Spraaksignaalanalyse per Computer

David Weenink
email: David.Weenink@hum.uva.nl

26 september 2002

Inleiding

Het college *Spraaksignaalanalyse per Computer* beslaat 10 collegeweken. Per week zijn er twee tot vier lesuren. Het college gaat hand in hand met het door Paul Boersma en David Weenink ontwikkelde computerprogramma *praat*[1].

In het college zullen een aantal basisklasses[2] van het programma *praat* aan de orde komen samen met conversies hiertussen. Enkele voorbeelden van klasses zijn **Sound**, **Spectrum**, **Spectrogram**, **Pitch**, **Cepstrum**, **LPC**, **Formant**, **Cochlegram**, **TextGrid**, **Analysis** en **PointProcess**. Enkele voorbeelden van conversies zijn **Sound_to_Spectrum** en **Sound_to_Pitch**. Deze conversies bieden een kapstok om in te gaan op de theorie achter de conversie.

Behalve bovengenoemde klasses zijn er nog tientallen andere die gebruikt kunnen worden voor spraakanalyse, -synthese, en -herkenning en voor statistische bewerkingen.

0.1 Overzicht van de inhoud van de colleges

Week 1 Administratie, inhoud van het college.
Inloggen, praktikum opdrachten, programma praat, scripts
Sound: Representatie, kwantisatie, bestandsformaten
Opdracht: via formulier en script: maken voor generatie eenvoudige signalen.

Week 2 Sound: Edit & TextGrid etc..
Modify & Query
Opdracht: Lineaire kwantisatie en μlaw

Week 3&4 Spectrum, Spectrogram
Fourieranalyse
Spectrogram_draw: opties
Opdrachten: Invloed vensters en representatie.

Week 4 Pitch:
Sound_to_Pitch: schema algorithme Boersma
Pitch_to_PointProcess
Opdracht 2: Pitch-analyse+manipulaties

Week 6&7 Filtering: LPC, Cepstrum, MFCC, Cochleagram
Inleiding LPC-analyse via autocorrelatiemethode, Z-transform (simpel) Sound

[1] `http://www.fon.hum.uva.nl/praat`
[2] In de zin van object georiënteerde klasses

+ LPC: filter en inverse filter
Opdrachten: synthese, resynthese

Week 8 Formant: analyse, problemen
Opdracht: meten eigen formanten

Week 9 Analysis: manipulaties
Opdracht: veranderen intonaties

Week 10 Databases: TIMIT, SQL, labelling
Opdrachten: maken van queries op TIMIT, gemiddelde duren beklemtoonde & onbeklemtoonde klinkers

Hoofdstuk 1

Het programma *praat*

1.1 Inleiding

Het computerprogramma *praat* is ontwikkeld door Paul Boersma en David Weenink, beiden werkzaam bij Fonetische Wetenschappen van de Universiteit van Amsterdam. Het programma is, onder anderen, erg geschikt om geluidssignalen te analyseren en te manipuleren. Wereldwijd wordt het programma veel gebruikt door fonetici en door fonologen. Er zijn ook biologen die het gebruiken om vocalisaties (geluiden) van bijvoorbeeld apen, dolfijnen of olifanten te analyseren.

Alhoewel het programma in voortdurende ontwikkeling is, zal er aan de manier waarop de gebruiker het programma bedient, de interface, voorlopig niet zo veel veranderen. De manier van werken die *praat* min of meer afdwingt verandert namelijk niet. Ontwikkelingen aan het programma betreffen hoofdzakelijk het toevoegen van nieuwe klasses en/of het toevoegen van methodes aan bestaande klasses. De huidige versie van het programma is 4.0.28. Via het interface kunnen we objecten manipuleren. Deze objecten hebben te maken met bijvoorbeeld de wereld van de spraaksignaalanalyse of de wereld van de fonetiek of de wereld van de fonologie. De objecten bevinden zich (meestal) in het geheugen van de computer. Dit betekent dat ze niet meer bestaan als je het programma *praat* verlaat.

Behalve voor het analyseren van geluiden kun je er ook plaatjes mee maken van hoge kwaliteit die je naar een PostScriptprinter kunt sturen of als (*encapsulated postscript*) bestand kunt bewaren om in een verslag te zetten.

Een ander kenmerk van het programma is dat je er zowel interactief als vanuit de *batch* mee kunt werken. Als je een bestand maakt met commando's voor *praat*, dan kun je de commando's in dit bestand laten uitvoeren door *praat*. Vooral als je duizenden bestanden wilt analyseren is dit erg handig.

Natuurlijk kun je ook als je *praat* hebt opgestart binnen het programma *hulp*

vinden. Deze hulp is in de Engelse taal en er zijn twee soorten. Allereerst zijn er een reeks *tutorials* die de gebruiker redelijk aan het handje meevoeren om hem vertrouwd te maken met iets nieuws. Verder hebben de meeste commando's waar de gebruiker een formulier moet invullen een `Help`-knop. De vertoonde informatie is vaak kort en bondig en op het nivo van iemand die weet waar hij mee bezig is.

Deze handleiding geeft hopelijk wat extra hulp bij het gebruik van *praat* en probeert ook een deel van de achterliggende signaalverwerkingstheorie te verduidelijken. Vaak zal hierbij ook tekst overgenomen worden uit of verwezen worden naar de interactieve hulppaginas van *praat*. Ook kan het zijn dat delen van deze handleiding (vertaald) weer opgenomen zullen worden in het programma zelf.

1.2 Het opstarten en verlaten van het programma

Wanneer je het programma *praat* interactief gebruikt, dan doe je dit in een grafische omgeving. Dit kan zijn op een Macintosh, een PC met Windows (98, NT, 2000, XP) of een Unix variant zoals Linux.
Meestal is klikken op het praat-ikoontje voldoende om het programma op te starten.

> Je verlaat het programma weer door onder het meest linkse menu, `Control`, de optie `Quit` te kiezen.
> Alle objecten die niet naar een bestand op schijf zijn geschreven worden vernietigd!

Als je nog nooit met *praat* gewerkt hebt is de snelste manier om wat te proberen de volgende. Kies onder het `New`-menu de optie `Sound` en hierbinnen weer `Create Sound...` en druk op de `OK`-knop.

1.3 Objecten

De wereld van *praat* bestaat uit objecten. Objecten worden gecreëerd, gemanipuleerd, gelezen van schijf, geschreven naar schijf, geconverteerd naar andere objecten, gequeried, getekend en uiteindelijk misschien wel weggegooid. In de wereld van de objecten is de werkvolgorde "Object-Verb": éérst moeten we een object selecteren en pas dán kunnen we er iets mee doen. Elk object is van één bepaalde *klasse*. Klasse kun je vergelijken met *datatype*. Er kunnen meerdere objecten van dezelfde klasse bestaan, bijvoorbeeld meerdere `Sound`-objecten.

1.4 Het maken van objecten

Het creëren van nieuwe objecten kan in praat op een aantal manieren:

New/Create Kies een optie uit het `New`-menu. Bijna alle keuzes die je nu hebt beginnen met het werkwoord `Create`. Voor lang niet alle klasses in *praat* kun je op deze manier een object creëren. Wat je altijd kunt doen om aan de gang te komen is een standaard geluidje maken: Kies de optie `Create Sound...` en druk op de `OK`-knop.

Read Via het Read-menu kun je één of meerdere objecten uit een bestand van schijf lezen. Dit object verschijnt, als alles goed is gegaan, in de objectenlijst (list of objects).

Conversie Door een analyse uit te voeren op een object van type A krijg je een nieuw object van type B. Bijvoorbeeld een spectrale analyse van een `Sound`-object geeft een `Spectrum`-object. Een toonhoogte-analyse van een `Sound`-object geeft een `Pitch`-object.

New/Record... Maak een opname die kan resulteren in één of meerdere `Sound`-objecten. Deze wijkt af van de bovenstaande methodes omdat hij alleen voor het `Sound`-object geldt.

1.5 Het bewaren van objecten

De enige manier om objecten te bewaren is als een bestand op schijf. De procedure is dan: selecteer één of meerdere objecten en kies een schrijfoptie uit het `Write`-menu.

> Heel vaak is het niet nodig om objecten die het resultaat zijn van een analyse te bewaren op schijf: het is veel beter om het script te bewaren dat tot de analyse-objecten gevoerd heeft.

Objecten die je altijd moet bewaren zijn objecten waar je zelf veel aan veranderd hebt. Als je bijvoorbeeld delen van een geluid hebt gelabeld dan wil je het `TextGrid`-object natuurlijk wel bewaren. Ook als je een `Pitch`-object handmatig hebt gecorrigeerd, dan wil je dit wel bewaren.

1.6 De syntax van het gebruikersinterface

In *praat* is een consequente commando-syntax doorgevoerd.

- Alle opschriften op knoppen beginnen met een hoofletter. De enige uitzondering zijn de getallen 10, 12,... in het `Picture`-venster onder het `Font`-menu.
- De text op knoppen die eerst een formulier op het scherm zetten voordat ze verdere actie ondernemen eindigt altijd op drie punten "...". Een voorbeeld is `Record Sound...` in het `New`-menu.
- De text van knoppen die de actie direct uitvoeren bevat geen punten ".". Voor een `Sound`-object is dit bijvoorbeeld het `Play`-commando.
- Alle acties met een object die geen nieuw object opleveren beginnen met een werkwoord zoals bovenstaand `Play`-commando.
- Acties onder knoppen die een nieuw object opleveren beginnen met `To`.
- In scripts worden muis-selectie-acties weergeven met kleine letter: `select`, `plus` en `minus`.
- In scripts worden de beschrijving van een formulier gezet tussen `form` en `endform`.
- In scripts beginnen de namen van *variabelen* met een kleine letter, variabelen die tekst bevatten eindigen dan nog op een extra dollarteken (`$`).

1.7 Een klasse-overzicht

De klasses die de gebruiker in het fonetische deel van *praat* ten dienste staan zijn de volgende:

- Algemene klasses.

  Matrix Een reeëlwaardige bemonsterde functie van twee variabelen.

  Polygon Een tweedimensionele veelhoek.

  Sound Een bemonsterd signaal. De belangrijkste klasse binnen *praat*. Vrijwel alle analyses beginnen met een object van dit type.

  LongSound Een Sound die niet in het geheugen van de computer zit maar op schijf.

  TableOfReal Een matrix waarvan de rijen en kolommen een naam kunnen hebben.

- Periodiciteitsanalyse.

Pitch De artikulatorische *fundamentele frequentie* of de akoestische *periodiciteit* of de perceptuele *pitch*.

Harmonicity De mate van periodiciteit.

Intensity Intensiteitscontour.

- Spectrale analyses.

  Spectrum Het complexwaardige frequentiespectrum.

  LTAS Het **L**ong **T**erm **A**verage **S**pectrum.

  Spectrogram Het spectrum als functie van de tijd.

  Formant Formantfrequenties en -bandbreedtes als functie van de tijd.

  LPC De *lineaire predictie* coëfficiënten als functie van de tijd.

  Excitation Het excitatiepatroon van het basilair membraan.

  Cochleagram Het excitatiepatroon als functie van de tijd.

- Labelen en segmenteren van geluid.

  TextGrid Een object met labelinformatie. Een TextGrid kan uit één of meer zogenaamde *tiers* bestaan.

  IntervalTier Een serie aaneensluitende tijdsintervallen die gemarkeerd kunnen zijn.

  PointTier Een op tijd gesorteerde verzameling gemarkeerde tijdstippen.

- Neurale netten

  FFNet Een feed forward neuraal net.

  Pattern De invoer van een neuraal net.

  Categories De uitvoercategorieën van een neural net.

- Numerieke en statistische analyses

  Eigen Eigenwaardes en eigenvectoren.

  Covariance Een covariantiematrix.

  PCA Een Principale Componenten Analyse.

  Discriminant Discriminanten analyse.

  ContingencyTable Two-way contingency table.

- Multidimensionele schaling

Configuration

Dissimilarity, Simmilarity

Distance Een matrix met afstanden tussen objecten.

- Optimality theorie.

  OTGrammar

1.8 Ontwerpprincipes

De interactie van *praat* met zijn omgeving, de gebruiker, wordt afgehandeld door de praat-kern (the shell). Deze kern is ontworpen met als doelen

flexibiliteit Zowel interactief- als batch-gebruik moet mogelijk zijn.

portabiliteit Macintosh, Windows en Unix moeten ondersteund worden.

herbruikbaarheid De werelden van bijvoorbeeld de fonetiek, de fonologie, de schalingsanalyses en zo voorts, moeten ondersteund kunnen worden.

bruikbaarheid De mens-machineinteractie moet voldoen aan algemeen aanvaarde principes.

onderhoudbaarheid Lichtgewicht, goed onderhoudbare kode. Dit impliceert, om met een modewoord te spreken, een objectgeöriënteerd ontwerp.

We proberen de fysische wereld zo *betrouwbaar* en *precies* mogelijk te modelleren.

Hoofdstuk 2

Het scripten van praat

2.1 Inleiding

In *praat* is een tutorial aanwezig over scripting die te vinden is in het `Scripting tutorials`-deel van het `Help`-menu. We zullen hiervan een korte samenvatting geven aan de hand van twee eenvoudige voorbeelden. Als eerste voorbeeld gaan we een script maken dat de gebruiker in staat stelt een kort toontje te beluisteren. In het tweede script gaan we een reeks "toontjes" bij elkaar optellen. We gebruiken hiervoor een zogenaamde *for-loop* om een bepaald commando meerdere keren te herhalen. We laten vervolgens zien hoe we een knop kunnen toevoegen (en weghalen) aan het vaste menu die dit script uitvoert zonder dat we een `ScriptEditor` geopend hoeven te hebben.

Het belangrijkste om te onthouden is dat je niet veel nieuwe commando's hoeft te leren om te gaan scripten. De commando's zijn namelijk identiek aan de opschriften op de knoppen die je moet kiezen als je het commando interactief uitvoert.

2.2 Het eerste script: een sinus

Acties achter knoppen worden onthouden door *praat*. Een snelle manier om een script te maken is door hier gebruik van te maken. We voeren de acties die gedaan moeten worden in het script één keer achter elkaar uit in de interactieve modus. Dan openen we de `ScriptEditor` en plakken de tekstrepresentatie van deze commando's in het vers geopende venster. We voegen wat toe en we veranderen wat en klaar is het script. Stapsgewijs:

1. Open een `ScriptEditor` door in het `Control`-menu op `New script` te klikken. Begin met een schone lei door in de `ScriptEditor` onder het `Edit`-menu op `Clear history` te klikken.

2. Ga terug naar het `Object`-venster, kies `Create Sound...` uit het `New`-menu en druk op de `OK`-knop.

3. Kies het `Play`-commando.

4. Kies het `Remove`-commando.

5. Kies nu in de `ScriptEditor` onder het `Edit`-menu de actie `Paste history`. Je hebt nu 3 regels text in de editor staan: de eerste regel begint met `Create` en de laatste regel is `Remove`.

6. Nu kun je de tekst in het venster zo aanpassen dat het er uiteindelijk uitziet zoals hieronder weergegeven (zonder de nummers aan het begin natuurlijk).

```
1  # djmw 20000914
2  #
3  form Speel een kort toontje
4    positive Frequentie_(Hz) 100.0
5  endform
6  Create Sound... kanweg 0 0.5 22050 sin(2*pi*'frequentie'*x)
7  Play
8  Remove
```

Enkele opmerkingen over dit scriptje.

- Je kunt het script uitvoeren met het `Run`-commando in de `ScriptEditor`. Na het klikken op de `OK`-knop hoor je dan een toontje.

- De eerste twee regels die beginnen met het tekentje "#" zijn commentaar. Je mag ook het uitroepteken, "!", of het puntcommateken, ";", als commentaarteken gebruiken. Het is handig om de datum waarop je het script gemaakt hebt ook bij het commentaar te zetten.

- De regels 3, 4 en 5 zorgen ervoor dat er een formuliertje op het scherm komt met als titel `Speel een kort toontje`.

- De text bij het invulveld wordt `Frequentie (Hz)` en de standaardwaarde `100` is dan al vooringevuld. Bij het op het scherm zetten worden de "_"-tekens vervangen door spaties.

- De variabele naam die in de rest van het script gebruikt kan worden, en bij dit invulveld hoort, is een gefilterde versie van de text die bij het veld hoort: de hoofdletter aan het begin wordt omgezet in een kleine letter en de tekst tussen haakjes wordt er met haakjes en het voorafgaande "_"-teken af gehaald. Dus de variable `frequentie` hoort bij het invulveld `Frequentie_(Hz)`.

- Het type van dit invulveld is `positive`. Als de gebruiker een waarde invoert die kleiner is dan nul, dan wordt er automatisch een foutmelding gegenereerd.

- De betekenis van regel 6 is als volgt: maak een nieuw geluidje dat de naam `kanweg` krijgt en waarvan het begin- en eindtijdstip op respectievelijk 0 en 1 seconde liggen. Het geluidje wordt beschreven met de volgende functie van x: $f(x) = sin(2 \cdot \pi \cdot frequentie \cdot x)$. De functie $f(x)$ beschrijft het "analoge" signaal. Het `Sound`-object wordt hieruit gemaakt door deze functie op tijdstippen, die $\frac{1}{22050}$ seconde van elkaar liggen, voor 11025 verschillende waardes van x uit te rekenen. We bemonsteren als het ware het analoge signaal $f(x)$ met een bemonsteringsfrequentie van 22050 Hz.

