaalnummers zouden alle samenklinkende melodieën alleen maar dezelfde klankkleur kunnen hebben, omdat ze op geen enkele manier van elkaar onderscheiden zouden kunnen worden.

Meten met twee maten...

Je zult zo dadelijk aan de hand van voorbeelden zien dat het in de computermuziek vaak erg handig is om te werken met twee verschillende

eenheden voor de toonhoogte. Datzelfde geldt voor de tijd. Daarom moet je die eenheden in WaveWizard ook kunnen aangeven.

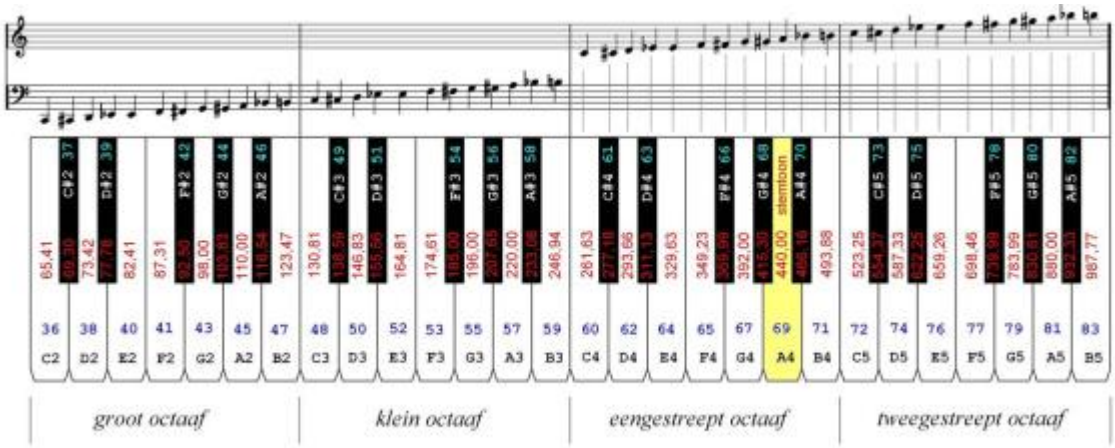
Toonhoogte

Tot nu toe hebben we de toonhoogte altijd uitgedrukt in Hertz (Hz). Maar muzikanten doen dat slechts in één enkel geval, namelijk als het over de frequentie van de stemtoon (440 Hz) gaat. (De stemtoon is de toon die je, voorafgaand aan een concert, hoort spelen door de hobo en waar alle andere orkestleden hun instrument mee stemmen).
Omdat in de muziek (meestal) alleen gebruik wordt gemaakt van de tonen van de toonladder (waarvan we in paragraaf 6.3 de frequenties hebben bepaald), is het veel simpeler om die tonen elk een naam te geven (a, ais, b, c, cis, d, dis, e, f, fis, g, gis). Ook de octaven zelf hebben namen, zie figuur 41.
In MIDI krijgen alle toetsen van het toetsenbord een nummer, beginnend bij nul en eindigend bij 127. Reken na dat je dus binnen MIDI de beschikking hebt over ruim tien en een half octaaf (laat ze ook alle 128 eens horen via WaveWizard!) Ook die toetsnummers en de corresponderende frequenties in Hz zie je in figuur 41.

In MIDI-bestanden wordt de toonhoogte aangegeven door een toetsnummer.

Daarom kun je in WaveWizard de frequentie niet alleen opgeven in Hz, maar ook in de "eenheid" TOETS. Bijvoorbeeld bij de toongenerator heb je gezien dat er bij frequentie staat: `Frequentie (Hz of TOETS).`

Als je achter `Frequentie` het woord `TOETS` typt, gaat WaveWizard ervan uit dat je een MIDI-toetsnummer bedoelt en rekent dat om naar de corresponderende frequentie in Hz.



Figuur 41: octaafnamen, toetsnummers en frequentie.

Voorbeeld

Als je bij de toongenerator achter `Frequentie` intypt: `64 TOETS`, berekent WaveWizard de frequentie van MIDI-toets 64 en die bedraagt 329,63 Hz. (zie figuur 41). Laat je dat woord `TOETS` weg, dan krijg je gewoon een toon met een frequentie van 64 Hz.

Tijd

In de muziek wordt de tijd niet gemeten in seconden, maar in ***tellen***. Hoe lang een tel duurt, hangt af van het ***tempo*** dat je kiest. Net zoals periode en frequentie zijn ook tel en tempo elkaars inversen. Het tempo wordt in de muziek niet uitgedrukt in seconden maar in het aantal tellen per minuut. Vaak kom je de afkorting ***BPM*** tegen: Beats Per Minute.
Bijvoorbeeld voor een marstempo geldt gewoonlijk 120 BPM (dat komt dus neer op "2 Hz") (als je marcheert zet je ongeveer 120 stappen per minuut, dat is twee stappen per seconde).
Vandaar dat je in WaveWizard de tijd ook kunt uitdrukken in de eenheid ***TEL***. Hoe lang die duurt hangt af van het tempo dat je kiest.

Als je achter een tijd-instelling het woord `TEL` typt, gaat WaveWizard ervan uit dat je de muziekeenheid 'tel' bedoelt en rekent die om naar het corresponderende aantal sample-tijden T_s .

Voorbeeld

Als je in het commandovenster uitsluitend een instructie voor de toongenerator hebt staan, en achter `Duur` hebt ingetypt: `20 TEL`, en je klikt op **Start**, dan krijg je een foutmelding! Want WaveWizard weet het tempo niet. Daarom moet je eerst (dus boven de instructie van de toongenerator) het tempo invoeren. Dat doe je met de instructie:
`Tempo:`
bijvoorbeeld `Tempo: 120`
In dat geval gaat WaveWizard een toon van 10 seconde maken, want tempo 120 betekent 120 tellen per minuut, dus 2 tellen per seconde, zodat 20 tellen 10 seconde duren.

Wat doet WaveWizard als je een MIDI-bestand opent?