- In de formule, op regel 6, moet `frequentie` tussen enkele aanhalingstekens "'' staan, terwijl dit niet geldt voor `pi` en `x`. De reden hiervoor is dat de formule voor het geluid na substitutie van de *waarde* van `frequentie` doorgegeven wordt aan een speciale methode, `Matrix_formula`, die de formule evalueert voor elk monster in het `Sound`-object. `Matrix_formula` kent alleen de constante variabele `pi` en variabelen die te maken hebben met rij- en kolomindexering zoals `x`, `y`, `col` en `row`.

2.3 Het tweede script: Fourier-synthese

We gaan een knop maken in het `New`-menu met als titel `Create Sound uit blokgolf...`. Het aanklikken van deze knop moet als gevolg hebben dat er een formulier op het scherm verschijnt waarin gevraagd wordt naar het aantal componenten. Na het aanklikken van de `OK`-knop verschijnt er dan een nieuw `Sound`-object in het `Object`-venster met als naam `blok`. Dit object bevat het resultaat van het optellen van een aantal basistoontjes waarvan de frequenties een harmonische relatie hebben, dit wil zeggen dat deze frequenties allemaal gehele veelvouden zijn van een zelfde basisfrequentie. Deze basisfrequentie wordt ook wel *grondfrequentie* genoemd en is in dit voorbeeld gelijk aan 100 Hz gekozen. De amplitudes van de individuele toontjes hebben we zo gekozen dat ze zo goed mogelijk een blokgolf benaderen. De formule voor een blokgolf is:

$$blokgolf(x) = \sum_{k=1}^{N} \sin(2\pi(2k-1)f_0 x)/(2k-1)$$

Het volgende script moet het doen:

```
1  # blokgolf-synthese
```

```
2  # djmw 20000917
3
4  form Maak een blokgolf
5    natural Aantal_componenten 6
6  endform
7
8  Create Sound... blok 0 1 22050 0
9  fnul = 100
10 for k to aantal_componenten
11   Formula... self+sin(2*pi*(2*'k'-1)*'fnul'*x)/(2*'k'-1)
12 endfor
```

Enkele opmerkingen over het script:

- Ook nu weer drie verschillende representaties van de tekenreeks `Aantal_componenten`: "`Aantal_componenten`", "`aantal_componenten`" en "Aantal componenten".
- In regel 5 dwingt de typering `natural` af dat de invoer een natuurlijk getal (1, 2,...) is.
- Deze toekenning van de waarde `100` aan de variabele `fnul` maakt de formule op regel 11 iets duidelijker. Het biedt ook al gelijk de mogelijkheid om het script zo uit te breiden dat we ook deze frequentie variabel kunnen maken.
- Regel 10 is equivalent aan het explicietere `for k from 1 to aantal_componenten`.
- In regel 11 voert `Formula...` de, hierop volgende, formule uit op alle monsterwaardes van het geselecteerde object. Het impliceert dus een loop, waarbij `self` refereert aan het geselecteerde object. Deze regel in woorden: de nieuwe waarde van elk monster in het `blok Sound`-object wordt gelijk aan de som van de oude waarde van het monster, `self`, plus de waarde van de functie $\sin(2\pi(2k-1)f_0x)/(2k-1)$ op dit bemonsteringstijdstip.

2.4 Het toevoegen van een knop aan een menu

Bij het toevoegen van een nieuwe knop hebben we de keuze tussen toevoegen aan het vaste menu of toevoegen aan het dynamische menu. Toevoegen aan het dynamische menu is voor bovenstaand script niet logisch omdat we én een nieuw object maken én het script niet willen uitvoeren op een geselecteerd object: immers, het

dynamische menu is leeg als er geen geselecteerd object is. De logische plaats is daarom onder het `New`-menu. Het toevoegen van een knop aan het menu kan door in de `ScriptEditor` onder het `File`-menu de optie `Add to fixed menu...` te kiezen. In het formulier kiezen we natuurlijk `Objects` als `Window`-keuze en `New` als het menu waar de knop moet komen. In het `Command`-veld vullen we in `Create Sound uit blokgolf...`. Voor `After command` kiezen we `Create Sound...`. Dit zit één laag diep, daarom kiezen we `1` voor `Depth`. Als je het script al bewaard hebt op schijf dan is de bestandsnaam al voor je ingevuld. Anders staat er `(please save your script first)` en kun je het alsnog bewaren.

De knop is nu toegevoegd aan het `New`-menu van het `Object`-venster. Deze knop blijft bewaard, ook als je *praat* verlaat.

2.5 Het weghalen van een knop van een menu

Wanneer je de functie die een, al dan niet toegevoegde, knop levert niet meer nodig hebt kun je de knop verwijderen met de `ButtonEditor`. Deze is te vinden onder de optie `Buttons...` van de optie `Preferences...` in het `Control`-menu. Selecteer eerst de `Objects`-knop en scroll dan door totdat je een regel vindt die inspringt: `ADDED` Objects: New: `Create Sound uit blokgolf...`, after `Create Sound...`, script `de-bestandsnaam-van-het-script`.

Als je op `ADDED` klikt verandert deze tekst in `REMOVED` en is tegelijkertijd de knop verdwenen uit het menu in het interface.

2.6 Scripten voor batchverwerking

Hoofdstuk 3

Klasse Sound

3.1 Inleiding

De klasse Sound is de belangrijkste klasse binnen het programma *praat*. Verreweg de meeste analyses die gebruikers uitvoeren beginnen met een object van dit type. Een Sound-object bevat een bemonsterd signaal. Dit signaal zou een stuk spraak van een man, een vrouw of een kind kunnen zijn. Het kan ook via een wiskundige formule gegenereerd zijn. Nog weer andere mogelijkheden zijn de ultrasone vocalisaties van een dolfijn of een vleermuis. Zelfs het subsoon geluid van een olifant past in een Sound-object. Al deze geluiden kunnen met *praat* geanalyseerd worden, het maakt voor het programma niet uit wat de oorsprong van een Sound-object is. Dit wil overigens niet zeggen dat alle analysemethodes zomaar toegepast kunnen worden op dit grote scala van Sound-objecten. De standaardinstellingen van de meeste analyses zijn zo gekozen dat ze voor de analyse van het *spraak*signaal van een *vrouw* redelijk optimaal zijn. De vrouw is de "gemiddelde" spreker. Voor analyses op geluiden van andere oorsprong zullen hoogstwaarschijnlijk de standaardinstellingen niet goed zijn en aangepast moeten worden. Wij zullen ons in dit boek hoofdzakelijk bezig houden met het spraakgeluid van mensen. Daarbij zullen we de termen *audio*, *geluid* en *signaal* vaak door elkaar gebruiken voor de aanduiding van de representatie van de inhoud van een Sound-object.

3.2 Attributen van een Sound

De attributen van het Sound-object bevatten die essentiële informatie die we nodig hebben om het geluid te analyseren of hoorbaar te maken. Deze attributen zijn:

$x_{min} \in R$ De begintijd van het geluid.

x_{max} De eindtijd van het geluid. Deze moet altijd groter zijn dan de begintijd. Als we maar één attribuut, *duur*, zouden hebben, in plaats van de twee attributen begintijd en eindtijd, dan verliezen we de mogelijkheid om geluiden op een bepaald moment te laten beginnen. Wanneer we een `Sound`-object uit een ander `Sound`-object gekopiëerd hebben is het handig om de begin- en eindtijd te bewaren.

$n_x \in N$ Het aantal monsters in het geluid.

$dx \in R^+$ De bemonsteringstijd, de inverse van de bemonsteringsfrequentie.

x_1 Het tijdstip van het eerste monster. Meestal is dit voor een geluid gelijk aan $x_{min} + dx/2$.

$z_{1,1..n_x}$ De monsterwaardes. Elke waarde is weergegeven als een reëel getal dat vier bytes in het geheugen van de computer inneemt. Als we het geluid zonder vervorming hoorbaar willen maken dan moet elke waarde z_i in het open interval $-1 < z_i < 1$ liggen.

3.3 Hoe maken we een `Sound`-object

We kunnen een `Sound`-object maken volgens één van de methodes die besproken zijn in paragraaf 1.4. Voor het maken van professionele opnames is een opnamestudio vereist. Voor het maken van opnames die niet aan de hoogste kwaliteitseisen hoeven te voldoen kunnen we de opnamemogelijkheid aan de computer zelf gebruiken. Voor het opnemen van een geluidje in *praat* via een microfoontje, gebruiken we de `SoundRecorder` die de vinden is onder de `Record Sound...` optie van het `New`-menu. De kwaliteit van deze opnames is natuurlijk afhankelijk van een aantal factoren. Om er voor te zorgen dat het invoeren in de computer zo goed mogelijk uitgevoerd wordt, moeten we op een aantal zaken letten.

microfoon De eerste schakel in de opnameketen. De (goedkope) microfoons die rechtstreeks met de ingang van de geluidskaart verbonden zijn garanderen in het algemeen een redelijke geluidskwaliteit. Belangrijk is hier dat de electrische eigenschappen van de microfoon aansluiten bij die van de geluidskaart. Plofklanken geven bij deze types microfoon vaak grote uitschieters in het signaal. Vaak stemt het nulnivo van de micofoon niet overeen met het nulnivo van de geluidskaart.

ingangsnivo het ingangsnivo moet voldoende zijn om zo optimaal mogelijk gebruik te maken van het ingangsbereik van de geluidskaart.

mixer De mixer regelt de geluidsinvoer en -uitvoerstromen. Bij opname is het in het algemeen aan te bevelen om het invoernivo van invoerstromen die niet gebruikt worden zo laag mogelijk te zetten.

achtergrondlawaai Dit wordt meestal mee opgenomen (ventilators van de computer, verkeerslawaai etc.)

3.4 Intermezzo: het digitaliseren van een geluid

De geluiden in de wereld om ons heen zijn van een *analoog* karakter. Een analoog signaal is op elk tijdstip tussen zijn begin- en eindpunt, gedefiniëerd. Wanneer we een geluidssignaal met de computer willen bewerken dan zullen we dit signaal eerst in een voor de computer aangepaste *digitale* vorm moeten brengen. Een aantal termen, met hun Engelse equivalent, die een rol spelen bij geluidsinname en -uitgifte via een computer zijn: *geluidskaart* (sound card), *bemonsteringsfrequentie* (sampling frequency, sample rate) of *bemonsteringstijd* (sampling interval), *kwantisatie* (quantization), *kanalen* (channels), *compressie* (compression), *bestandsgrootte* (file size) en *bestandstype* (file type).

3.4.1 De geluidskaart

De geluidskaart is het medium tussen het geluid in de computer en buiten de computer. In de, voor de computer geschikte, digitale vorm wordt een signaal gerepresenteerd door een rijtje getallen. De conversie van een analoog signaal naar een digitaal signaal gebeurt door een apparaat dat een Analoog-Digitaal Converter (ADC) genoemd wordt: het analoge signaal wordt door een ADC *bemonsterd* en *gekwantiseerd*. Het omgekeerde proces, het omzetten van een digitaal signaal naar een analoog signaal, kan gebeuren met een Digitaal-Analoog Converter (DAC). Vaak zijn deze twee apparaten geïntegreerd op één kaart in de computer, de zogenaamde geluidskaart.

3.4.2 De bemonsteringsfrequentie

Bij het invoeren van een analoog signaal in de computer wordt het analoge signaal eerst *bemonsterd* en daarna *gekwantiseerd*. De bemonsteringsfrequentie bepaalt in hoeveel stukjes elke seconde van het analoge signaal opgedeeld wordt. Van elk klein stukje wordt dan de (gemiddelde) amplitude gemeten en onthouden door de computer. Voor een mono signaal van audio-CD kwaliteit is de bemonsteringsfrequentie 44100 Hz. Dit houdt in dat elke seconde van het geluid dan opgedeeld

is in 44100 stukjes. Het resultaat van de bemonstering is dat de computer gevoed wordt met een reeks van 44100 getallen per seconde.

Het getal 44100 voor de bemonsteringsfrequentie is bedacht door de muziekindustrie, het had net zo goed een ander getal kunnen zijn. Het precieze getal is ook niet van belang als het maar groter is dan ongeveer 2×20000 Hz. De allerhoogste frequentie die mensen nog kunnen horen ligt in de buurt van de 20000 Hz. De muziekindustrie heeft 44100 gekozen voor audioCD's, en 48000 Hz voor DAT-recorders, om in eerste instantie het kopiëren van audioCD's van en naar DAT-tape moeilijker te maken.

Wat is de relatie tussen deze 44100 en het signaal waar het om gaat, het muzieksignaal of het spraaksignaal? Volgens het bemonsteringstheorema van Shannon (zie paragraaf A.10) is het gedigitaliseerde signaal een getrouwe afspiegeling van het analoge signaal als we er voor hebben gezorgd dat de bemonsteringsfrequentie minstens twee keer zo hoog is als de bandbreedte van het analoge signaal. Als we even aannemen dat de laagste frequentie in het analoge signaal in de buurt van de 0 Hz ligt dan wordt de bandbreedte bepaald door de hoogste frequentie die in het signaal voorkomt.

> Een simpele vuistregel voor geluidssignalen is dat de bemonsteringsfrequentie minstens twee keer zo hoog moet zijn als de hoogste frequentie die in het signaal voorkomt.

Wanneer we de bemonsteringsfrequentie niet aan het signaal kunnen aanpassen dan moeten we het signaal aan de bemonsteringsfrequentie aanpassen door er voor te zorgen dat frequenties die hoger zijn dan de helft van de bemonsteringsfrequentie uit het signaal *weggefilterd* worden vóórdat we gaan digitalizeren. Als niet aan de voorwaarde van Shannon is voldaan dan is het digitale signaal geen betrouwbare afspiegeling meer van het analoge signaal. We hebben dan te maken met *vouwvervorming* (aliasing) in het digitale signaal. Figuur 3.1 maakt dit hopelijk duidelijk. In deze figuur wordt gerekend met een fictieve bemonsteringsfrequentie van 8 Hz om zaken te verduidelijken. De absolute waarde van deze frequentie is niet van belang, het gaat meer om de relatie tussen de bemonsteringsfrequentie en de frequentie van het te bemonsteren signaal. In (a) zien we één periode van een sinusvormig signaal van 1 Hz weergegeven. Dit signaal wordt 8 maal per seconde bemonsterd en de resulterende monsterwaardes als functie van de tijd zijn weergegeven in (b). In (c) hebben we te maken met een analoog signaal van 9 Hz. De bemonsterde versie zien we in (d). Wanneer we naar (d) kijken dan zien we dat (d) veel lijkt op (de omgeklapte versie van) (b) en dat we zijn oorsprong, het analoge signaal (c), hierin helemaal niet herkennen. Als we in (d) de paaltjes met een vloeiende curve zouden verbinden dan zou dit een (in fase) verschoven versie van signaal (a) opleveren. Het lijkt erop alsof de frequentie van het digitale signaal lager geworden is. Hebben we een truc gevonden om de frequentie van een signaal

te verlagen?! Bij het digitalizeren is dit een ongewenste situatie: uit de monsterwaardes is immers niet meer te achterhalen van welke frequentie-componenten ze afkomen. Het volgende script maakt het effect van aliasing hóórbaar:

```
1  # aliasing hoorbaar gemaakt
2  # 20000919 djmw
3
4  fs = 11025
5  f1 = 1000
6  f2 = 4 * fs + 500
7  Create Sound... s1 0 1 'fs' sin(2 * pi * 'f1' * x)
8  Play
9  Create Sound... s2 0 1 'fs' sin(2 * pi * 'f2' * x)
10 Play
11 Create Sound... s3 0 1 'fs' (sin(2 * pi * 'f1' * x) +
12    ... sin(2 * pi * 'f2' * x)) / 2
13 Play
```

Hierbij hebben we weer gebruik gemaakt van het feit dat de analoge functies, in de regels 7, 9 en 11, bemonsterd worden op intervallen $1/fs$. De frequentie van het analoge signaal `s2` is 44600 Hz en kan normaal niet eens door mensen gehoord worden. Omdat we dit signaal niet goed bemonsteren horen we het als een toontje van 500 Hz. In regel 11 tellen we de twee signalen `s1` en `s2` op. We delen door 2 om er voor de zorgen dat de amplitude van `s3` altijd tussen -1 en $+1$ ligt. Ook hier kiezen we weer bewust een te lage bemonsteringsfrequentie. Wanneer we zouden luisteren naar het analoge geluidje $s3(x) = (\sin(2\pi f_1 x) + \sin(2\pi f_2 x))/2$ dan zouden we alleen een toontje van 1000 Hz horen, de component $\sin(2\pi f_2 x)$ is immers veel te hoog om te horen. Door onze foute bemonstering hebben we een frequentie-component van 500 Hz geïntroduceerd die helemaal niet in het oorspronkelijke signaal zat. Dit is natuurlijk een bijzonder kwalijke zaak.

We kunnen er voor zorgen dat vouwvervorming niet kan optreden door het signaal van te voren te *filteren*. We kunnen dan de bandbreedte van het signaal in overeenstemming brengen met de bemonsteringsfrequentie die we kiezen. Voor spraaksignalen betekent filteren dat we alle frequenties die groter zijn dan de helft van de bemonsteringsfrequentie zoveel mogelijk moeten verzwakken. Deze speciale frequentie, de helft van de bemonsteringsfrequentie, noemt men ook wel de *Nyquist-frequentie*.

Filter frequentiecomponenten die hoger zijn dan de Nyquist-frequentie uit het signaal vóórdat er gedigitaliseerd wordt.

3.4.3 De kwantisatie

In de voorgaande paragraaf hebben we het alleen gehad over het bemonsteringsproces in termen van getalletjes die ingevoerd worden in de computer. We gaan het nu over deze getalletjes zelf hebben. In de computer wordt alle informatie, en dus ook getallen, gerepresenteerd met een rijtjes opeenvolgende bits. Een bit kan 2 verschillende toestanden aannemen. Een serie van twee bits kan $2 \times 2 = 4$ verschilllende toestanden aannemen. Een serie van N bits kan 2^N toestanden aannemen. Er geldt altijd: hoe meer bits hoe meer (potentiële) precisie. De precisie waarmee de amplitude van het analoge signaal kan worden benaderd hangt af van het aantal bits dat de ADC gebruikt voor de kwantisering.

Er zijn heel veel vormen van kwantisatie, die we in twee grote categoriën kunnen verdelen: *uniforme* en *niet-uniforme* kwantisatie. Bij uniforme kwantisatie wordt het bereik van de amplitude van het analoge signaal opgedeeld in een gelijk aantal stapjes. Bij de niet-uniforme kwantisatie is dit niet zo. Meestal worden hier de kleine amplitudes met meer stapjes benaderd en de grote amplitudes met minder. We gaan van elk van deze twee categoriën een methode bespreken: lineaire kwantisatie en μlaw kwantisatie. We bespreken er enkele en beginnen met de meest simpele en meest gebruikte:

Lineaire kwantisatie

Lineaire kwantisatie is de simpelste en meest gebruikte kwantisatie, bijna alle standaard geluidskaarten doen aan lineaire kwantisatie. Figuur 3.2 laat zien wat er met de amplitude van een sinusvormig signaal gebeurt als het aantal bits dat ons ter beschikking staat varieert. De figuur laat duidelijk zien dat als we meer bits tot onze beschikking hebben, de benadering van het analoge signaal steeds beter wordt.

ADPCM kwantisatie

μlaw kwantisatie

Deze kwantisatie hoort tot de groep van niet-uniforme kwantisaties en wordt ook gebruikt in de telefonie in de VS en Japan. De kwantisatie is met 8 bit en gaat volgens een logarithmische curve:

$$y(x) = sign(x) x_{max} \frac{ln(1 + \mu|\frac{x}{x_{max}}|)}{ln(1 + \mu)}$$

We merken het volgende op over deze formule:

- x_{max} is de maximale waarde die de amplitude van het analoge signaal x kan aannemen.

- De functie $sign(x)$ volgt het teken van x en is daarom gelijk aan 1 voor $x > 0$ en gelijk aan -1 voor $x < 0$.
- Omdat $|\frac{x}{x_{max}}| \leq 1$ is de maximale $y(x)$ ook gelijk aan x_{max}.
- De meest gebruikte waardes van μ zijn 100 en 255.

In figuur 3.3 kunnen we het verschil tussen μlaw en lineaire kwantisatie zien.

A-law kwantisatie

Deze kwantisatie is net als de μlaw-kwantisatie met 8 bits precisie. Hij wijkt in zoverre af dat de kleine amplitudes lineair zijn. In formulevorm

$$y(x) = \begin{cases} \frac{A}{1+x\ln A}, & \text{als } |\frac{x}{x_{max}}| \leq \frac{1}{A} \\ \frac{sign(x)x_{max}}{1+\ln A}(1+\ln A|\frac{x}{x_{max}}|), & \text{als } \frac{1}{A} \leq x \leq 1 \end{cases}$$

3.4.4 Het aantal kanalen

3.4.5 De compressie

3.4.6 De bestandsgrootte

3.4.7 Het bestandstype

In de computerwereld zijn een groot aantal bestandstypes in omloop. Er zijn bestandstypes voor audio, video, tekstverwerkers, spreadsheets, presentatieprogramma's en zo voorts. Alleen voor audio-bestanden zijn er al tientallen verschillende bestandstypes[1] Op veel computersystemen (o.a. Windows en Unix-varianten) wordt het type van een bestand kenbaar gemaakt door een *bestandsextensie* (file extension). Voorbeelden hiervan onder Windows zijn bijvoorbeeld de extensie `.doc` voor een bestand dat gegevens bevat voor het programma Word en de extensie `.ppt` voor Powerpoint bestanden.

De meeste moderne bestandsformaten zijn *zelfbeschrijvend* (we zullen bestandsformaat en bestandstype losjes door elkaar gebruiken). Zelfbeschrijvend betekent dat het bestand behalve de data genoeg informatie bevat (meta data), om zonder externe kennis gelezen te kunnen worden. Stel we hebben een audiobestand waarvan we alleen weten dat het de monsterwaardes bevat en geen verdere informatie. Dit bestand bestaat voor de soundfilelezer[2] in *praat* alleen maar uit een

[1] Zie bijvoorbeeld het Audio File Formats FAQ op `http://home.sprynet.com/sprynet/cbagwell/audio.html`.

[2] Een *soundfilelezer* is een programmaonderdeeltje dat als invoer een bestand op schijf nodig heeft en als uitvoer een Sound-object levert. Elk type audiobestand heeft zijn eigen soundfilelezer.

continue reeks bytes. Om deze bytes om te kunnen zetten naar een Sound-object moet de soundfilelezer nog het volgende weten:

kwantisatie Hoeveel bytes moeten er samen genomen worden om één monsterwaarde te krijgen? Populaire formaten nemen bijvoorbeeld 1 byte (8 bit) en interpreteren dit als een geheel getal tussen -256 en +255.
Wanneer we 2 byte nemen (16 bit) en deze combinatie als getal met teken interpreteren dan kunnen de monsterwaardes tussen -32768 en $+32767$ liggen. Bij deze 2 bytes kunnen we nog kiezen hoe we bytes aan elkaar plakken om hiervan één twee-byte getal te maken: komt het eerst gelezen byte vòòr- of áchteraan? In de Engelse terminologie komen we dan de termen *big-endian* en *little-endian* tegen. Deze keuze is vaak afhankelijk van het computerplatform. Computers met Intel-processors zijn little-endian, die met MIPS-processors zijn big-endian. De POWERPC-processorarchitectuur kent zowel het big- als het little-endian spelletje.
Met vier bytes hebben we nog meer keuzes: vormen de 4 bytes één geheel getal (integer) of een reëel getal (real). Is het gehele getal big- of little-endian? Is de real *fixed point* of *floating point*?

kanalen Is het geluidje mono of stereo, of bevat het nog meer kanalen? Als het meerkanaals is en bijvoorbeeld stereo, hoe is dan de verdeling van de monsterwaardes over de kanalen: komen eerst alle monsters van het ene kanaal en dan alle monsters van het andere kanaal? Of zitten de monsterwaardes om en om in het bestand? Is dan het eerste monster dat van het linkerkanaal of van het rechterkanaal?

bemonsteringsfrequentie Hoe veel monsterwaardes zijn er per tijdseenheid? Dit is o.a. voor de uitgifte van het geluid belangrijk.

compressie Is het bestand gecomprimeerd? Zo ja, met welk algoritme. Er zijn heel veel compresie-algorithmes ontwikkeld. Ze vallen uiteen in twee groepen:

- verliesloze compressie waarbij het origineel exact gereconstrueerd kan worden uit het gecomprimeerde. Voorbeeld van verliesloze compressie-algoritmes zijn `FLAC`[3] en `shorten`[4]. Een compressiefactor van 2:1 is net haalbaar.

[3] Met broncode, zie `http://flac.sourceforge.net/`
[4] Zonder broncode, zie `http://www.softsound.com/Shorten.html`

- compressie met verlies. Voorbeelden zijn `LAME`[5], `mp3`[6] en `ogg-vorbis`[7]. Afhankelijk van het getolereerde verlies kan de compressie aanzienlijk zijn (10:1).

Uit bovenstaande opsomming blijkt dat, om fouten te voorkomen, het in het algemeen handig is als deze informatie in het audio-bestand zelf te vinden is. De manier waarop deze informatie in het bestand te vinden is, verschilt voor de verschillende audiobestandstypes.

Enkele extensies die in de audiowereld gebruikt worden en werden zijn:

.aiff Het **A**udio **I**nterchange **F**ile **F**ormat. Het standaard audio-bestandsformaat op de Apple Macintosh computers en op de grafische werkstations van Silicon Graphics. Big-endian formaat.

.aifc Het **A**udio **I**nterchange **F**ile with **C**ompression. Als AIFF maar met mogelijkheden voor compressie.

.wav Ongeveer een (overbodige) replica van AIFC maar dan in Little-endian formaat. Tegenwoordig het meest gangbare audioformaat.

.voc Soundblaster files

.au Sun audioformaat.

.mp3 Niet-verliesloze compressie die gebruik maakt van bekende kenmerken van het oor. Dit formaat is minder geschikt voor spraaksignaalanalyse omdat er de compressie niet-verliesloos is en er, voor het oor moeilijk hoorbare, vervorming van het oorspronkelijke signaal is opgetreden.

.ogg Ogg Vorbis audio. Maakt net als `mp3` gebruikt van de eigenschappen van het oor bij compressie. In tegenstelling tot `mp3` is het compressie en decompressie algorithme, de zogenaamde `codec`, voor iedereen vrij beschikbaar.

3.5 Afwijkende bemonsteringsfrequenties

Bij het afspelen van een geluid moeten we bedenken dat er een relatie is tussen de bemonsteringsfrequentie van het geluid, de geluidskaart en de kwaliteit van het uitgevoerde geluid. De meeste geluidskaarten kunnen niet zomaar een digitaal geluid met een willekeurige bemonsteringsfrequentie analoog maken. Vaak kan

[5]Met broncode, zie `http://www.mp3dev.org/mp3/`

[6]Zie `http://www.iis.fhg.de/amm/techinf/layer3/`

[7]Uit hun folder ”a completely open, patent-free, professional audio encoding and streaming technology”. Zie `http://www.vorbis.com/`

alleen een bemonsteringsfrequentie gekozen worden uit de reeks 44100, 22050, 11025 Hz en/of de reeks 48000, 24000, 16000, 12000, 8000 Hz. In de eerste reeks komen alleen delers voor van 44100 Hz, de bemonsteringsfrequentie van audioCD's. De getallen in de tweede reeks zijn allen delers van 48000 Hz, de bemonsteringsfrequentie van Digitale Audio Tape's (DAT).

Om `Sound`-objecten met een andere dan deze, door de hardware ondersteunde, bemonsteringsfrequentie hoorbaar te maken, voert *praat* een simpele herbemonstering met een hogere bemonsteringsfrequentie uit. Deze nieuwe frequentie is de dichtstbijzijnde, hogere, door de kaart ondersteunde. De amplitudes op de nieuwe monsterpunten worden via lineaire interpolatie uit de oude amplitudes berekend. Deze methode werkt snel, maar is niet erg precies en introduceert daarom ruis in het geluid. De beste methode is een herbemonstering op de nieuwe bemonsteringsfrequentie via een exacte interpolatie.

> Als een `Sound`-object bemonsterd is met een frequentie die niet door de geluidskaart ondersteund wordt, voer dan een herbemonstering uit met het commando `Resample...`.

Je kunt dit testen met het volgende script door te luisteren of je verschil hoort tussen beide geluidjes.

```
1  Create Sound... s_10000 0 1 10000 sin(2*pi*1000*x)
2  Create Sound... s_11025 0 1 11025 sin(2*pi*1000*x)
3  plus Sound s_10000
4  Play
```

3.6 Sound: Scale...

Om een `Sound`-object zonder vervorming hoorbaar te maken in *praat* moeten alle amplitudes liggen in het open interval $(-1, +1)$. We noemen het signaal *maximaal uitgestuurd* als de maximale amplitude ± 1 is. Het gebeurt vaak dat van een stuk spraaksignaal de amplitudes niet maximaal uitgestuurd zijn. Dit hoeft meestal voor de meetprocedures geen probleem te zijn. Toch kan het soms nodig zijn om het signaal maximaal uit te sturen. We kunnen hiervoor bij een geselecteerd `Sound`-object het commando `Scale...` gebruiken. Dit commando heeft één parameter, de gewenste (absolute waarde van de) maximale amplitude van het geschaalde signaal. De standaardinstelling is zo gekozen dat een maximaal uitgestuurd met 16 bit gekwantiseerd signaal net niet vervormd wordt door de geluidskaart[8]

[8] Soms worden amplitudes, die voor 16 bit gekwantiseerd zijn en precies de extreme waardes -32768 en $+32767$ hebben, door de logica van de kaart omgekeerd. Dit veroorzaakt heel storende bijgeluiden.

- Voor betere visualisatie. Wanneer we het signaal bijvoorbeeld willen labelen door het samen met een `TextGrid`-object te selecteren en het `Edit`-commando te kiezen, dan hoeven we geen ”autoscaling” optie te gebruiken. Ook als we een plaatje willen maken in het `Picture`-venster is een goed uitgestuurd signaal beter.

- Voor betere opslag. Wanneer we het signaal gekwantiseerd opslaan in bijvoorbeeld een `aiff`- of `wav`-bestand, is het altijd aan te raden om eerst het signaal maximaal uit te sturen[9]. Op deze manier verliezen we de minste informatie.

- Voor betere hoorbaarheid als we onszelf de moeite willen besparen van het telkens opnieuw afregelen van het volume van de geluidskaart.

3.7 Intermezzo: Codeerkwaliteit

In O’Shaughnessey (1987, hfdstk 7) wordt een overzicht gegeven van het coderen van spraaksignalen. Alhoewel wij ons alleen met PCM-gecodeerd materiaal bezighouden is het toch nuttig wat over andere technieken te zeggen. Codering in het boek van O’Shaughnessey staat hoofdzakelijk in het teken van de transmissie van signalen via een (ruisig) medium. Hierbij spelen dan een aantal aspecten:

transmissiebandbreedte Hoeveel informatie kan er per tijdseenheid worden getransporteerd over het medium. Deze capaciteit wordt gemeten in bits/s.

transmissiekosten Hoeveel kost het om een bepaalde hoeveelheid data te transporteren over het medium.

opslagkosten De kosten van het tussentijds bufferen, waarbij ”tussentijds” variabel is.

algoritme Hoe complex is het codeer-algoritme.

snelheid Kan de codering in werkelijke tijd (real-time) geschieden. Als de codeerder meer bits/s geeft dan de transmissiebandbreedte toelaat, dan zal er niet in werkelijke tijd gewerkt kunnen worden.

spraakkwaliteit Hoe goed is de verstaanbaarheid na de sequentie codering-transmissie-decodering? In het algemeen zal de kwaliteit een monotone functie zijn van het aantal bits/s dat gebruikt wordt voor de codering.

[9] Tenzij we amplituderelaties met andere bestanden willen blijven behouden.

Hoeveel bits/s zijn er nodig voor de codering? Uit metingen aan de verwerkingscapaciteit van mensen blijkt dat deze niet meer dan 50 bit/s aan nieuwe informatie kunnen verwerken. Een audio-CD daarentegen levert aan de computer voor het linker en het rechter kanaal een constante datastroom van 16 (bits) x 44100 (monsters/s) = 705600 bits/s.

3.8 Opgaves

1. Maak een $1/2$ seconde durend simpel toontje met een frequentie van 500 Hz. Bekijk het geluidje in de SoundEditor. Luister naar het geluidje. Selecteer, door in te zoomen, een stuk dat begint en eindigt op een lokaal maximum van de amplitude en beluister deze selectie. Waarom klinkt dit geluid anders?

2. Maak een toontje met ruis. De toon heeft een frequentie van 400 Hz een en amplitude $1/2$. Voeg daarbij additieve standaard normaal verdeelde Gaussische ruis met standaarddeviatie $\sigma = 0.2$ en gemiddelde 0. Je kunt hiervoor de functie *randomGauss (μ, σ)* gebruiken, waarbij μ het gemiddelde voorstelt en σ de standaard deviatie.

3. Pas het script in paragraaf 2.3 zo aan dat er ook naar de *grondfrequentie* gevraagd wordt in het formulier.

4. Maak analoog aan het script in de vorige opgave scripts die een zaagtandfunctie en een driehoeksfunctie benaderen met sinusfuncties.

$$zaagtand(x) = \sum_{k=1}^{N} \sin(2\pi k f_0 x)/k$$

$$driehoek(x) = \sum_{k=1}^{N} (-1)^{k+1} \sin(2\pi(2k-1) f_0 x)/(2k-1)^2$$

Vraag in het formulier zowel naar het *aantal componenten* (N) als ook naar de *grondfrequentie* (f_0).

5. Lineaire kwantisatie.
Maak een Sound-object waarin een signaal $s(t)$ lineair oploopt van -1 tot $+1$. Maak dit signaal bijvoorbeeld 1 seconde lang en neem als bemonsteringsfrequentie 22050 Hz. Kwantiseer de amplitude met N bits/monsterwaarde. Maak op één pagina A4 een plaatje bestaande uit vijf onderdelen:

- Het lineaire signaal $s(t)$.