Als je een MIDI-bestand opent, gebeurt er het volgende:

1. WaveWizard gaat kijken of het MIDI bestand zo ongeveer "deugt". Op internet tref je honderdduizenden MIDI-bestanden en daar zitten er ook tussen die fouten bevatten. WaveWizard probeert die te repareren; soms lukt dat wel, soms niet en dan krijg je een melding. WaveWizard gaat vervolgens het bestand "uitpakken" en de nootparameters in de eerste zeven F-buffers plaatsen. Zeven? Er zijn toch maar vijf nootparameters? Jawel, maar (a) de toonhoogte wordt in twee eenheden weergegeven, namelijk zowel in Hz als in TOETS, en (b) ook van het volume van de toon bestaan twee versies, zie hieronder bij `Amp` en `Velocity`.

2. In het Commandovenster wordt een standaardcode aangemaakt.
   De namen van de eerste zeven F-buffers worden aangepast zodat je je niet steeds hoeft af te vragen in welke buffer de toetsnummers of in welke buffer de toonduren ook al weer stonden. Hiervoor beschikt

WaveWizard over een apart commando dat je ook zelf kunt gebruiken als je een F-buffer graag een naam wilt geven die de inhoud van de buffer beter aanduidt. Dit commando heet **`Naam Buffer`**.

Vanaf dat moment heten de buffers:

`F1` heet `Begin`	begintijdstippen van de noten.
`F2` heet `Duur`	duur van de toon (nootlengte).
`F3` heet `Toets`	toonhoogte van de toon, uitgedrukt als toetsnummer (0...127). Als je gebruik maakt van deze buffer, dan moet je er in de code altijd TOETS achter schrijven! (WaveWizard doet dat automatisch voor je, zie verderop in deze paragraaf.)
`F4` heet `Freq`	toonhoogte van de toon, uitgedrukt in Hz. Dit is niets anders dan een omrekening van buffer F3! Als je gebruik wilt maken van F4, dan mag je er Hz achter zetten, maar dat hoeft niet per se.
`F5` heet `Amp`	amplitude van de toon is niets anders dan een opschaling van de "Key-Velocity", zie F6.
`F6` heet `Velocity`	***"Key-Velocity"*** ofwel de kracht waarmee de toets wordt aangeslagen. Hoe groter die kracht, des te groter het volume. De key-velocity wordt gerepresenteerd door een getal in het bereik 0 ....127. Hierin correspondeert 0 met onhoorbaar en 127 met maximale sterkte.
`F7` heet `Kanaal`	kanaalnummer.

Je kunt achter de nieuwe naam gewoon weer een rangnummer tussen haken plaatsen, bijv. **`Duur[7]`** of **`Toets[k]`**.
Ook vind je bovenaan de code de instelling van het tempo, zoals die door het MIDI-bestand zelf wordt opgegeven.

Vanwaar dat onderscheid tussen `Toets` en `Freq`?

Het komt in de muziek vaak voor dat je een stuk iets hoger of lager wilt laten klinken (***"transponeren"***). Stel bijvoorbeeld dat je alle noten drie toonsafstanden hoger wilt horen. Dan kun je twee dingen doen:

1. vermenigvuldig alle frequenties met $2^{3/12}$. Bijv. **`Freq[k] * 2 ^ (3 / 12)`**
2. tel bij alle toetsen *3* op . Bijv. **`Toets[k] + 3 TOETS`**.

Aan jou de keuze...
Als je een "akoestische" bewerking doet, zul je vaker geneigd zijn om **`Freq`** te kiezen, terwijl een "muzikale" bewerking vaak eenvoudiger is via **`Toets`**.

Meerdere kanalen

In het MIDI-bestand van een compositie waarin verschillende instrumenten voorkomen, heeft elk instrument z'n eigen kanaalnummer. Het kan

bijvoorbeeld zijn dat de noten 0 t/m 173 het kanaalnummer 1 hebben en de noten 174 t/m 219 het kanaalnummer 2. Voor elk kanaalnummer maakt WaveWizard een **`FOR...TO...NEXT`** lus aan en stelt daarbij gelijk de juiste start- en eindwaarden in. Binnen elke lus staat de code voor de toongenerator, waarin TOETS en TEL al op de juiste plaatsen worden aangegeven.

WaveWizard - MIDI samengevat

Als je een MIDI-bestand hebt geopend, verricht WaveWizard alle handelingen die nodig zijn om direct op **Start** te kunnen klikken en het audio-signaal te laten berekenen. Het enige dat jij moet doen is die standaardcode zo aanpassen dat je de klank krijgt die jij graag wilt horen. Doe je geen aanpassingen dan hoor je een (gortdroog) "standaardklankje".

63. Opdracht: Bach
Ga met de muis naar Bestand → Openen Midi bestand (.MID).
Open ►Bach_Kleine_Preludien_und_Fughetten_No_1.mid
In het commandovenster verschijnen automatisch codes waarmee je het stuk direct kunt doorrekenen, maar wij maken daar geen gebruik van.
Open ►Preset_H6_Bach.txt en klik op **Start**.

a. Als je naar **Lijsten** gaat, dan zie je dat de namen van de buffers nu inderdaad zijn aangepast. In figuur 42 zijn de gegevens van de eerste 12 noten van dit MIDI-bestand afgebeeld. Een horizontale rij beeldt alle nootparameters van één noot af.
   Hint: je hoeft niet alle noten te laten uitrekenen (dat duurt vrij lang); na zo'n 10 seconde kun je al op **Play** drukken terwijl WaveWizard gewoon doorrekent aan de volgende noten. Het kan zijn dat het afspelen sneller gaat dan het uitrekenen; dan hoor je vanaf een zeker moment gehakkel of helemaal niets meer. Druk dan op de **s-toets** en klik vervolgens weer op **Play**.
b. In figuur 42 kun je aflezen dat de eerste toon een toonhoogte heeft die correspondeert met toets 60. Reken voor dat **`Toets[0] TOETS = Freq[0] Hz =`** 261,6256 Hz is.
c. Achter `Frequentie (Hz of TOETS)` staat nu: `Toets[k] TOETS`. Wat moet je daar tussenvoegen om het stuk een octaaf hoger of lager te laten klinken? (HINT: hoeveel toetsen bevat een octaaf?)
d. Door ergens in de commando-tekst één symbool te vervangen door **`(0,1*k+1)`** wordt elke toon ietsje zachter dan de vorige. Waar? Kun je uitleggen waarom dit zo is?
e. Door ergens in de commando-tekst in te voegen **`*0,01*k`**, neemt bij elke noot het tempo ietsje af. Waar moet je die invoeging plaatsen?
f. Door ergens in de commando-tekst toe te voegen: **`/(0,1*k+1)`** wordt elke toon weer ietsje korter. Waar?
g. Selecteer met de muis het code-blok van Toongenerator ADSR (dus 12 regels tekst). Maak er een kopie van (toets Ctrl C) en plak de kopie