- Het gekwantiseerde signaal $s_q(t)$, met de amplitudes afgebeeld op het interval $[-2^{N-1}, 2^{N-1} - 1]$.
- Het (geschaalde) gekwantiseerde signaal $s_{qs}(t)$, met amplitudes in het interval $[-1, +1]$.
- De kwantisatiefout $e(t) = s(t) - s_{qs}(t)$.
- De relatieve kwantisatiefout $r(t) = e(t)/s(t)$. Signaal $r(t)$ geeft een probleem met het tekenen: het quotiënt $e(t)/s(t)$ wordt heel erg groot als $s(t)$ klein wordt. Begrens daarom de maximale en minimale amplitudes van $r(t)$. Dit kan bijvoorbeeld op de volgende manier:
  `Formula... if self < -1 then -1 else if self > 1 ... then 1 else self endif endif`

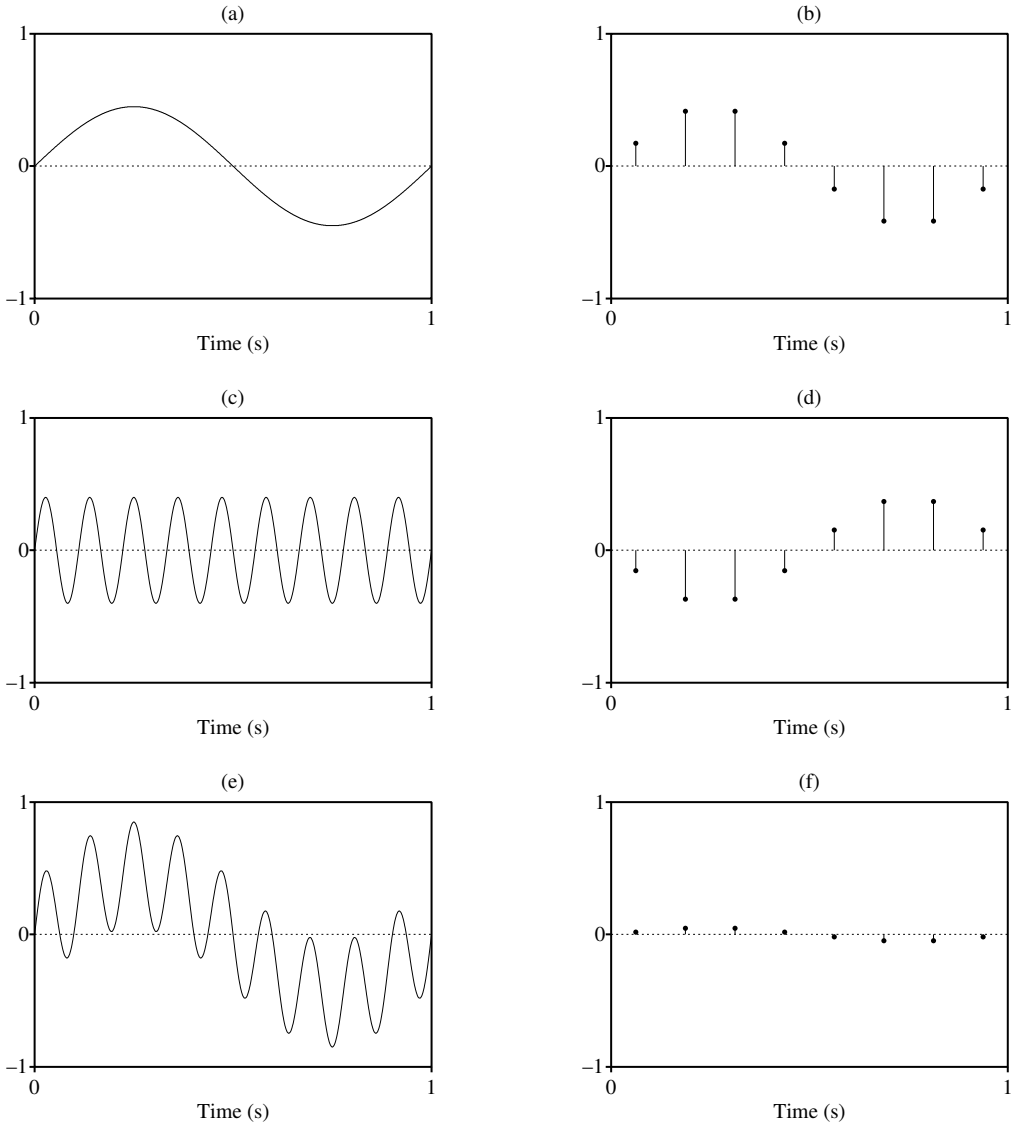
6. μlaw-kwantisatie.

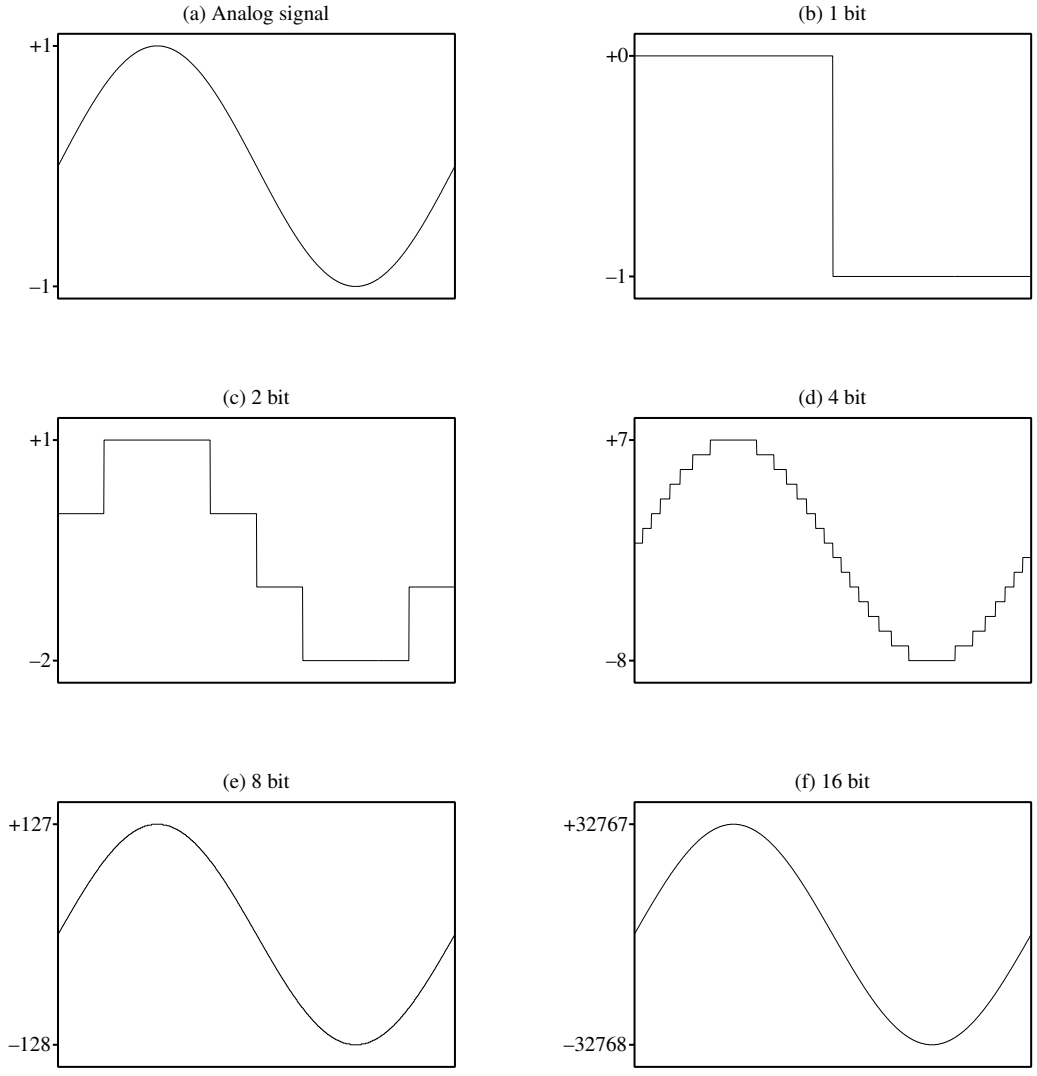
Probeer de μlaw-kwantisatie na te bootsen zoals die beschreven is in paragraaf 3.4.3. Het te kwantiseren signaal $s(t)$ is een signaal dat lineair oploopt van -1 tot $+1$. De formule voor de compressie vereenvoudigt hierdoor omdat $x_{max} = 1$. Vraag de waarde van μ en het aantal bits voor de kwantisatie via een formulier aan de gebruiker. Maak op één pagina A4 een plaatje bestaande uit de volgende zeven onderdelen:

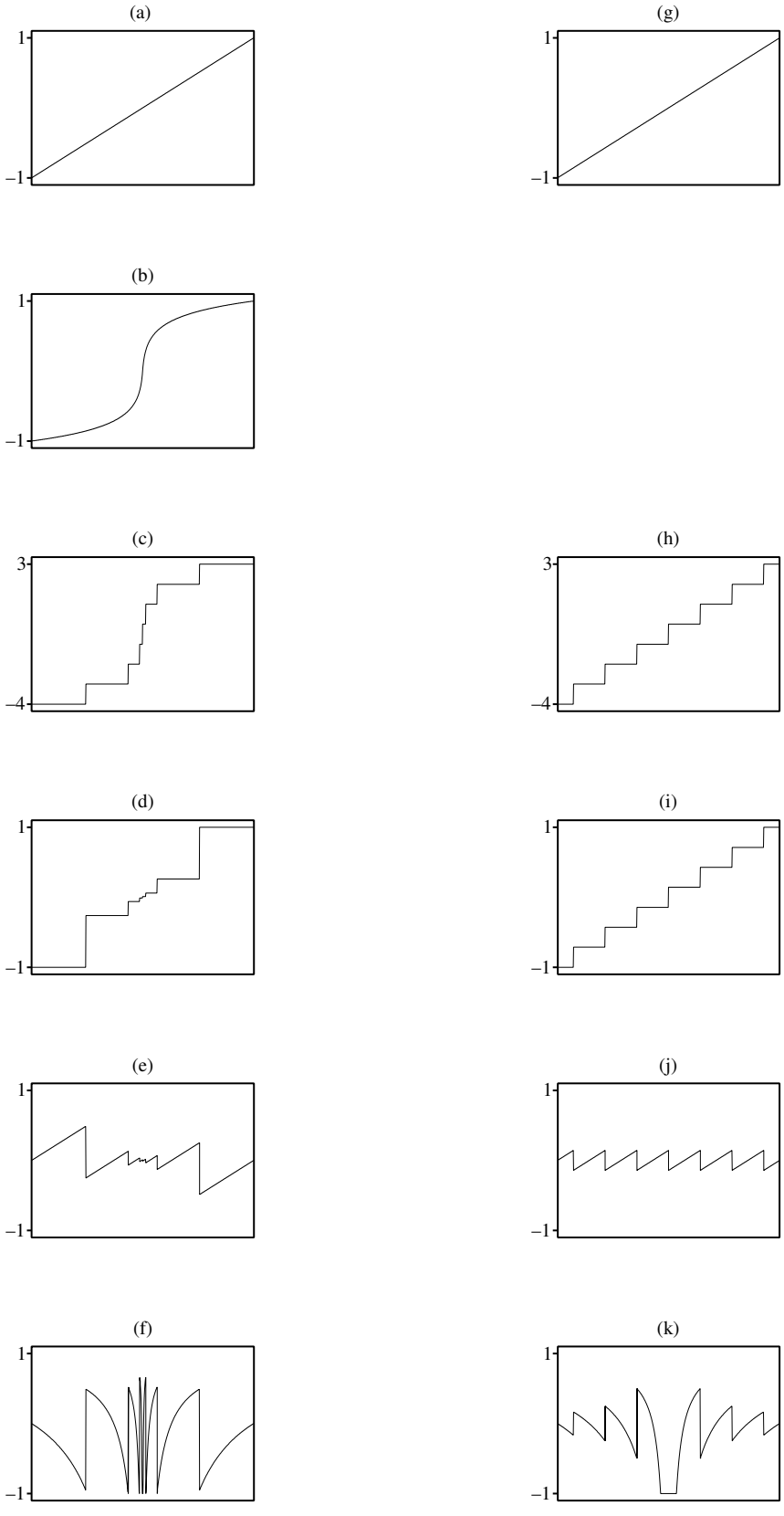
- Het lineaire signaal $s(t)$.
- Het via de μlaw-formule gecomprimeerde signaal $s_c(t)$.
- Het signaal $s_{cq}(t)$, de kwantisatie van $s_c(t)$, met de amplitudes afgebeeld op het interval $[-2^{N-1}, 2^{N-1} - 1]$.
- Het geschaalde gekwantiseerde signaal $s_{cqs}(t)$, met amplitudes in het interval $[-1, +1]$.
- Het geëxpandeerde signaal $s_{cqse}(t)$, de inverse van de compressie.

$$s_{cqse}(t) = sign(s_{cqs}(t))e^{|s_{cqs}(t)| \ln(1+\mu)} - 1)/\mu$$

- De kwantisatiefout $e(t) = s(t) - s_{cqse}(t)$.
- De relatieve kwantisatiefout $r(t) = e(t)/s(t)$. Zie de vorige opgave over de begrensing van amplitudes.


(a)
1
0
–1
0
1
Time (s)
(b)
1
0
–1
0
1
Time (s)
(c)
1
0
–1
0
1
Time (s)
(d)
1
0
–1
0
1
Time (s)
(e)
1
0
–1
0
1
Time (s)
(f)
1
0
–1
0
1
Time (s)


(a) Analog signal
+1
–1
(b) 1 bit
+0
–1
(c) 2 bit
+1
–2
(d) 4 bit
+7
–8
(e) 8 bit
+127
–128
(f) 16 bit
+32767
–32768


(a)
1
–1
(g)
1
–1
(b)
1
–1
(c)
3
–4
(h)
3
–4
(d)
1
–1
(i)
1
–1
(e)
1
–1
(j)
1
–1
(f)
1
–1
(k)
1
–1

Hoofdstuk 4

Klasse Spectrum

4.1 Inleiding

De klasse Spectrum is net als klasse Sound één van de basisklasses in *praat*.

Een `Spectrum`-object bevat het complexe spectrum als functie van de frequentie. Een aantal analyses resulteren in een `Spectrum`-object; de meest gebruikte analyse is `Sound: To Spectrum (fft)`. Bij het filteren van een `Sound`-object wordt vaak een `Spectrum`-object als intermediair gebruikt.

4.2 Attributen van een Spectrum

De attributen zijn die van klasse Matrix, alleen de interpretatie is anders:

$x_{min} \in R$ De beginfrequentievan het spectrum, meestal 0 Hertz.

x_{max} De eindfrequentie van het spectrum in Hertz. Deze moet altijd groter zijn dan de beginfrequentie.

$n_x \in N$ Het aantal frequenties in het spectrum.

$dx \in R^+$ De afstand in Hertz tussen de frequenties.

x_1 De frequentie van de eerste component.

ny Het aantal rijen (altijd 2) voor het reële en imaginaire deel van het spectrum.

$z_{1,1..n_x}$ Het rijtje met het reëele deel van het spectrum.

$z_{2,1..n_x}$ Het imaginaire deel van het spectrum.

4.3 Sound: To Spectrum (fft)

Het algoritme dat uit een `Sound`-object een `Spectrum`-object maakt gebruikt van de Fast Fourier Transform (zie paragraaf A.8). We onderscheiden de volgende stappen hierin:

- Het bepalen van de bufferlengte. De FFT werkt met signalen waarin het aantal monsterwaardes een macht van 2 is. Als het aantal monsterwaardes, n_x, in het `Sound`-object een macht van 2 is, zeg 2^p, dan zijn we klaar met deze stap. Anders zoeken we het getal tussen n_x en $2n_x$ dat een macht van twee is. We kunnen dit opschrijven als: we zoeken een exponent p zo dat

$$n_x \leq 2^p < 2n_x.$$

- We alloceren een buffer van lengte 2^p en kopiëren hierin de n_x monsterwaardes. Als er nog ruimte over is in de buffer (het aantal monsters in het geluid is geen macht van 2), dan maken we het restant van de buffer leeg: we vullen het met nullen.
- We voeren een FFT uit op het signaal in de buffer.
- We selecteren de $2^p/2 + 1$ reële en imaginaire frequenties uit de getransformeerde buffer en kopiëren deze in de rijtjes z_1 en z_2 van het `Spectrum`-object.

Het `Spectrum`-object dat nu verschijnt heeft de volgende waardes voor zijn attributen:

- $x_{min} = 0$
- $x_{max} = \frac{bemonsteringsfrequentie}{2}$
- $n_x = 2^p/2 + 1$
- $dx = \frac{bemonsteringsfrequentie}{2^p}$
- z_1, z_2 De reële en imaginaire delen van het spectrum.

4.4 Spectrum: To Sound (fft)

Hier wordt de inverse operatie van `Sound: To Spectrum (fft)` uitgevoerd. Het algoritme maakt gebruik van de *inverse* FFT. We onderscheiden de volgende stappen:

- Alloceer een buffer van lengte $2(n_{x;s}-1)$, waarbij $n_{x;s}$ het aantal frequenties in het `Spectrum`-object zijn. De bufferlengte is nu weer een macht van 2. Vul deze buffer met het reële en imaginaire deel uit het `Spectrum`-object.
- Doe een inverse FFT
- Kopiëer de getransformeerde buffer naar het (gecreëerde) `Sound`-object.

Het `Sound`-object dat nu verschijnt heeft de volgende waardes voor zijn attributen:

- $x_{min} = 0$
- $n_x = 2(n_{x;s} - 1)$
- $dx = \frac{1}{(n_{x;s}-1)\Delta f}$
- $x_1 = dx/2$
- $x_{max} = n_x d_x$
- z_1 De monsterwaardes.

De cyclus `Sound:To Spectrum (fft)` gevolgd door `Spectrum:To Sound (fft)` geeft het oorspronkelijke geluid weer terug, mogelijk aangevuld met stilte.

4.5 Invloed van signaallengte en FFT op het spectrum

In figuur 4.1 zijn in de linkerkolom een aantal elementaire signalen en in de rechterkolom hun spectra getekend. De `Spectrum`-objecten zijn uit hun corresponderende `Sound`-objecten berekend via de methode van paragraaf 4.3, dus met behulp van het FFT-algoritme. Alle signalen in de linkerkolom zijn gegenereerd met de formule $\sin 2\pi f x$ en verschillen alleen in duur. De waardes voor de bemonsteringsfrequentie (22050 Hz) en de frequentie ($f = 4/1.4860771$ Hz) zijn zo gekozen dat als het signaal 1.4860771 seconde duurt er:

- precies 4 periodes in het signaal zitten,
- het aantal monsterwaardes, 32678, precies een macht van twee is (2^15).

We worden in de figuur geconfonteerd met een raadsel: ondanks het feit dat de signalen (a), (c), (e) en (g), dezelfde frequentie hebben, zien hun spectra er verschillend uit. Er is wel een piek op de ”goede” plaats, maar alleen in figuur 4.1.b lijkt het spectrum uit één frequentie te bestaan. Een deel van het raadsel wordt

opgelost als we bedenken van welk signaal het FFT-algoritme gebruik maakt. Het FFT-algoritme berekent het spectrum van een signaal warvan het aantal monsterwaardes een macht van twee is. Alleen het signaal (a) beantwoordt in eerste instantie aan deze voorwaarde. Bij de andere signalen, die allemaal korter zijn, worden nullen toegevoegd. In figuur 4.2 zijn aan de rechterkant van het plaatje de signalen getekent waarvan de feitelijke Fourier-transformatie wordt berekend via het FFT-algoritme (dit zijn dus de signalen zoals ze in de "FFT-buffer" zitten). Verder moeten we erbij denken dat het onderliggende model ook nog voorschrijft dat het hele signaal waarop de feitelijke Fourier-transformatie plaatsvindt ook nog periodiek voorgezet wordt. Alleen het sinusvormige signaal in (a) is dan "mooi" periodiek, dat wil zeggen dat als we het signaal achter zichzelf plakken de sinus continu doorloopt[1]. Stel we definiëren T als de tijd waarover we nullen hebben toegevoegd in het spectrum. Voor plaatje 4.2.d is dit bijvoorbeeld $T = 1.4860771 - 1.3003175 = 0.1857596$ s. De afstanden in Hertz tussen de zijlobben in de spectra (d), (f), (h) van figuur 4.1 zijn dan $1/T$. Dit geeft respectievelijk 5.4 Hz, 3.6 Hz en 2.7 Hz[2].

Als we naar het spectrum kijken van het signaal in figuur 4.1.f dan zien we dat amplitude van de frequenties bij 50 Hz veel hoger zijn dan die in de andere spectra. Dit komt omdat er een discontinuïteit in het signaal zit (zie ook figuur 4.2.f): het signaal verandert bij $t = 1.3$ s heel plotseling van een grote waarde naar nul. Een snelle verandering in het signaal kan alleen gemaakt worden met basiscomponenten die ook snel kunnen veranderen: dit kunnen alleen sinussen en cosinussen met een hoge frequentie zijn. Deze zijn alleen nodig om de discontinuïteit de modeleren, niet het signaal "zelf". In het algemeen zal, als we het spectrum van een signaal via de FFT bepalen, het signaal nooit "mooi" op nul eindigen. We hebben daardoor meestal een discontinuïteit aan het einde (en/of begin) van het signaal. Deze discontinuïteit zorgt ervoor dat er allerlei ongewenste frequenties in het spectrum komen. Om deze te vermijden gebruiken we, zeker als we lopende spectra gaan bepalen, zoals bij Spectrogrammen, een *venster*. Zie A.12 voor meer informatie hierover.

---

[1] Dit is alleen het geval als er *precies* een geheel aantal periodes in de "FFT-buffer" passen.

[2] Wanneer T groter wordt dan de helft van het signaal dan kunnen we de afstanden tussen de zijlobben berekenen door voor T de duur te nemen waarvoor *geen* nullen zijn toegevoegd. Deze situatie treedt normaal in *praat* niet op omdat de "FFT-buffer" nooit meer dan twee keer zo lang wordt als de duur van het signaal. Natuurlijk is deze situatie kunstmatig te creëren via bijvoorbeeld

```
Create Sound...  s 0 1 11025 if x<0.4 then sin(2*pi*4*x) else 0
fi
```

4.6 Het amplitude spectrum

Wanneer we in *praat* met een geselecteerd `Spectrum`-object de methode `Draw...` kiezen, dan wordt het amplitude-spectrum getekend. De dimensionele eenheid die bij het amplitude-spectrum hoort is dB/Hz. Eerst berekenen we uit het `Spectrum`-object de one-sided-power-spectral-densities a_k uit de reële component $H_{k,r}$ en de imaginaire component $H_{k,i}$ op de volgende manier:

$$a_k = 2|H_k|^2\Delta f = 2(H_{k,r}^2 + H_{k,i}^2)\Delta f \tag{4.1}$$

De factor 2 staat er omdat we de positieve en de negatieve frequenties bij elkaar nemen. De term tussen haakjes, de som van de kwadraten van het reële en imaginaire deel, is de standaard manier om van een complex getal (het kwadraat van) de amplitude te krijgen. De term Δf is de breedte van het frequentie-interval. We bepalen via bovenstaande berekening de oppervlakte van (twee maal) een rechthoekje met "breedte" Δf en het kwadraat van de amplitude als "hoogte".

Omdat de waardes van de a_k's in het algemeen ordes van grootte verschillen, gaan we over naar een logaritmische schaal met referentie, de decibel-schaal. De referentie die we nemen is de gehoordrempel, $4 \cdot 10^{-10} Pa^2$. Dit resulteert dan in de getalletje p_k die we kunnen tekenen:

$$p_k = 10 \cdot \log(\frac{a_k}{4 \cdot 10^{-10}}) \tag{4.2}$$

4.7 Opgaves

1. FFT en bufferlengte.

We gebruiken in deze opgave een bemonsteringsfrequentie van 16000 Hz.

We maken twee signalen s_1 en s_2 met een tijdsduur van onveer 1 s, maar wel zó dat het aantal monsters precies een macht van 2 is.

- Maak in s_1 een toontje met een frequentie van 113 Hz.
- Vul s_2 met 113 periodes van een sinus.

Maak twee signalen s_3 en s_4 die precies 1.01 s duren.

- Maak in s_3 een toontje met een frequentie van 113 Hz.
- Vul s_4 met 113 periodes van een sinus.

- De signalen s_1 en s_2 hebben ongeveer dezelfde frequentie. waarom ziet hun spectrum er zo verschillend uit?

- Het spectrum van s_2 laat ruis zien van en zeer laag nivo, ongeveer -60 dB. Waar komt deze ruis vandaan? Bedenk dat de monsters van een `Sound`-object in de computer gerepresenteerd worden door een getal met *drijvende komma* van 32 bits.
- Waar komt het verschil in hoge frequenties tussen de spectra s_3 en s_4 vandaan?

Maak van al deze signalen een `Spectrum`-object. Teken de afzonderlijke amplitude-spectra samen op één pagina. Laat hierbij de dB-schaal lopen van -100 tot +100 dB.

2. Filteren via het Spectrum.

Maak een geluidje s van 1 s, bemonsteringsfrequentie is 22050 Hz, met hierin gaussische ruis met $\mu = 0, \sigma = 0.1$. Doe de filtering op twee manieren:

1. In twee stappen via een intermediair `Spectrum`-object.

```
select Sound s
To Spectrum (fft)
Formula...  if x...
To Sound (fft)
```

2. Rechtstreeks via `Sound: Filter (formula)...`

Vergelijk de twee resulterende `Sound`-objecten met elkaar. Hoe groot zijn de verschillen in de eerste seconde?
Filter het geluidje s op de volgende manieren:

Laagdoorlaat Filter alle frequenties hoger dan 1000 Hz uit s.

Hoogdoorlaat Filter alle frequenties lager dan 1000 Hz uit s.

Banddoorlaat Filter alle frequenties die kleiner zijn dan 500 Hz of groter dan 1000 Hz uit s.

Bandsper Filter alle frequenties tussen 500 en 1000 Hz uit s.

Maak een serie plaatjes op één vel A4 met daarin twee kolommen. De linker kolom bevat de amplitude van de filterfunctie (de vorm van het filter) als functie van de frequentie, de rechter kolom het amplitude-spectrum van het corresponderende gefilterde signaal s.

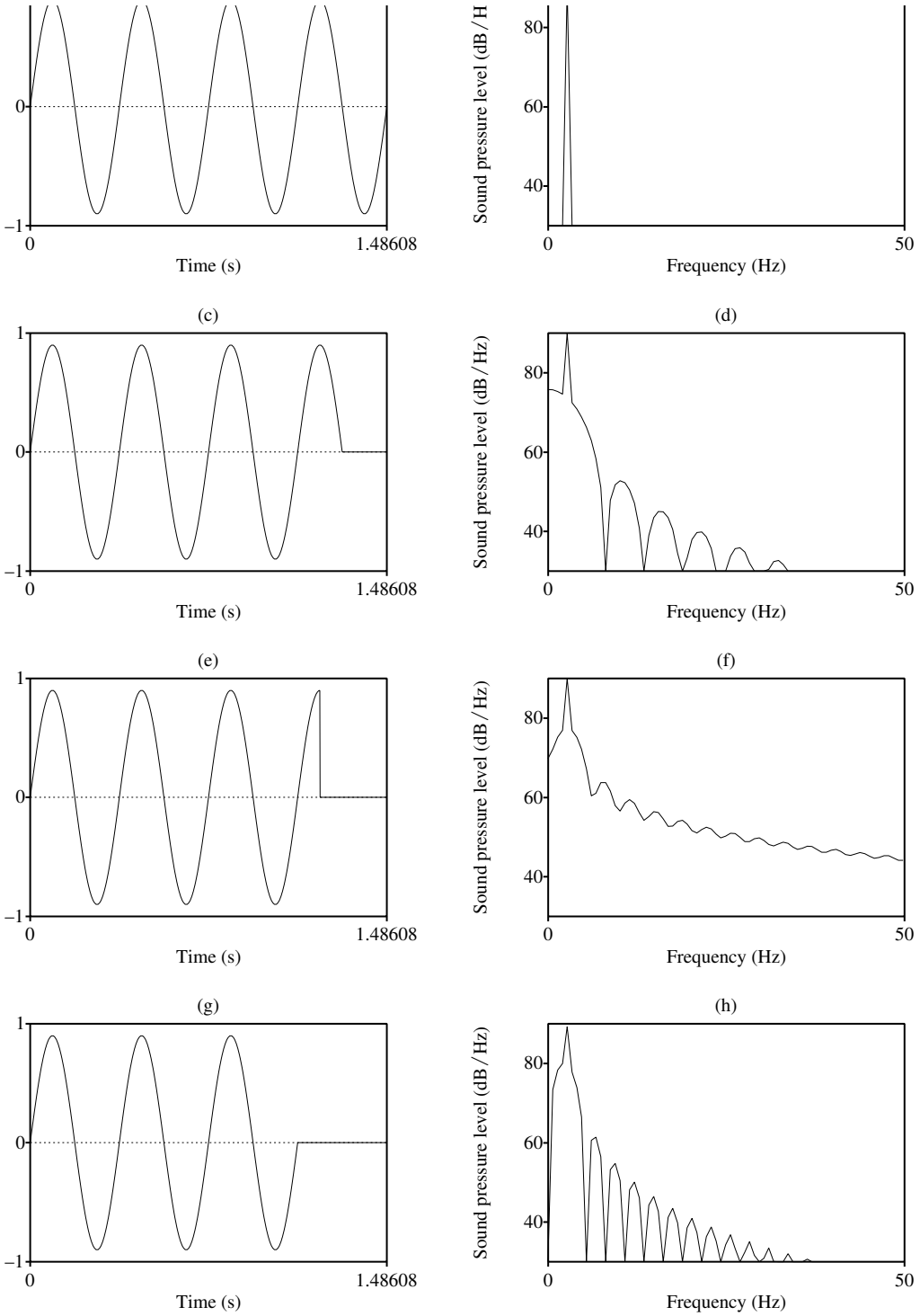
3. Amplitudemodulatie.

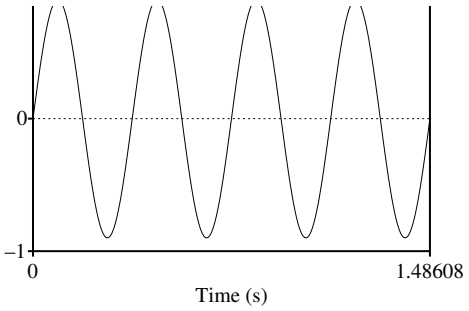
We gaan modulatie en demodulatie simuleren.

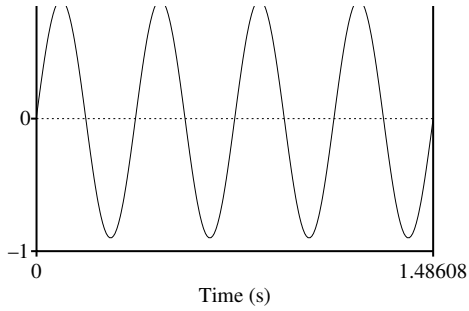
1. Neem een kort geluidje op (1 á 2 s) met een bemonsteringsfrequentie van 48000 Hz.
2. Maak hiervan het spectrum en filter alles boven de 4000 Hz weg.
3. Synthetiseer het geluid terug vanuit het Spectrum.
4. Vermenigvuldig dit geluid met een cosinus van 10000 Hz. Dit is de eigenlijke *modulatie* stap.
5. Maak het Spectrum. Wat is de totale bandbreedte geworden van het deel dat de spraak bevat?
6. Filter een zodanig deel weg dat de totale bandbreedte gelijk wordt aan de bandbreedte na stap 2 en de informatie in het signaal behouden blijft.
7. Transformeer weer naar een Sound. We hebben nu een stuk spraaksignaal waarvan alle frequenties over dezelfde afstand verschoven zijn (gemoduleerd). Het is niet meer verstaanbaar.
8. De *demodulatie*stap. Vermenigvuldig de Sound van stap 7 weer met een cosinus van 10000 Hz en maak het Spectrum.
9. Filter zodat het signaal beneden de 4000 Hz blijft. We hebben nu ongeveer op een schaalfactor na, het oorspronkelijke signaal terug.
10. Probeer nu twee verschillende stukjes spraak die precies even lang zijn te moduleren naar een signaal. Het eerste stukje blijft in het frequentie gebied van 0 tot 4000 Hz, het tweede gaat naar 4000 tot 8000 Hz.

Op een ”analoge” manier kunnen we dus in het frequentiegebied van 0 tot 24000 Hz dan 6 verschillende stukjes spraak kwijt van elk 4000 Hz bandbreedte door een combinatie van modulatie en filtering!

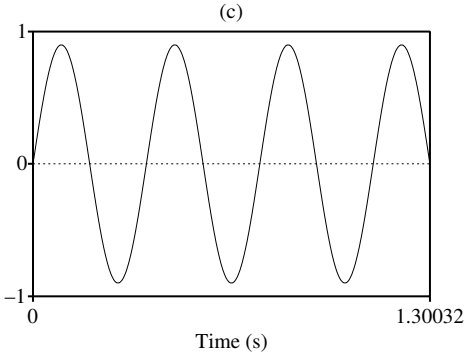
Op een vergelijkbare manier worden bijvoorbeeld meerdere telefoongesprekken tegelijkertijd over een kabel verzonden of worden door de kabelmaatschappij tegelijkertijd over één kabel meerdere televisiezennders en radiozenders aangeboden.