direct eronder (toets Ctrl V). Breng enkele veranderingen aan in het nieuwe tekstblok zodat het stuk nu ook op S2 komt en wel 19 toonsafstanden hoger! Hogere tonen klinken meestal wat korter dan lage, daarom maak je de Duur op S2 vier maal zo kort. Hetzelfde doe je met de Release tijd. Klik op **Start**.
Hoe verhouden zich de frequenties (we hebben het nu over Hz en niet over Toets!) van de tonen op S1 tot de corresponderende op S2? (HINT:

```
FOR k = 0 TO 507
  Print k
  Toongenerator
    Frequentie (Hz of TOETS)  Toets[k] TOETS
    Amplitude                 Amp[k] / 2
    Golfvorm                  zaag
    Spoor: S?[?]              S2[Begin[k]]  TEL
    Duur                      Duur[k]  TEL
    Aantal harmonischen       15
NEXT k
```

	Begin	Duur	Toets	Freq	Amp	Velocity	Kanaa
noot k = 0	0,2833	0,9917	60,0000	261,6256	4183,0000	89,0000	1,000
1	0,5333	0,5000	72,0000	523,2511	3290,0000	70,0000	1,000
2	0,7833	0,5000	75,0000	622,2540	3102,0000	66,0000	1,000
3	1,0333	0,8333	79,0000	783,9909	3807,0000	81,0000	1,000
4	1,2833	0,5000	75,0000	622,2540	3337,0000	71,0000	1,000
5	1,5333	0,5000	72,0000	523,2511	3666,0000	78,0000	1,000
6	1,7833	0,5000	75,0000	622,2540	4089,0000	87,0000	1,000
7	2,0333	0,5000	72,0000	523,2511	3854,0000	82,0000	1,000
8	2,2833	0,5000	67,0000	391,9954	4089,0000	87,0000	1,000
9	2,5333	0,5000	72,0000	523,2511	4089,0000	87,0000	1,000
10	2,7833	0,5000	63,0000	311,1270	2820,0000	60,0000	1,000
11	3,0333	0,5000	72,0000	523,2511	3948,0000	84,0000	1,000
12	3,2833	0,7417	60,0000	261,6256	3854,0000	82,0000	1,000

Fig. 42

Figuur 42: gegevens eerste 12 noten MIDI-bestand.

zet **Zoom** op 2 pix / Ts en selecteer door middel van een linker en rechter muisklik vijf periodes van S1 (zodat ze rood worden afgebeeld; kijk vervolgens hoeveel rode periodes je krijgt op S2). Is die frequentieverhouding harmonisch?

Samenvatting

Door middel van de instructie **`FOR...TO...NEXT`** zijn we in staat om met een toongenerator een lange reeks tonen met verschillende toonhoogten, amplituden, begintijdstippen en toonduren te produceren. Dat kunnen random reeksen zijn, maar ook MIDI-bestanden kunnen we gebruiken om toongeneratoren en filters mee aan te sturen.
In feite hebben we hiermee de beschikking over een volwaardige "klassieke" synthesizer die gekoppeld is aan een Sequencer. Die klassieke synthesizer bestaat in elk geval uit drie basisonderdelen: (1) toongenerator, (2) contour-generator ADSR en (3) filters.
Met alle drie hebben we inmiddels ervaring. Maar wat nog ontbreekt zijn methoden waarmee we meer realistisch klinkende tonen kunnen maken, dat wil zeggen tonen die voldoen aan de klankwet. Daarover meer in hoofdstuk 7.

Woordenlijst

BPM §4
evenredig zwevende stemming §3
feedbackgenerator §2
FOR...TO...NEXT §2
kanaalnummer §4
key-velocity §4
MIDI §4
nootparameters §4
random §2
sequencer §2
TEL §4
tellen §4
tempo §4
TOETS §4
transponeren §4

7 Tonen waarvan de klankkleur verandert

7.1 Inleiding

Er zijn vele verschillende methodes waarmee we de klankkleur van een toon dynamisch kunnen veranderen en waarmee we geluiden kunnen maken die voldoen aan de klankwet. In dit hoofdstuk zullen we er twee bespreken:

1. ***De snaarplukvergelijking***
   De ***snaarplukvergelijking*** is een zeer efficiënte methode om het geluid van een getokkelde snaar realistisch na te bootsen. Je hoorde er een voorbeeld van in opdracht 6 (►H1_Gedempt_en_GedemptDoffer.mp3). De snaarplukvergelijking wordt vaak beschouwd als het eenvoudigste natuurkundige model (zie paragraaf 1.3) van een snaar.
   We gaan met snaarpluk eerst het geluid van een klavecimbel imiteren. Dan voegen we een Butterworth Laagdoorlaat filter toe om daar een soort pianoachtige klank van te maken.

2. ***Tijdvariante filters***
   Een ***tijdvariant filter*** is een filter waarvan de amplitudekarakteristiek *A(f)* voortdurend verandert. In opdracht 5b heb je het effect van een tijdveranderlijk filter al beluisterd in ►H1_Klankwet_demo.mp3. In dit hoofdstuk leer je hoe je de Laag- en Banddoorlaatfilters uit hoofdstuk 5 op een tijdvariante manier kunt instellen.
   Dat de klankwet niet alleen opgaat voor muziekinstrumenten, maar voor een veel grotere klasse van trillingsverschijnselen wordt mooi geïllustreerd door het geluid van huilende wind. We gaan proberen om huilende wind te maken met behulp van een bijzonder filter dat niet alleen tijdvariant, maar ook ***adaptief*** is: het stuurt automatisch de afsnijdfrequentie van het filter voortdurend bij op basis van het amplitudeverloop van het inputsignaal. Dus wat dit filter doet, kun je kort omschrijven met: "hoe harder hoe helderder". Daarom zou je het met recht een "klankwet-filter" mogen noemen.