0
–1
0
1.48608
Time (s)
Sound pressure level (dB / H
80
60
40
0
50
Frequency (Hz)
(c)
1
0
–1
0
1.48608
Time (s)
(d)
Sound pressure level (dB / Hz)
80
60
40
0
50
Frequency (Hz)
(e)
1
0
–1
0
1.48608
Time (s)
(f)
Sound pressure level (dB / Hz)
80
60
40
0
50
Frequency (Hz)
(g)
1
0
–1
0
1.48608
Time (s)
(h)
Sound pressure level (dB / Hz)
80
60
40
0
50
Frequency (Hz)

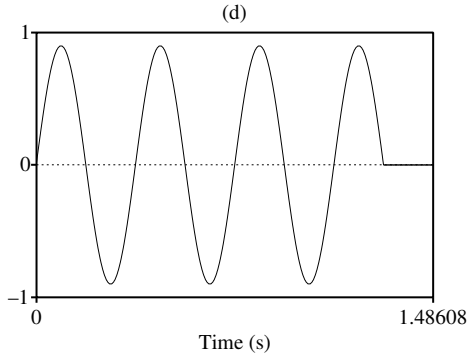

0
–1
0
1.48608
Time (s)


0
–1
0
1.48608
Time (s)

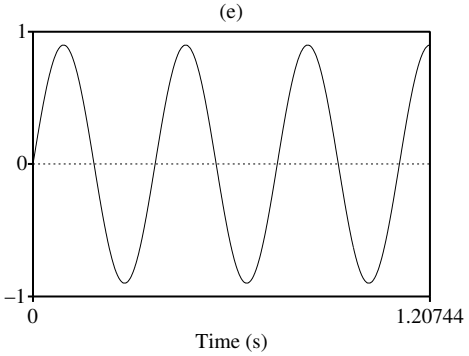
(c)


1
0
–1
0
1.30032
Time (s)

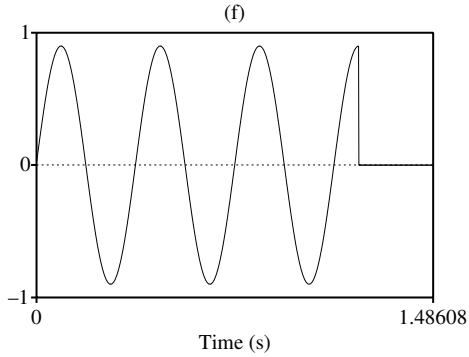
(d)


1
0
–1
0
1.48608
Time (s)

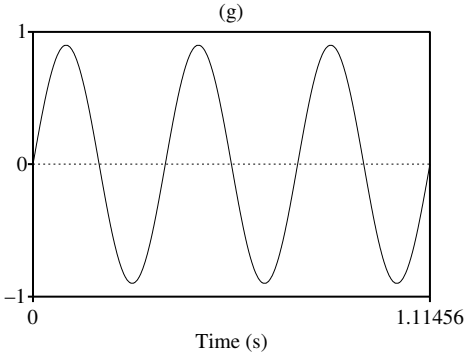
(e)


1
0
–1
0
1.20744
Time (s)

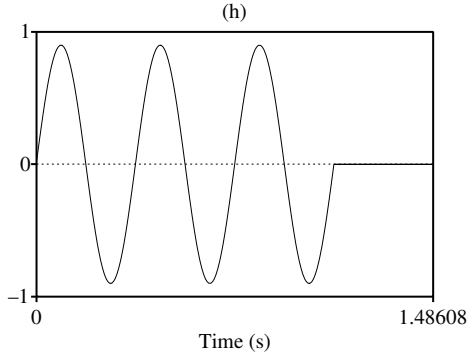
(f)


1
0
–1
0
1.48608
Time (s)

(g)


1
0
–1
0
1.11456
Time (s)

(h)


1
0
–1
0
1.48608
Time (s)

Hoofdstuk 5

Klasse Spectrogram

5.1 Inleiding

Het spectrogram is waarschijnlijk de meest gebruikte spectro-temporele afbeelding binnen de signaalanalyse in het algemeen en binnen de spraaksignaalanalyse in het bijzonder. Het spectrogram hoort tot de klasse van de zogenaamde kwadratische tijd-frequentie signaalrepresentaties[1]. In tegenstelling tot het spectrum biedt het spectrogram inzicht in de *variatie* van de spectrale inhoud als functie van de tijd. Inzicht hierin is uitermate belangrijk omdat spraakklanken door spectrale variatie van elkaar onderscheiden kunnen worden.

5.2 Attributen van een Spectrogram

Een spectrogram kan beschouwt worden als een bemonsterd signaal in twee dimensies, *tijd* en *frequentie*. De attributen zijn:

$x_{min}, x_{max} \in R$ De begin- en eindtijd.

$n_x \in N$ Het aantal spectra in het object.

$dx \in R^+$ De tijdstap: de afstand tussen twee opeenvolgende frames *(*timeStep).

x_1 Het tijdstip van het eerste frame. Als het object het resultaat is van een analyse van een Sound-object dan is het tijdstip van het eerste frame gelijk aan het midden van de gekozen vensterduur.

y_{min}, y_{max} Minimum en maximum frequentie.

[1]Dit betekent, onder andere, dat het *geen* lineaire afbeelding is: Spectrogam($s(t) + h(t)$) $\neq$ Spectrogram($s(t)$) + Spectrogram($h(t)$).

n_y Het aantal frequentie-componenten

dy De afstand tussen twee frequenties.

y_1 De frequentie die bij de eerste rij van z hoort. Meestal is dit $dy/2$.

$z_{ij}, i = 1 \ldots n_x, j = 1 \ldots n_y$ De spectrale vermogensdichtheid in Pa^2/Hz.

5.3 Sound: To Spectrogram

5.3.1 Parameters

Voor het berekenen van een `Spectrogram`-object uit een `Sound`-object moeten een aantal parameters bekend zijn. Deze worden de gebruiker middels een invulformulier gevraagd (zie ook paragraaf 5.3.2 over hoe deze parameters worden gebruikt bij de berekening van het spectrogram.)

analysisWidth De lengte van een analyseframe, het stukje signaal waarvan het specrum wordt uitgerekend. Voor een *breedband* spectrogram nemen we telkens een stukje van 0.005 s. Dit komt overeen met een *bandbreedte* van ongeveer 300 Hertz. Voor betere spectrale resolutie, het zogenaamde *smalband* spectrogram, kiezen we stukjes van 0.03 s, wat overeenkomt met een bandbreedte van ongeveer 45 Hertz. De exacte relaties tussen *bandbreedte* en *analysisWidth* kun je uitrekenen met de formules in tabel A.3. Het gaussische venster bijvoorbeeld heeft een bandbreedte van 301 Hertz als de $analysisWidth$ 0.0043 s is.

maximumFrequency De maximale frequentie die in de analyse meegenomen wordt. Hoe hoog deze waarde ook gekozen wordt, de Nyquist-frequentie bepaalt uiteindelijk de maximale analyse frequentie.[2]

timeStep De afstand tussen opeenvolgende analyseframes in secondes. Deze parameter bepaalt het aantal analyseframes, n_x, in het resulterende `Spectrogram`-object (zie formule 5.2).

frequencyStep bepaalt de frequentie-resolutie in Hertz.

windowShape bepaalt de keuze van het analysevenster. Enkele karakteristieke analysevensters en hun eigenschappen staan vermeld in tabel A.3. Figuur A.6 geeft hun representatie in het tijd- en frequentie-domein.

[2] Dit betekent dat in het resulterende `Spectrogram`-object de `ymax` nooit groter zal zijn dan de Nyquist-frequentie.

5.3.2 Algoritme

We starten met een Sound-object genaamd me.

- Bereken de grootte van de fft-buffer.
- Bereken het venster.
- Bereken het aantal analyseframes (nx) volgens onderstaand algoritme:

$$me.duration = me.x_{max} - me.x_{min} \tag{5.1}$$
$$if(windowType == Gaussian) analysisWidth = 2 \times analysisWidth$$
$$nx = floor((me.duration - analysisWidth)/timeStep) + 1 \tag{5.2}$$

- Voor elk frame:
  - Bereken begin- en eindpositie van frame
  - kopiëer frame naar fft-buffer (vul aan met nullen).
  - Vermenigvuldig met venster
  - bereken de FFT
  - Kwadrateer en schaal voor PSD.
  - Sommeer de PSD's van een aantal frequenties.
  - Kopiëer naar naar een kolom in het `Spectrogram`-object.

Het resultaat van de analyse op `Sound me` is een `Spectrogram` dat we tewr referentie de naam `thee` geven. De attributen van `thee` hebben de volgende waarde gekregen:

- Het domein van het `Spectrogram`-object is gelijk aan dat van het `Sound`-object:
  $thee.x_{min} = me.x_{min}; thee.x_{max} = me.x_{max}$.
- De bemonstering wordt bepaald door de $timeStep$ parameter (zie paragraaf 5.3.1):
  $thee.dx = timeStep$
- Het midden van het eerste frame (x1) wordt bepaalt als:

$$thee.duration = thee.n_x \times timeStep$$
$$thee.x_1 = (me.duration - thee.duration + timeStep)/2$$

- Het bereik in het frequentie-domein wordt bepaalt door de $maximumFrequency$ parameter (5.3.1)
  $thee.y_{min} = 0; thee.y_{max} = maximumFrequency.$
- De frequentie-stap is (5.3.1):
  $thee.dy = frequencyStep$
- Het aantal frequentie-bandjes wordt bepaald door te kijken hoeveel er in het totale frequentie-bereik passen:
  $thee.n_y = floor(maximumFrequency/frequencyStep)$
- De eerste frequentie kennen we toe aan het midden van de eerste frequentie-band:
  $y_1 = frequencyStep/2$

5.4 Het tekenen van spectrogrammen

Het spectrogram wordt getekent als een matrix met grijswaardes, waarbij de hoogste intensiteit zwart is en de laagste wit. De tijd loopt van links naar rechts en de frequentie van beneden naar boven. De volgende parameters beïnvloeden het tekenen van een `Spectrogram`-object via het `Paint`-commando:

From time & To time bepalen het het tijdsinterval.

From frequency & To frequency bepalen het frequentieinterval.

Dynamic range bepaalt het verschil in dB tussen zwart en wit.

Pre-emphasis bepaalt hoeveel de hoge frequenties bevoordeeld worden t.o.v. de lage frequenties.

Dynamic compression de totale hoeveelheid zwart in het plaatje.

5.5 Opgaves

1. Simulatie van Sound to Spectrogram.
Maak een geluidje waarin de frequentie van het signaal verandert:
`Create Sound... sweep 0 1 11025 sin(2*pi*(100+750*x)*x)`.
Maak een ”Spectrogram” door telkens een klein stukje hieruit te snijden, met een Hamming-venster, via `Convert / Extract part...`, dit stukje om te zetten naar een `Spectrum`-object en dit object te vragen naar de energie in een frequentiegebiedje via `Get band energy...`.

Vergelijk het plaatje van het op deze manier verkregen spectrogram met het `Spectrogram`-object dat je krijgt door rechtstreeks van een `Sound`-object een `Spectrogram`-object te maken (uiteraard met dezelfde parameter-instellingen).
Aanwijzingen:

- Je hebt een loop in een loop nodig. De buitenste loop heeft een index die de tijd representeert en de index van de binnenste loop representeert de frequentie.
- Het `Spectrogram`-object moet je via een omweg maken: creëer eerst een `Matrix`-object met de goede dimensies. Via `Set value...` kun je elke individuele cel van dit object bereiken. Als je de inhoud van alle cellen hebt berekend kun je het `Matrix`-object omzetten naar een `Spectrogram`-object.

Hoofdstuk 6

Klasse Pitch

6.1 Inleiding

Een `Pitch`-object bevat kandidaten voor periodiciteit, met hun sterktes, als functie van de tijd. Alhoewel er op elk tijdstip meerdere kandidaten kunnen zijn voor periodiciteit, horen we er hier maar één van. De periodiciteit in het `Pitch`-object kan refereren aan periodiciteit in de produktie, in de perceptie of in de akoestiek. Bij periodiciteit in de produktie denken we aan het regelmatig openen en sluiten van de stembanden. Periodiciteit in de perceptie kunnen we horen en we noemen het toonhoogte of ook wel *pitch*. In het akoestische domein spreken we van periodiciteit. We zullen de termen periodiciteit, toonhoogte en pitch een beetje door elkaar heen gebruiken.

6.2 Attributen van een Pitch

De attributen bevatten de essentiële informatie van het `Pitch`-object. Deze attributen zijn:

$x_{min}, x_{max} \in R$ De begin- en eindtijd.

$n_x \in N$ Het aantal frames in het object.

$dx \in R^+$ De tijdstap: de afstand tussen twee opeenvolgende frames.

x_1 Het tijdstip van het eerste frame. Afhankelijk van de analysevensterbreedte.

$ceiling$ Kandidaten boven deze frequentie worden als stemloos beschouwd.

Pitch_frame $frame_i, i = 1..n_x$ de frames met de pitch gegevens. De attributen van een Pitch_frame structuur zijn:

$nCandidates$ Het aantal kandidaten in dit frame.

Pitch_candidate $candidate_j, j = 1..nCandidates$ De kandidaten. De attributen van een Pitch_candidate structuur zijn:

$frequency$ De frequentie van een kandidaat in Hertz. Deze frequentie is 0 als de kandidaat stemloos is.

$strength$ De mate van periodiciteit van deze kandidaat uitgedrukt met een getal tussen 0 en 1.

6.3 De toonhoogte-analyse

6.3.1 Inleiding

Er zijn veel verschillende manieren ontwikkeld om toonhoogte te bepalen. In Hess (1992) wordt een overzicht gegeven van veel gangbare methodes tot op dat moment. Een ruwe categorisering van de methodes in de drie groepen:

- Methodes die werken in het tijddomein.
- Methodes die werken in het frequentie-domein.
- Methodes die werken met modelering van de auditieve periferie.

Elke goede methode moet in ieder geval meerdere kandidaten voor toonhoogte opleveren. De beste methode voor periodiciteitsbepaling is ontwikkeld door Boersma (1993) en in *praat* geïmplementeerd. De methode werkt via de autocorrelatiefunctie en bevat een technische vernieuwing. Met de stelling van de auteur:

> ”De nauwkeurigheid van de autocorrelatiemethode voor het bepalen van de toonhoogte en harmoniciteit van een periodiek signaal neemt toe met een factor van een miljoen als men corrigeert voor de autocorrelatie van het gebruikte analysevenster.”

De in *praat* geïmplementeerde autocorrelatiemethode voor het vinden van periodiciteit bestaat uit twee stappen:

1. Het vinden van de kandidaten voor de toonhoogte en hun sterktes binnen elk analyseframe.
2. Het vinden van de beste toonhoogte binnen elk frame. Op basis van *kosten* (octaafsprongen en stemhebbend-stemloos overgangen) en *baten* (de sterkte van een kandidaat) wordt een *optimaal pad* gezocht (via een dynamisch programmeer algoritme).

6.3.2 Het vinden van de kandidaten

Het vinden van de kandidaten in elk frame gebeurt via een autocorrelatie van het gevensterde spraaksegment. In het akoestische domein liggen per definitie de beste kandidaten voor periodiciteit bij maxima van de autocorrelatiefunctie. *Bemonstering* en *venstering* veroorzaken echter problemen bij het nauwkeurig bepalen van de *positie* en de *hoogte* van deze maxima.

Het *bemonsteren* van het signaal heeft als gevolg dat de autocorrelatiefunctie ook bemonsterd is. Dit betekent dat, zonder een goede interpolatie, de positie van een (lokaal) maximum niet nauwkeuriger dan op ongeveer de helft van de bemonsteringstijd kan geschieden. Voor een bemonsteringsfrequentie van 10 kHz en een fundamentele frequentie van 300 Hz betekent dit een nauwkeurigheid van ongeveer 9 Hz. Ook de hoogte van de piek is heel erg afhankelijk van de positie van het maximum ten opzichte van de bemonsteringsmomenten, zoals figuur 6.1 voor twee extreme situaties laat zien.

De *venstering* kan ook nog een storend effect op de relatieve hoogtes van pieken hebben. Figuur 6.2 laat het resultaat zien dat vensteren heeft op het spraakachtig signaal

$$s(t) = (1 + 0.3 \sin 2\pi 140t) \sin 2\pi 280t, \tag{6.1}$$

met een fundamentele frequentie van 140 Hz en een sterke "formant" bij 280 Hz. Boven-links wordt een stukje van 0.024 s van het signaal vermenigvuldigd met een venster (boven-midden) resulterend in het gevensterde signaal (boven-rechts). Onder-links staat de autocorrelatie van het gevensterde signaal. Een fundamentele frequentie van 140 Hz zou dan een maximum hebben bij het tijdstip 7.14 ms. We zien echter dat het maximum bij 3.07 ms hoger ligt en daarom (ten onrechte) een betere kandidaat voor de periodiciteit is. De speciale truc, de autocorrelatiefunctie (links-onder) delen door autocorrelatiefunctie van het venster (midden-onder), herstelt de zaak (rechts-onder).

Het probleem dat nu nog rest is het vinden van de positie en de amplitude van de lokale maxima in de autocorrelatiefunctie. Dit probleem is te herleiden tot het gebruiken van de *juiste* interpolatiefunctie: de vertrouwde "sinus-x-over-x"-functie. Dit stelt het algoritme in staat om, met een vensterduur van 40 ms, van een signaal met een 3777 Hz frequentie, bemonsterd met 10 kHz, de fundamentele frequentie te bepalen met de volgende ongelofelijke nauwkeurigheid: 3777.000000 ± 0.000001 Hz. De amplitude van de piek ligt dan tussen 0.99999999 en 1.

6.3.3 Het vinden van het optimale pad

We hebben hier het volgende probleem: we hebben N analyseframes met in elk frame maximaal N kandidaten, inclusief de stemloze kandidaat. In principe heb-

ben we dan M^N verschillende keuzes voor de toonhoogte. Elke mogelijke keuze voor de toonhoogtes kunnen we zien als een *pad* in een $N \times M$ matrix. Alle keuzes zijn gelijkwaardig en we hebben nog geen manier om het "juiste" pad te kiezen. Om een verantwoorde keuze te kunnen maken gaan we *kosten* en *baten* introduceren. We kunnen dit doen *binnen* elk frame, maar ook voor de *overgang* van elk frame naar het volgende. De optimale toonhoogte in elk frame maakt dan deel uit van een pad dat *minimale kosten* heeft (of maximale baten).

Kosten en baten per frame

We kennen aan elke kandidaat baten toe ter grootte van zijn sterkte (dit is $r(\tau_{max})$, de genormaliseerde autocorrelatie, een getal tussen 0 en 1). Dit is nog niet voldoende omdat voor een perfekt periodiek signaal ook subharmonische pieken in de autocorrelatie dezelfde sterkte hebben als de "echte" piek. We willen hier dan graag de *hoogste* frequentie bevoordelen. Dit gebeurt door de introductie van de parameter *octaveCost*. De netto baten per kandidaat bedragen dan:

$$r(\tau_{max}) - octaveCost \cdot \log_2(minimumPitch \cdot \tau_{max}) \tag{6.2}$$

We zien dat hoe groter de parameter $octaveCost$ wordt, hoe meer hogere frequenties bevoordeeld worden. Een andere belangrijke reden om deze parameter in te voeren heeft te maken met het verschil in akoestische fundamentele frequentie en de perceptieve toonhoogte. Dit kan worden geïllustreerd met het volgende signaal:

$$s(t) = (1 + d_{mod} \sin 2\pi \cdot F \cdot t) sin 2\pi \cdot 2F \cdot t, \tag{6.3}$$

een in amplitude gemoduleerd signaal met modulatiediepte d_{mod}. Formule 6.1 is een voorbeeld waarin de modulatiediepte 30% is (zie ook figuur 6.2 links-boven). Bovenstaand signaal heeft een akoestische toonhoogte F en een perceptieve toonhoogte $2F$ als de modulatiediepte kleiner is dan 20 of 30%. Met behulp van de parameter $octaveCost$ kunnen we de sterktes van de F en de $2F$ kandidaten beïnvloeden. Het kantelpunt ligt bij een waarde voor $octaveCost$ gelijk aan d_{mod}^2. Stel we willen dat bij een modulatiediepte van 20% de $2F$ component de sterkste wordt, dan de veranderen we $octaveCost$ naar $(0.2)^2 = 0.04$.

Er is nu al een optimaal pad mogelijk, namelijk de verbinding van alle kandidaten met de hoogste baten.

Kosten en baten tussen frames

Hier hebben we potentiëel te maken met stemhebbend-stemloos overgangen of octaafsprongen. Er worden hiervoor twee parameters geïntroduceerd, *voicedUn-*

voicedCost en *octaveJumpCost*.

$$\begin{array}{rlll} voicedUnvoicedCost & f_1 = 0 & \text{of} & f_2 = 0 \\ octaveJumpCost \cdot |\log_2 f_1/f_2| & f_1 \neq 0 & \text{en} & f_2 \neq 0 \end{array} \tag{6.4}$$

We hebben te maken met stemhebbend-stemloos kosten zowel bij de overgang stemhebbend-stemloos als ook bij stemloos-stemhebbend. Hoe groter *voicedUnvoicedCost* is, hoe minder overgangen van stemhebbend naar stemloos er zullen voorkomen in het pad.

Wanneer *octaveJumpCost* groter gekozen wordt dan zullen er minder grote frequentiesprongen voorkomen in het pad. Met formulde 6.4 worden frequentiesprongen naar boven en naar beneden relatief even zwaar gewogen.

Door *voicedUnvoicedCost* en *octaveJumpCost* beiden gelijk nul te kiezen zet men als het ware het padzoek-algoritme uit: het optimale pad bestaat dan uit de aaneenrijging van de optimale keuzes van elk frame.

6.3.4 Het algoritme

6.4 Sound: To Pitch

6.5 Parameters

De volgende parameter beïnvloeden de selectie van de kandidaten.

timeStep (0.01 s) De afstand tussen twee opeenvolgende analyseframes.

minimumPitch (75 Hz)Kandidaten met een lagere frequentie worden niet meegenomen. Deze parameter bepaalt tevens de lengte van het analysevenster.

maximumNumberOfCandidates (15) Het maximale aantal kandidaten per frame waarmee wordt gewerkt (inclusief de stemloze kandidaat).

veryAccurate (uit) Als de schakelaar *uit* staat wordt een Hanning-venster met een lengte van $3/minimumPitch$ gebruikt. Als de schakelaar *aan* staat wordt een Gaussisch venster met lengte $6/minimumPitch$ gebruikt.

De volgende parameters hebben invloed op de nabewerking die het optimale pad zoekt.

silenceThreshold (0.03) Frames waarin de maximale signaalamplitude niet boven deze waarde uitkomt zijn waarschijnlijk stemloos (relatief ten opzichte van de globale maximale amplitude).

voicingThreshold (0.45) De grenswaarde voor stemhebbend-stemloos beslissing. Als de sterkte van een kandidaat groter is dan deze waarde dan wordt hij beschouwd als stemhebbend. Verhoging van deze waarde vergroot het aantal stemloos beslissingen.

octaveCost (0.01) De mate waarin hogere frequentie worden bevoordeeld ten opzichte van de lagere (zie ook paragraaf 6.3.3). Verhoging van deze waarde bevoordeeld hogere frequenties sterker.

octaveJumpCost (0.35) Kosten voor octaafsprongen. Verhoging van deze waarde verlaagt het aantal frequentiesprongen.

voicedUnvoicedCost (0.14) Boete op stemloos-stemhebbend overgangen. Verhoging van deze waarde verlaagt het aantal stemloos-stemhebbend overgangen.

ceiling (600 Hz) kandidaten boven deze frequentie doen niet mee bij het bepalen van het optimale pad.

6.6 Sound: To PointProcess

6.7 Opgaves

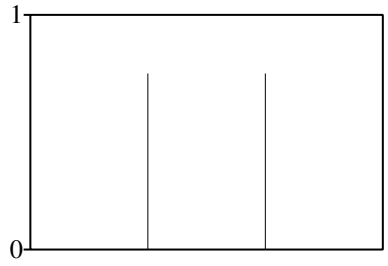
1. Reproduceer figuur 6.2. Het signaal wordt gegeven door

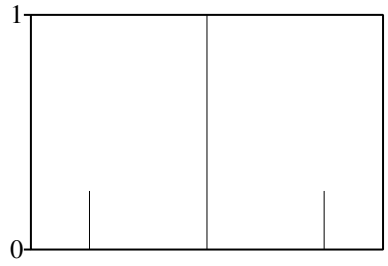
$$s(t) = (1 + 0.3 \sin 2\pi 140t) \sin 2\pi 280t.$$

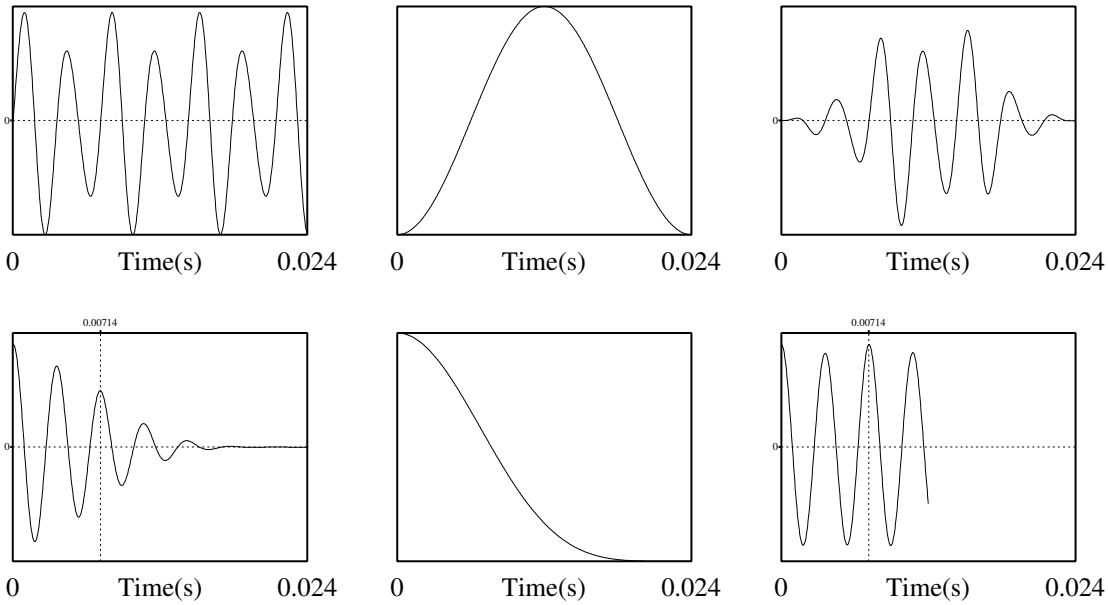
De autocorrelatiefunctie van het Hanningvenster is:

$$r(t) = \left(1 - \frac{|t|}{T}\right)\left(\frac{2}{2} + \frac{1}{3}\cos\frac{2\pi t}{T}\right) + \frac{1}{2\pi}\sin\frac{2\pi|t|}{T} \qquad (6.5)$$

2. Bepaal voor bovenstaand signaal hoe groot we de parameter $octaveCost$ moeten maken om een fundamentele frequentie van 280 Hz te vinden. Je kunt dit het snelste bepalen door in de `PitchEditor` onder het `Edit`-menu de `Pathfinder`-optie te gebruiken.


1
0
Time(s)


1
0
Time(s)


0
Time(s)
0.024
0
Time(s)
0.024
0
Time(s)
0.024
0.00714
0
Time(s)
0.024
0
Time(s)
0.024
0.00714
0
Time(s)
0.024

Hoofdstuk 7

Klasse LPC

7.1 Inleiding

Een LPC-object representeert filtercoëfficiënten als een functie van de tijd. De afkorting LPC betekent *Linear Predictive Coding*. Achtergrondinformatie over de LPC-analyse kan gevonden worden in hoofstuk C en de hiermee samenhangende digitale filtering in hoofdstuk B.

7.2 Attributen van een LPC

x_{min}, x_{max} De begin- en eindtijd.

n_x Het aantal frames in het object.

$dx \in R^+$ De tijdstap: de afstand tussen twee opeenvolgende frames.

x_1 Het tijdstip van het eerste frame. Afhankelijk van de analysevensterbreedte.

samplingInterval De afstand tussen twee opeenvolgende monsterwaardes van de uitvoer van het filter. Als het LPC-object het resultaat is van de analyse van een Sound-object dan is het *samplingInterval* gelijk aan de bemonsteringstijd dx van het geanalyseerde Sound-object.

LPC_frame $frame_i, i = 1..n_x$ de frames met de filter-gegevens. De attributen van een LPC_frame structuur zijn:

nCoefficients

$a_j, j = 1..nCoefficients$

gain

7.3 Sound: To LPC

De standaard manier om een `LPC`-object te maken is het uitvoeren van een lpc-analyse op een `Sound`-object. Er zijn in *praat* vier verschillende methodes geimplementeerd om de filtercoëfficiënten uit te rekenen. Er zijn twee standaard methodes bij, gebaseerd op voorwaartse predictie. Deze methodes zijn de *autocorrelatie*methode en de *covariantie*methode. De algoritmes hiervoor zijn bijvoorbeeld te vinden in het boek van Markel & Gray (1976). De algoritmes voor de twee andere methodes, die van *Marple* en *Burg* zijn ook publiek, zie Marple (1980)en Press et al. (1996).

In het formulier dat op het scherm verschijnt na één van de vier lpc-analyses gekozen te hebben zijn de volgende parameters van belang:

Prediction order Het maximale aantal coëfficiënten per analyseframe dat berekend mag worden. Het aantal coëfficiënten dat we nodig hebben voor een lpc-analyse is afhankelijk van de bemonsteringsfrequentie (en het aantal formanten dat we "willen"). Elke formant is een tweede orde sectie (zie B.5) en voegt daarom twee coëfficiënten toe aan het totaal. Het minimale aantal coëfficiënten is daarom gelijk aan twee maal het aantal te verwachten formanten. Markel (1972) suggereert dat voor bemonsteringsfrequenties tussen 6 en 18 kHz voor de analyse van mannenstemmen $bemonsterings-frequentie/1000 + 4$ coëfficiënten voldoende zijn. Dit komt overeen met orde 14 voor een bemonsteringsfrequentie van 10 kHz.

Voor vrouwenstemmem moeten we om dezelfde hoeveelheid formanten te meten het frequentiegebied uitbreiden met ongeveer 10%, óf het aantal formanten verminderen. Voor een orde 14 analyse kunnen we dan een bemonsteringsfrequentie van 11 kHz gebruiken. Het alternatief is een orde 12 analyse voor een bemonsteringsfrequentie van 10 Khz.

Voor kinderstemmen moeten we de bandbreedte nog een keer met 10% verhogen tot ongeveer 12 kHz om een orde 14 analyse te rechtvaardigen. Wanneer we 10 kHz als bemonsteringsfrequentie nemen dan zijn 10 coëfficienten wel voldoende.

Analysis width De afstand tussen opeenvolgende analyseframes. Het analysevenster bepaalt het aantal analyseframes (n_x) en het midden van het eerste frame (x_1). De keuze voor de lengte van het analysevenster wordt bepaalt door twee factoren: de frequentieresolutie en de spectrale middeling. Als het venster te kort is kunnen spectrale componenten die dicht bij elkaar liggen niet opgelost worden. Als het venster te lang is zullen in een deel van het signaal aanwezige sterke formantpieken zwakker kunnen worden door

middeling. Meestal wordt wel een aantal periodes van het signaal genomen zodat we de duur nemen tussen 10 en 35 ms.

Time step De afstand tussen twee opeenvolgende analyseframes.

Pre-emphasis from De frequentie F van waaraf pre-emfase moet moet worden toegepast. Vanaf deze frequentie zal de helling met ongeveer +6 dB/octaaf toenemen. De coëfficiënt a van dit eerste orde filter, zie vergelijking B.8, is dan $e^{-2\pi FT}$, waarbij T de bemonsteringstijd is.

De andere attributen van het `LPC`-object worden óf overgenomen van het `Sound`-object óf berekend.

7.4 LPC & Sound

Wanneer we een `LPC`-object en een `Sound`-object samen geselecteerd hebben dan hebben elk van de twee hierna gegeven mogelijkheden een nieuw `Sound`-object tot gevolg.

1. `Filter...`
   Het geselecteerde `Sound`-object wordt als bronsignaal van het lpc-filter genomen. De resulterende uitvoer van het filter, de monsterwaardes y_n, worden dan gegeven door formule C.1:

   $$y_n = -\sum_{k=1}^{p} a_k(t) y_{n-k} + x_n,$$

   waarbij de x_n de monsterwaardes van het geselecteerde `Sound`-object representeren en $a_k(t)$ de predictiecoëfficiënten in het lpc-frame dat correspondeert met deze n-de monsterwaarde. De spectrale representatie van deze mogelijkheid is $Y(z) = H(z)X(z)$.

2. `Filter (inverse)`
   Het geselecteerde `Sound`-object wordt beschouwd als het uitgangssignaal van het lpc-filter. Bij inverse filtering wordt het bronsignaal van het filter berekent uit de karakteristiek van het filter en het uitvoersignaal. Dit bronsignaal kunnen we berekenen door het toepassen van vergelijking C.4:

   $$x_n = y_n + \sum_{k=1}^{p} a_k(t) y_{n-k},$$

   waarbij de $a_k(t)$ weer de tijdafhankelijke filtercoëfficiënten zijn. In het frequentiedomein hoort hier de vergelijking $X(z) = A(z)Y(z)$ bij.

Wanneer we starten met een willekeurig signaal `s` en we voeren het volgende script uit:

```
1  select Sound s
2  To LPC (autocorrelation)... 16 0.025 0.05 50
3  plus Sound s
4  Filter (inverse)
5  Rename... bron
6  plus LPC s
7  Filter
8  Rename... resyn
```

dan is het geresynthetiseerde signaal met naam `resyn`, op afrondingsfouten na, gelijk aan het originele signaal `s` (behalve twee kleine stukjes in het begin en aan het eind van het signaal ter lengte van een half analysevenster). Dit geldt onafhankelijk van de gekozen methode waarmee we het filter berekenen in regel 2 van het script. Het lpc-filter en het bronsignaal zijn altijd een complete beschrijving van het signaal `s`. De beschrijving wordt pas incompleet als we bijvoorbeeld de stemhebbende delen van het bronsignaal gaan benaderen met een pulstrein, of de ruisachtige delen met kunstmatig gegenereerde ruis.

7.5 LPC: To Spectrum

We kunnen uit de lpc-coëfficiënten het spectrum van het filter berekenen. Formule C.19 geeft op een schaalfactor na aan hoe dit moet gebeuren. Het volgende script geeft aan hoe dit met één lpc-frame gebeurt.

```
1  select LPC a
2  To Polynomial (slice)... <een_tijdstip>
3  To Spectrum... 5000 1025
4  Copy... h
5  Formula... (if row=1 then 1 else -1 fi)
6  ... *self/(Spectrum_a[1,col]^2+Spectrum_a[2,col]^2)
```

In regel 2 worden de p lpc-coeëfficiënten uit een lpc-frame gekopiëerd naar de coefficiënten van een `Polynomial`-object, waarbij de relatie tussen de coëfficiënten c_i van het polynoom en de a_k van het `LPC`-object de volgende is:

$$c_i = a_{p-i+1}, \quad \text{en} \quad c_{p+1} = 1.$$

In regel 3 wordt dit polynoom geëvalueerd over de eenheidscircel op 1025 punten, waarbij het laatste punt ligt bij de Nyquist-frequentie die hier bij 5000 Hz gekozen

is[1] We hebben nu het spectrum van het inverse filter $A(z)$ berekend. In regel 5 en 6 wordt tenslotte $H(z)$ berekend, de inverse van $A(z)$.

7.6 LPC: To Spectrogram

Het spectrogram bestaat in dit geval uit de spectra van de lpc-frames. Deze spectra zijn berekend op de manier zoals aangegeven in paragraaf 7.5.

7.7 LPC: To Formant

Wanneer we geïnteresseerd zijn in formantfrequenties en bandbreedtes, dan kunnen we uit een `LPC`-object een `Formant`-object maken. De formantfrequenties kunnen berekend worden uit de nulpunten van de $A(z)$ van vergelijking C.21. Deze nulpunten moeten we met een speciaal algoritme uit deze vergelijking berekend worden. Hoe hoger de orde p van de $A(z)$ hoe moeilijker het is. Nulpunten bepalen van polynomen is rekentechnisch een moeilijke klus omdat kleine veranderingen in de coëfficiënten van een polynoom al grote veranderingen in de positie van de nulpunten kunnen hebben. Het algoritme dat in *praat* geïmplementeerd is numeriek stabiel tot ongeveer $p = 24$.

$$\begin{aligned} A(z) &= 1 + \sum_{k=1}^{p} a_k z^{-k} \\ &= \frac{z^p + \sum_{k=1}^{p} a_k z^{p-k}}{z^p} \end{aligned}$$

We vergeten de noemer, want deze geeft overal een evengrote bijdrage:

$$\begin{aligned} A(z) &= \prod_{k=1}^{p} (z - z_k) \\ &= \prod_{k=1}^{p/2} (z - z_k)(z - z_k^*) \end{aligned}$$

[1] Dit is equivalent aan de Fourier-transformatie van een buffer ter lengte van 1024 met daarin de lpc-coëfficiënten $(a_p, a_{p-1}, \ldots, a_1, 1, \ldots)$.