7.2 De snaarplukvergelijking

We beginnen met een inleidende opdracht.

64. Opdracht: echo's met de Feedbackgenerator
Open ►Brekend_glas.wav. Open
►Preset_H7_HogeBrug_Feedbackgenerator.txt. en klik op **Start**.

a. Luister en kijk naar S2 en S3. Bespeur je enig verschil?
Als je naar de code in het commandovenster kijkt, dan zie je dat er twee verschillende methodes werden gebruikt met hetzelfde effect. De methode van S3 heb je zelf al gebruikt in opdracht 42; je ziet daar de recursieve formule van het echo-feedbackcircuit dat we in paragraaf 4.4 hebben besproken.
Het geluid van S2 werd gemaakt met de feedbackgenerator die we zijn tegengekomen in opdracht 61. De feedbackgenerator voert "achter de schermen" precies diezelfde recursieve formule van een echo-feedbackcircuit uit, maar doet nog veel meer. Misschien is het je ook opgevallen dat de feedbackgenerator een stuk sneller is.

b. Wis in het commandovenster alle code onder de stippellijn uit. Dat is de code voor S3 en die hebben we niet meer nodig, want we werken nu uitsluitend nog met de feedbackgenerator. We gaan nu eerst demonstreren dat de feedbackgenerator een gewone toongenerator is - ook dat deden we al in opdracht 42). Vul achter **`Frequentie (Hz of TOETS)`** in: **`10 Hz`** en klik op **Start**. Herhaal dat met achtereenvolgens 20, 100 en 1000 Hz.

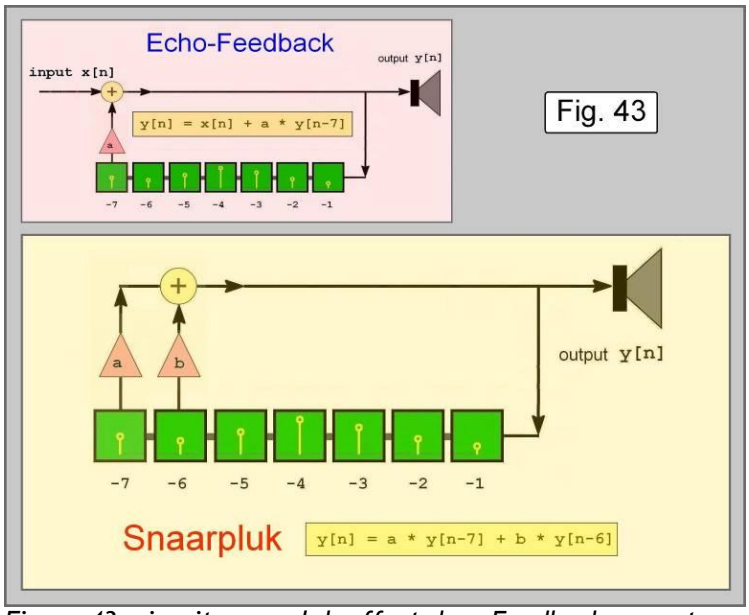
c. Nu gaan we een nieuwe karaktertrek van de feedbackgenerator bekijken. Open ►Preset_H7_Feedbackgenerator.txt. Kijk naar de code. Je hebt nu voor S2 en S3 een aparte feedbackgenerator. Ze zijn bijna hetzelfde ingesteld. Vergelijk de waarden van twee onderste instellingen nl. **`Filtercoefficient a`** en **`b`**. In het bovenste codeblok (voor S2) zijn die getallen resp. 1 en 0. In het onderste blok zijn ze alle twee gelijk aan 0,5.
Klik op **Start** en ga naar het Scoop / Spectrum-venster. Selecteer achtereenvolgens S2 en S3. Wat is het verschil in klank en in spectrum?

d Ga naar het commandovenster en stel van beide generatoren de frequentie in op achtereenvolgens 20, 100 en 1000 Hz. Ga telkens naar het Scoop / Spectrum-venster en vergelijk S2 en S3. Wat is het verschil in klank en in spectrum?

Conclusie: als de twee getallen **`Filtercoefficient a`** en **`b`** gelijk zijn aan 0,5, dan produceert de feedbackgenerator het geluid van een getokkelde snaar! De vraag is nu natuurlijk: hoe komt dat?

Circuit en formule van snaarpluk

In figuur 43 is het circuit te zien waarmee de feedbackgenerator dit snaarpluk-effect maakt. Ter vergelijking zie je boven, in 't klein, ook het oorspronkelijke echo-feedbackcircuit nog eens, zoals we dat in hoofdstuk 4 hebben besproken. Dat oorspronkelijke circuit had slechts één versterkingsfactor (of beter: verzwakkingsfactor), namelijk *a*.
Het snaarpluk-circuit daarentegen heeft een extra signaalpad met een extra versterkingsfactor *b*. De paden van *a* en *b* worden gemixt voordat ze naar de output en de delay-lijn gaan.



Figuur 43: circuit snaarpluk-effect door Feedbackgenerator.

In hoofdstuk 4 hebben we gezien dat je een feedbacksysteem wiskundig kunt formuleren met de recursieve differentievergelijking:

$$y[n] = x[n] + ay[n-N] \quad \text{echo-feedbackvergelijking}$$

Als je ervan uitgaat (zoals we in opdracht 64 in feite al stilzwijgend deden) dat op tijdstip 0 de hele delay-lijn al gevuld is met een signaalfragment, dan kunnen we het circuit een toon laten maken zonder dat er sprake van een inputsignaal. Je krijgt dan de vergelijking:

$$y[n] = ay[n-N] \quad \text{echo-feedbackvergelijking zonder input}$$

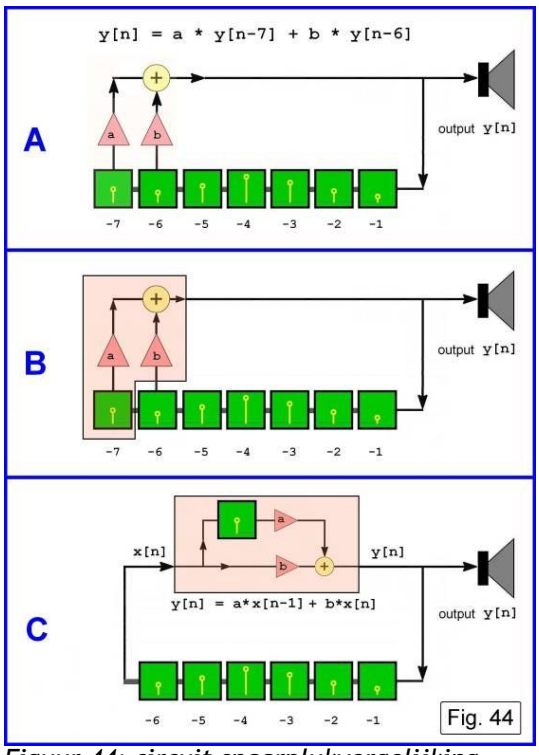
Door aan de rechterkant van het = teken één extra term in te voegen, krijgen we dan het circuit waarmee we het snaarpluk-effect hebben gemaakt:

$$y[n] = by[n-N+1] + ay[n-N] \quad \text{snaarplukvergelijking} \; (16)$$

Omdat we zo altijd, ongeacht wat voor signaalfragment we kiezen (in opdracht 64 was dat het geluid van brekend glas!) een geluid krijgen dat niet alleen steeds zachter maar ook steeds doffer wordt en verbluffend veel lijkt op dat van een getokkelde, " geplukte" snaar, noemen we bovenstaande vergelijking de ***snaarplukvergelijking***.

De werking van snaarpluk

Om de werking van de snaarplukvergelijking te verklaren, hoef je eigenlijk alleen maar het circuit een beetje anders te tekenen. In figuur 44 zie je het drie keer onder elkaar afgebeeld. De bovenste (A) zag je ook al in figuur 43. In de middelste (B) is een gedeelte van het circuit afgebakend door een vak met een licht rode achtergrond. In de onderste (C) zie je dat afgebakende circuit nog eens, maar dit keer zijn de elementen alleen maar een beetje verschoven. Nu wordt goed zichtbaar dat het afgebakende circuit exact hetzelfde is als figuur 31: een feedforward-echo-circuit! En zo'n circuit heeft de werking van een filter.



Figuur 44: circuit snaarplukvergelijking.

Kortom: het snaarpluk-circuit is een feedbackcircuit met een feedforward filtertje erin! De samples doorlopen steeds de delay-lijn en worden bij elke doorgang weer opnieuw door dat filter gehaald. Wat er bij herhaald filteren gebeurt, hebben we in opdracht 58 al bekeken: als een filter een amplitudekarakteristiek *A(f)* heeft en je een "wit" geluid met constant spectrum *c* (zie paragraaf 5.2) door het filter haalt, dan krijg je als output $c \cdot A(f)$. Haal je dat outputsignaal vervolgens nog eens door datzelfde filter, dan wordt de output $c \cdot A(f)^2$. Als je steeds weer opnieuw filtert, dan zal het outputsignaal na de *K*-de filterdoorgang een spectrum $c \cdot A(f)^K$ hebben.

Hoe dat er precies uitziet bij het snaarpluk-circuit onderzoeken we in opdracht 65.

65. Opdracht: herhaald filteren met het snaarplukfilter

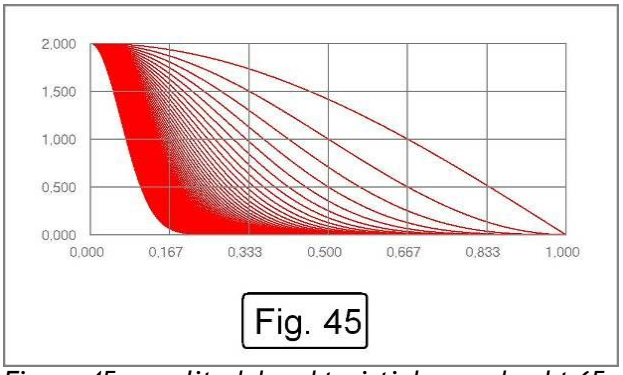
In opdracht 64 waren **`Filtercoefficient a`** en **`b`**. alle twee gelijk aan 0,5. We krijgen dan de differentievergelijking:

$$y[n] = 0{,}5x[n] + 0{,}5x[n-1]$$

In opdracht 56 heb je nagerekend dat de amplitudekarakteristiek $A(\omega)$ van dit filter gelijk is aan $\cos(\frac{1}{2}\omega)$.

a. Herhaal opdracht 56 om de grafiek te tekenen van amplitudekarakteristiek $A(\omega) = \cos(\frac{1}{2}\omega)$. Maak daarbij gebruik van ►Preset_H7_Opdracht_65_a.txt, vul (helemaal onderaan de code!) de juiste formule in en klik op **Start**.

b. Teken in één grafiek de amplitudekarakteristieken van $A(\omega)$, $A(\omega)^2$, $A(\omega)^3$,...., $A(\omega)^{999}$, $A(\omega)^{1000}$.
Maak daarbij gebruik van ►Preset_H7_Opdracht_65_b.txt, vul (weer helemaal onderaan de code) de ontbrekende gegevens in en klik op **Start**. Als resultaat krijg je figuur 45.



Figuur 45: amplitudekarakteristieken opdracht 65.

c. Je hebt bij vraag (b) 1000 grafieken in één figuur gezet. Je zag daarbij een animatie die illustreert hoe een oorspronkelijk "wit" spectrum door het steeds maar weer te filteren de hogere frequenties steeds verder onderdrukt, waardoor de klank ook steeds doffer wordt. Precies datzelfde zou je te zien moeten krijgen in het spectrum-venster als je gebruik maakt van de feedbackgenerator en **`ruis`** invult achter **`Golfvorm`**. Is dat inderdaad zo? Ga dat na door gebruik te maken van ►Preset_H7_Opdracht_65_c.txt, vul in, klik op **Start**, ga naar het Scoop / Spectrumvenster, selecteer S1. Hoe ontwikkelt zich het spectrum?

In opdracht 66 gaan we snaarpluk toepassen in een stuk muziek.

66. Opdracht: Chopin op snaarpluk-klavecimbel

a. Open (niet in WaveWizard, maar via bijv. Windows Mediaplayer) het MP3-bestand ►H4_Chopin_Etude_3_Clavecimbel.mp3 en beluister het goed. De klank lijkt wel een beetje op die van een klavecimbel, een tokkelinstrument. Als je op een klavecimbel een toets aanslaat, tokkelt een harde pen de snaar. Daardoor ontstaat een heel heldere, briljante toon.

b. Dit MP3-bestand werd met WaveWizard gemaakt en jij gaat nu proberen om die klank na te maken!

Open MIDI-bestand ►H4_Chopin_Etude_3.mid.

Open ►Preset_H7_Opdracht_66_Clavecimbel.txt.

c. Klik op **Start**. Alleen de eerste strofe (104 van 1884 noten) wordt uitgerekend. Luister naar de klank. Je hoort dat dit geluid niet hetzelfde is als dat van het MP3-bestand, ook al lijkt het er wel op. Probeer kort onder woorden te brengen waarin de twee geluiden van elkaar verschillen.

d. Net zoals bij een piano kun je ook op een klavecimbel met één toets twee (of zelfs drie) snaren tegelijk aanslaan. Dat geeft niet alleen meer volume, maar ook lichte zwevingen, want snaren zijn nooit helemaal identiek gestemd. Door die zwevingen wordt de klank voller en levendiger.

Maak een kopie van het codeblok van de feedbackgenerator (selecteren met de muis en dan Ctrl C en Ctrl V). Je hebt nu twee generatoren; zorg dat die tweede het geluid naar S2 stuurt.

Ook de toonhoogte van de tweede generator is gelijk aan **`Freq[k]`**. Maak een lichte zweving door **`Freq[k]`** te vermenigvuldigen met een getal dat iets groter (of kleiner) is dan 1. Experimenteer een beetje.

e. Echte snaren hebben een zekere mate van stugheid. Het klankeffect daarvan is dat de boventonen niet meer zuiver harmonisch zijn; daardoor wordt de klank ietsje metaalachtiger. Een van de vele wonderen van het snaarpluk-circuit is, dat je dit effect kunt maken door de filtercoëfficiënten *a* en *b* ongelijk aan elkaar te maken, maar zo dat hun som nooit groter is dan 1, bijv. *a* = 0,1 en *b* = 0,9. Geef beide generatoren een verschillende mate van stugheid. Experimenteer!

f. Bij de snaarplukvergelijking klinken hoge tonen wat zachter dan lage tonen. Dat merk je pas als je er een heel stuk mee gaat spelen. Wat je kunt doen om het volumeverloop over het klavier afhankelijk te maken van de toonhoogte is achter **`Amplitude`** een formule neerzetten waarin naast Amp[k] ook Freq[k] voorkomt. Bijvoorbeeld **`Amp[k] * 0,004 * Freq[k]`**. Luister hoe de "balans" daardoor verandert.

g. Als je tevreden bent over de klank, laat de computer dan alle noten van het stuk uitrekenen.

67. Opdracht: don't shoot the piano...

Een piano is, in tegenstelling tot een klavecimbel, geen *tokkel*instrument, maar een *slag*instrument: door een toets aan te slaan, sla je met een vilten hamertje tegen de snaar. De klank is daardoor veel krachtiger, maar ook veel doffer dan de klank van een klavecimbel. Een ander gevolg van dit aanslaan is dat de geluidsenergie direct in de attack van de toon grotendeels vrijkomt. Ook daardoor maakt de toon een vollere en luidere indruk. We proberen het geluid van een piano te benaderen (een heel klein beetje...!) door het geluid van de feedbackgenerator door een laagdoorlaatfilter te halen. We kiezen nu geen ruis als golfvorm maar een zaagtand. (Je kunt daar natuurlijk mee experimenteren! Pak eens een blokgolf!) De bedoeling van deze opdracht is vooral om te zien hoe het laagdoorlaatfilter ingesteld moet worden.

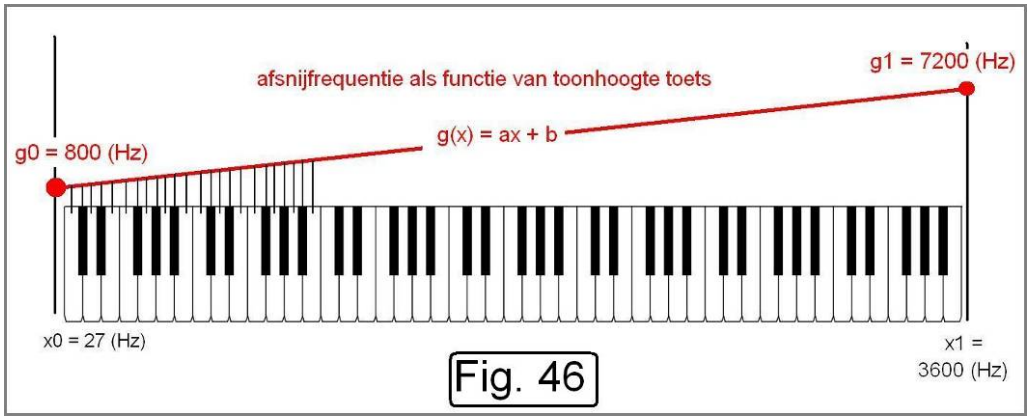
a. Open ►Preset_H7_Test_Pianofilter.txt.
Het is een hele lap tekst, maar als je even kijkt zie je dat het vrij eentonig is. Om het geluid van een enkele snaar te maken worden twee feedbackgeneratoren gebruikt; de ene met een heel lange, de andere met een korte decay-tijd; daarmee wordt dat "slag-"geluid van de hamer benaderd. Beide snaarcomponenten worden gemixt en vervolgens gefilterd. Daarmee heb je dan het geluid voor spoor S1. Op S2 gebeurt precies hetzelfde, alleen is de snaar, net zoals in opdracht

66, een beetje ontstemd. Let op de Duur van de tonen: elke volgende toets heeft een iets kortere duur, zodat ze op het einde allemaal tegelijk klinken en ophouden. Zo creëer je het effect van het doorklinkpedaal (zie hoofdstuk 1). Klik op **Start** en beluister het geluid.

b. De klank lijkt totaal niet op die van een piano. Dat komt omdat de afsnijdfrequentie van het laagdoorlaatfilter staat afgesteld op 20000 Hz; het doet dus eigenlijk nog niets. Twee regels onder de `FOR...TO`-instructie staat de variabele `CutOff`. Hiermee wordt de afsnijdfrequentie van het laagdoorlaatfilter geregeld. Verander de waarde in achtereenvolgens 400 en 2000 en beoordeel de effecten daarvan. Wat gebeurt er bij elke instelling met de hoge tonen? En met de lage?

c. Bij (b) heb je twee dingen kunnen horen: (1) als je het filter instelt op 2000 Hz, dan klinken de hoge tonen vrij helder, maar in de lage toetsen hoor je die storende "buzz"-klank van de zaagtandgolf. (2) Die buzz kun je wegfilteren door de afsnijdfrequentie veel lager in te stellen, bijvoorbeeld op 400 Hz. Maar dan klinken die hoge tonen heel dof of je hoort ze zelfs helemaal niet meer! Kortom, het is nooit goed! Toch is er een oplossing: maak de afsnijdfrequentie afhankelijk van de toets! Dat kunnen we doen met de techniek die je al vaak hebt toegepast: definieer een lineaire functie $g(x) = ax + b$. Zie figuur 46. In de code kun je dat snel veranderen door in de regel van `CutOff` de formule die nu rechts van het uitroepteken staat te activeren.

d. Om te horen hoe die variabele afsnijdfrequentie uitpakt in een pianostuk open je eerst ►Rachmaninov_Rhapsodie_Var18.mid en vervolgens ►Preset_H7_Opdracht_67_Piano.txt. Dit romantische stuk van Rachmaninov is echt een wereldhit en figureert in nogal wat speelfilms, bijvoorbeeld in de komedie Groundhog Day, met Bill Murray achter de toetsen!



Figuur 46: afsnijdfrequentie als functie van toonhoogte toets.

7.3 Tijdvariante filters

In opdracht 67 hebben we tonen zodanig gefilterd dat voor elke toets op de piano de afsnijdfrequentie opnieuw werd ingesteld. Een ***tijdvariant*** filter is een filter waarvan de eigenschappen (zoals afsnijdfrequentie en bandbreedte) voor elk tijdstip *n* opnieuw worden ingesteld. Het enige dat we daarvoor moeten doen is achter **`Afsnijfrequentie`** (van het Laagdoorlaatfilter) een formule invoeren waarin **`n`** voorkomt. Bij het Banddoorlaatfilter zijn er twee velden die je op dezelfde manier tijdvariant kunt maken: **`Centrumfrequentie`** en **`Bandbreedte`**.

68. Opdracht: tijdvariant filter voor klanksynthese en -analyse

a. Van een Banddoorlaatfilter moet de Centrumfrequentie in 6 seconde lineair toenemen van 200 Hz tot 2500 Hz. De bandbreedte is 300 Hz.

Open ►Preset_H7_Opdracht_68a.txt en klik op **Start**. Lees de commentaartekst bij de code goed door. Daar lees je dat het veranderen van de Centrumfrequentie met dezelfde techniek gebeurt als die we gebruikten voor fade in en fade out (zie paragraaf 4.1). Ga naar het Scoop / Spectrum-venster en observeer het resultaat (selecteer S2).

b. Je kunt een banddoorlaatfilter prima gebruiken om de boventonen van een toon afzonderlijk hoorbaar te maken. Dit is het omgekeerde van "door-fluiten". Bij doorfluiten stuur je een sinustoon waarvan de frequentie langzaam toeneemt door een filter, om de eigenschappen van dat filter (de amplitudekarakteristiek) vast te stellen. Bij "door-filteren" halen we een toon waarvan we graag de afzonderlijke boventonen willen horen door een filter dat een smalle doorlaatband heeft waarvan de centrumfrequentie geleidelijk toeneemt.
Open ►Preset_H7_Opdracht_68b.txt. Er wordt een zaagtandtoon van 400 Hz aangemaakt die 30 sec duurt. Laat de Centrumfrequentie in 30 sec toenemen van 200 tot 4000 Hz. Vul de ontbrekende gegevens, activeer de Sonogram-code in en klik op **Start**. Beluister het resultaat.

"Klankwet-filter"

WaveWizard beschikt over een speciaal filter dat altijd automatisch voldoet aan de klankwet. Dit filter heet **`Low Pass, amplitude-afhankelijk`**. Het meet voortdurend de amplitude van het inputsignaal en op basis daarvan wordt de afsnijdfrequentie steeds bijgestuurd: als een geluid sterker wordt, dan gaat ook de afsnijdfrequentie mee omhoog. Daalt de amplitude, dan daalt ook de afsnijdfrequentie mee.
Het enige dat je zelf moet instellen is de gevoeligheid. Dat doe je via twee invoerregels **`F0`** en **`F1`**.
Bij **`F1`** vul je de afsnijdfrequentie in die het filter dient te hebben als de maximale amplitude wordt bereikt. Bij **`F0`** geef je de afsnijdfrequentie aan

die het filter dient te hebben bij amplitude nul (dat wil zeggen "het zachtste geluid").
Dit klankwetfilter heeft nog een andere bijzondere eigenschap die je ook bij synthesizer-filters aantreft: op de afsnijdfrequentie kun je een resonantie maken, dat wil zeggen het filter werkt dan als de combinatie van een laagdoorlaat- en een (nogal primitief) banddoorlaatfilter, waarvan de bandbreedte **`Reso-piekbreedte`** heet.

69. Opdracht: huilende wind
Als wind "vlagerig" is dan hoor je een ruisachtig geluid in vlagen harder en zachter worden. Als de wind bovendien tegen een scherp voorwerp botst, bijvoorbeeld een dakrand of een spelonk in een rotswand, dan krijgt het ook een fluitend geluid (zogeheten wigtoon, gevolg van turbulentie): evenals het volume neemt ook de toonhoogte van dit fluitgeluid toe met de windkracht. Die voortdurende schommeling van toonhoogte neem je waar als een huilerig geluid.
Hieruit blijkt dat de klankwet nog iets meer omvat dan we tot nu toe beweerden. Bij wind en bij blaasinstrumenten moeten we eigenlijk formuleren als: hoe harder, hoe helderder èn hoger.
Open ►Preset_H7_Opdracht_69_Huilende_wind.txt.
In de code zelf vind je meer uitleg over de opbouw van dit geluid. Met **`Reso-piekbreedte`** stel je de bandbreedte in. Die is bepalend voor de "fluiterigheid - huilerigheid" van de ruis. (Je weet uit opdracht 50 dat ruis steeds "sinus-toniger" gaat klinken naarmate je de doorlaatband van het filter smaller maakt.)
Experimenteer met **`F0`** en **`F1`**. en **`Reso-piekbreedte`** om verschillende fluitende, huilende, gierende geluiden te maken!!

70. Vraag
Waarom is het onmogelijk om een filter te maken waarmee je ook die uitgebreidere versie van de klankwet kunt realiseren, namelijk hoe harder, hoe helderder èn hoger?
Het antwoord las je in paragraaf 5.4 bij de uitleg over cos $\omega(n - ½)$. Een filter is en blijft niets anders dan een (complex) echo-effect.

71. Slotopdracht
Je hebt inmiddels heel wat technieken leren kennen en toepassen. In deze laatste opdracht neem jij het initiatief. Maak eens zelf (of samen met een ander) een interessant, leuk, mooi of raar geluid en vertel in de klas wat je precies gedaan hebt aan de hand van de preset die je hebt gemaakt.
Er zijn heel veel mogelijkheden. Je kunt uitgaan van een MIDI-bestand (bedenk wel dat in WaveWizard nog niet de volledige MIDI-taal is geïmplementeerd! De bestanden die je vindt op ►URL7 werken in elk geval goed op WaveWizard).

Maar je kunt ook een random toonreeks of random filterreeks maken of je kunt uitgaan van een WAV-bestand dat je gaat vervormen met filters of echo-effecten!
Een paar tips:

- Ga nooit rechtstreeks instructies zitten intypen in het Commando-venster! De kans is groot dat je typefouten maakt en dan krijg je allerlei foutmeldingen, wat heel gauw gaat irriteren. Ga altijd uit van presets en open er zo nodig meerdere (met **Open Append**). Je krijgt dan wel een boel tekst, maar daarin kun je dan vrij gemakkelijk een beetje gaan "shoppen": gooi weg wat je niet nodig hebt. Let goed op de eventuele aanpassingen die je moet doen. (Bijvoorbeeld: welke variabelen worden gebruikt? Worden die in de aangevoegde codeblokken ook goed doorgekoppeld?)
- Herinner je opdracht 1: doe wat elke sound designer doet: maak het geluid eerst eens globaal met je stem of met een handbeweging. Een ontwerper van Ferrari begon een nieuw sportwagenmodel 't liefst met een vluchtig schetsje op een envelop of in de marge van een oude krant! Het gaat eerst om de grote lijnen, dan pas de details.
- Begin altijd simpel en breid je code dan voorzichtig uit naar meerdere instructies. Test de afzonderlijke elementen ook afzonderlijk, als het even kan.
- Breng kleine veranderingen aan in waarden en formules en luister hoe dat uitpakt.
- Laat de computer niet steeds duizenden noten berekenen. Probeer eerst even een paar noten, dat scheelt veel tijd en geeft je toch al een indruk van het geluid.
- Op ►URL1 vind je nog meer presets voor WaveWizard die je misschien kunnen inspireren!
- Heb je zelf een interessante preset gemaakt? Stuur die op naar info@muziekexact.nl onder vermelding van "NLT preset"!

Suggestie

Om je een idee te geven, wat dacht je van de volgende verrassende effecten die je kunt maken als je het klankwetfilter toepast in bijvoorbeeld die piano van opdracht 67. Luister eens naar:
►H1_Chopin_Etude_3_Resonantiefilter.mp3,
►H1_Chopin_Etude_3_Reso_zwiep_omhoog.mp3, en
►H1_Chopin_Etude_3_Reso_zwiep_omlaag.mp3.
De corresponderende preset laat zien wat er gebeurt:
►Chopin_Etude_3_reso_zwiep_omhoog.txt.
Experimenteer met het zwiep-effect door andere waarden voor **`F0`** en **`F1`** en **`Reso-piekbreedte`** te kiezen.

Woordenlijst

adaptief	§1
Reso-piekbreedte	§3
Snaarplukvergelijking	§1, §2
tijdvariant filter	§1, §3

Tot slot

Hiermee zijn we aan het einde gekomen van de module **Sound Design**. We hebben een aantal technieken bestudeerd waarmee je behoorlijk complexe geluiden kunt maken die bovendien voldoen aan een algemene, akoestische vuistregel, de klankwet.
We hopen dat je het leuk bent gaan vinden om zelf te experimenteren met geluid en dat je dat ook in de rest van dit schooljaar af en toe nog eens doet!
Mocht je opmerkingen, vragen of suggesties hebben, stuur dan een mail naar info@muziekexact.nl.

Bijlage 1 URL-lijst

URL1	Muziek Exact http://www.muziekexact.nl/ Website voor sound design projecten t.b.v. het voortgezet onderwijs
URL2	Muziek Exact http://muziekexact.nl/MM_DSP15/MM_DSP_15.htm Informatie over vocoding
URL3	Budapest University of Technology and Economics http://home.mit.bme.hu/~bank/ Proefschrift Balázs Bank
URL4	Koninklijk Conservatorium in Den Haag, Sonologie http://www.koncon.nl/nl/Studierichtingen/Sonologie/
URL5	Hogeschool voor de Kunsten Utrecht, Sound Design http://www.hku.nl/web/Studiekeuze/Muziek/SoundDesign.htm
URL6	Muziek Exact http://muziekexact.nl/Harmonischen.pdf Informatie over harmonischen
URL7	Muziek Exact http://muziekexact.nl/MIDI.htm Bestanden MIDI