Zoals we in paragraaf B.5.1 hebben laten zien kunnen we uit de positie van de nulpunten de formantfrequentie F_k en de bandbreedte B_k van het k-de nulpuntspaar worden afgeleid:

$$F_k = \frac{1}{2\pi T} \arctan \frac{z_{ki}}{z_{kr}}$$
$$B_k = \frac{1}{2\pi T} \ln(z_{kr}^2 + z_{ki}^2),$$

waarbij

$$z_k = z_{kr} + i z_{ki}$$

Deze berekende waardes voor F_k en B_k worden gecopiëerd naar het `Formant`-object. Standaard worden dan formantfrequenties die te dicht bij de nul Hertz en/of de Nyquist-frequentie liggen niet overgenomen in `Formant`-object omdat deze frequentie grote kans maken geen formantfrequenties te zijn maar alleen globale spectrale karakteristieken te beschrijven. Dit betekent dat in het algemeen de transformatie van een `LPC`-object naar een `Formant`-object gepaard gaat met verlies van informatie. Wanneer we geen informatieverlies willen maar alle frequenties en

7.8 Opgaves

Hoofdstuk 8

Klasse Formant

8.1 Inleiding

Voor klinkers is een representatie met formantfrequenties waarschijnlijk één van de meest gebruikte. Een plaatje met de klinkers van een taal in het F_1F_2-vlak geeft een snel overzicht van de klinkerstructuur van de betreffende taal. Formantfrequenties sluiten ook goed aan bij een beschrijving van het spraaksignaal vanuit de productie, als zijnde resonantiefrequenties van het spraakkanaal. Akoestisch gezien kunnen we formantfrequenties definiëren als maxima in het amplitudespectrum.

8.2 Attributen van een Formant

$x_{min}, x_{max} \in R$ De begin- en eindtijd.

$n_x \in N$ Het aantal frames (≥ 1).

$dx \in R^+$ De tijdstap: de afstand tussen twee opeenvolgende frames *(*timeStep).

x_1 Het tijdstip van het eerste frame. Als het object het resultaat is van een analyse van een Sound-object dan is het tijdstip van het eerste frame gelijk aan het midden van de gekozen vensterduur.

Formant_Frame $frame_i, i = 1 \ldots n_x$ De frames met formant gegevens. De attributen van een Formant_Frame zijn:

intensity Een indicatie van de intensiteit in dit frame.

nFormants Het aantal gevonden formanten in dit frame.

Formant_Formant $formant_j, j = 1 \ldots nFormants$ Informatie over elke formant. De attributen van een Formant_Formant zijn:

frequency De frequentie van de formant.

bandwidth De bandbreedte van de formant.

8.3 Het probleem met Formanten

- Interpretatie. Met behulp van formantfrequenties kun je geen afstand tussen klinkers definiëren. Een mogelijke afstandsmaat tussen twee klinkers is bijvoorbeeld
$$(F_{1,a} - F_{1,b})^2 + \alpha(F_{2,a} - F_{2,b})^2 + \ldots \quad (8.1)$$
Idealiter zou α in de buurt van de 1 moeten liggen als de eerste twee formantfrequenties twee onafhankelijke perceptuele dimensies zouden zijn van de klinkerruimte. Boersma (1998, pag. 104) laat zien dat zelfs als je de formule 8.1 interpreteert in eenheden van JND's, de factor $\alpha = 0.3$.

- Het meetproces. Het is na dertig jaar onderzoek naar methodes nog steeds niet mogelijk om formantfrequenties op een objectieve manier automatisch te meten. Alle metingen vereisen in meer of mindere mate interventie van de onderzoeker en zijn gebaseerd op ad hoc procedures.

Vraag: Waarom proberen mensen dan nog steeds formantfrequenties te meten? Antwoord: omdat een plaatje met de klinkers van een taal uitgezet in het formantvlak een mooie compacte representatie vormt.

8.4 Het meten van formanten

We zullen het meten van formanten verduidelijken aan de hand van twee audiobestanden met spraakmateriaal uit het dialect van Volendam. Deze opnames zijn gemaakt door Sijmen Tol[1]. Het materiaal bestaat uit woordjes waarin alle korte klinkers, lange klinkers en diftongen uit het dialect van Volendam zijn vertegenwoordigd.

8.4.1 Het labelen van het materiaal

Het materiaal waar we mee gaan werken bestaat uit twee bestanden: het eerste bestand bevat een aantal woordjes met daarin kort-lang opposities. Het tweede bestand bevat een aantal lange klinkers met diftongen.

[1] email:bl@konbib.nl

We beginnen met het signaal te bekijken in de `SoundEditor`. In figuur 8.1 hebben we een plaatje gemaakt van het ruwe materiaal. Een aantal dingen vallen dan op.

- Ondanks het feit dat het plaatje een zeer gecomprimeerd overzicht geeft van de variatie van de amplitude van het signaal in de tijd kunnen we toch nog redelijk goed de individuele woordjes onderscheiden.

- De amplitude van het signaal is vrij laag, de maximale amplitude in het signaal is -0.29. Voor de meetprocedures maakt dit niet veel uit, de signaal-ruis verhouding wordt hierdoor ongeveer $10 \log(1/0.29) \approx 5.4$ dB slechter. Om tijdens het labelen van het materiaal beter zicht te hebben op het signaal is het aan te raden het signaal zo te schalen dat de (absolute waarde van) de maximale amplitude ongeveer 1 is.[2]

- Het "nulnivo" van het signaal is verschoven. Stiltes in het signaal moeten amplitudes gelijk aan nul hebben en in het oscillogram moeten deze stiltes ook bij nul zitten. Bij opnames gemaakt met (vaak goedkopere) microfoons of slecht afgeregelde apparatuur zie je vaak dat er een constante *offset* in het signaal zit. Deze offset kunnen we weghalen met het commando `Subtract mean`. Het is efficiënter om eerst de offset weg te healen en dan pas de amplitude te schalen.

- De pauzes tussen de woorden die eigenlijk stilte zouden moeten bevatten zijn gevuld met een ruizig signaal. Dit is vooral duidelijk te zien in het onderste plaatje van figuur 8.1. Nadere analyse[3] van deze stukken ruis levert op dat het bestaat uit ademhalingsgeluid, open en sluiten van de lippen en een periodiek verschijnsel, met harmonischen van ongeveer 11 Hertz, dat waarschijnlijk kan worden geassociëerd met het motortje van het cassettedeck dat gebruikt is voor het maken van de opname. De analyses die we gaan gebruiken zijn niet al te gevoelig voor deze ruis en we laten deze daarom gewoon zitten.

Na een deel van) het materiaal bekenen te hebben start men meestal met het labelen van het materiaal. Het labelen bestaat uit het markeren van interessante segmenten in het spraakmateriaal. Dit heeft onderanderen als voordeel dat men

[2]Dit kan door bij het geselecteerde geluid de optie `Scale...` met de default waarde te kiezen (zie ook paragraaf 3.6).

[3]Dit kan bijvoorbeeld door in de `SoundEditor` een aantal van deze stukjes met ruis via `Extract selection` te selecteren, deze allemaal in het `Object`-venster via `Concatenate` aan elkaar te plakken, via `Scale` te versterken, en te beluisteren.

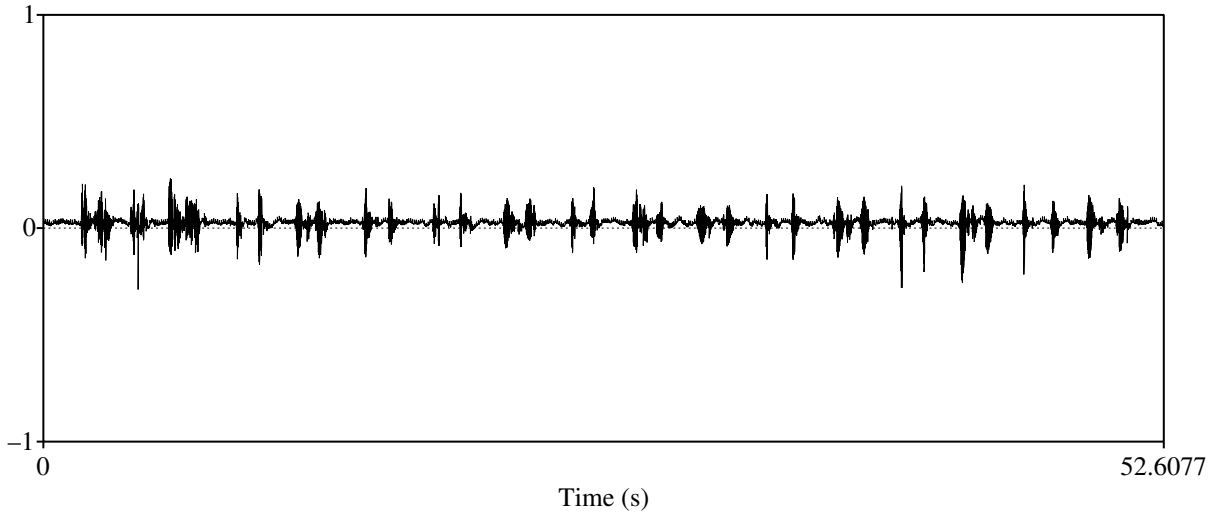
de interessante segmenten snel terug kan vinden én dat men analyses reproduceerbaar kan maken. De labelinformatie wordt gescheiden van het `Sound`-object bewaart in een `TextGrid`-object. Schalen voor betere zichtbaarheid (Subtract mean +Scale...) To Analysis... Fmax=5000 voor man Tweeklanken, waar gemeten??

8.5 Opgaves

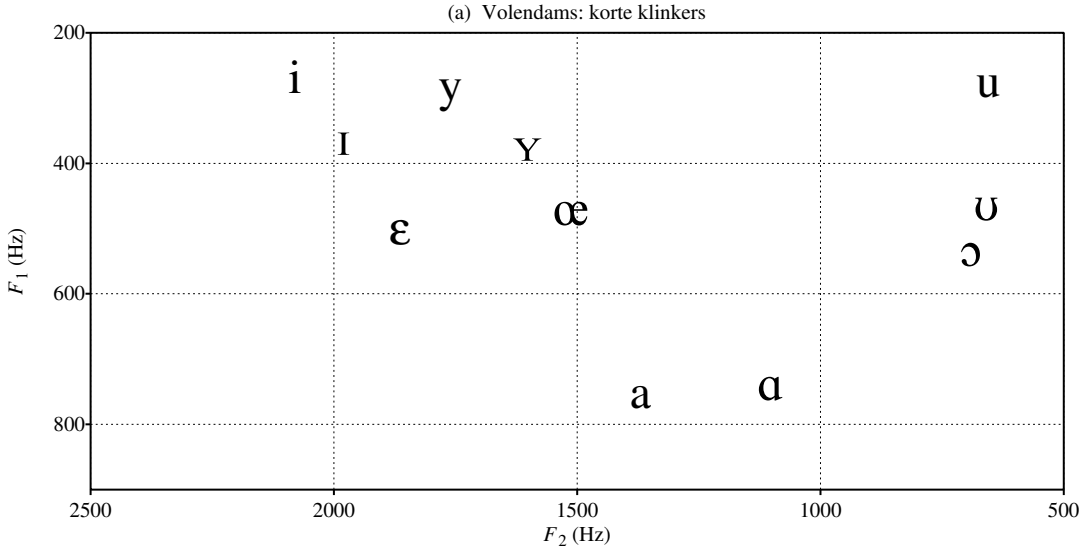
1. Bepaal de eerste twee formantfrequenties van de lange- en korte klinkers van het Volendams. Probeer zoveel mogelijk klinkers die in figuur 8.2 staan te meten uit het spraakmateriaal in de twee bestanden `volendams_kort_lang.aifc` en `volendams_tweeklank.aifc`. Het meetproces bestaat uit de volgende onderdelen:

- Voorbehandeling: Herbemonsteren naar 10 kHz, eventuele offset uit het materiaal halen. Amplitudes schalen tot maximaal bereik.
- Labelen van het materiaal. Maak twee tiers, één met woord labels en een met de klinkerlabels.
- Het meten van de formantfrequenties van het gelabelde klinkermateriaal. Bepaal de frequenties via lpc-analyses. Vergelijk het lpc-spectrum met het spectrogram en bepaal op grond hiervan welke lpc-orde het geschikst is.
- Het maken van drie plaatjes met dezelfde assenindeling als figuur 8.2

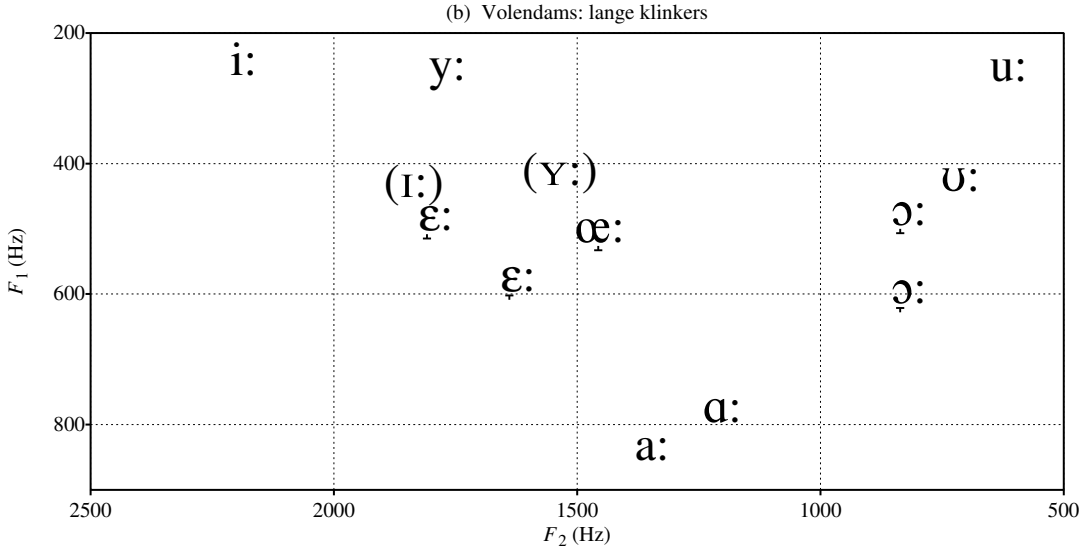
2. Bepaal voor drie klinkers bijvoorbeeld /a/, /u/ en /i/ de eerste twee formantfrequenties voor verschillende lpc-orde. Neem orde 8, 10, 12, 14 en 16.


1
0
–1
0
52.6077
Time (s)

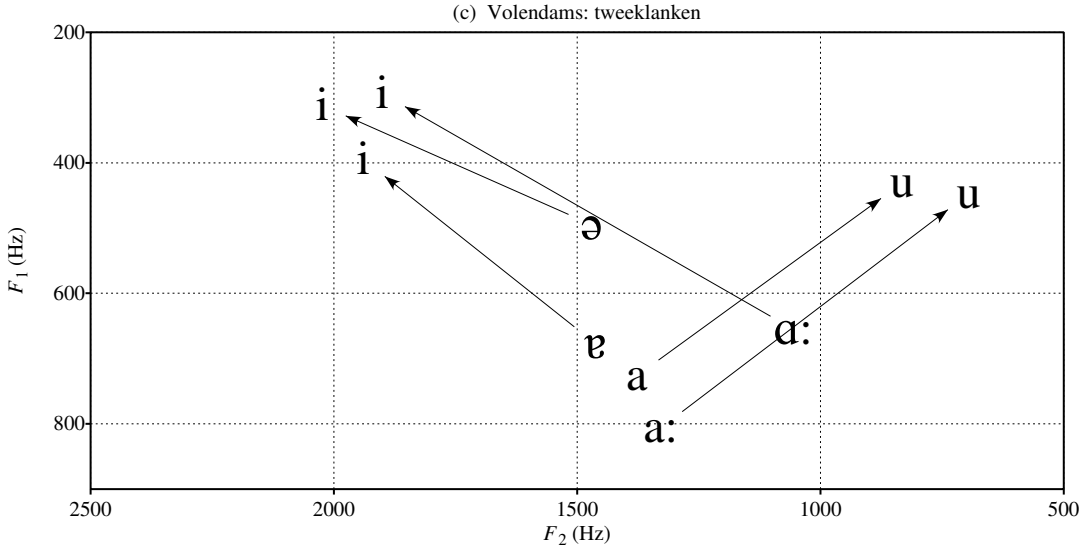
(a) Volendams: korte klinkers



(b) Volendams: lange klinkers



(c) Volendams: tweeklanken



Hoofdstuk 9

Klasse TextGrid

9.1 Inleiding

De klasse TextGrid is geschikt voor het segmenteren en labelen. Het labelen van spraakmateriaal kan bijvoorbeeld gebeuren op zinsnivo, woordnivo en foneemnivo. Er zijn twee soorten van labels:

- Interval labels. Een interval heeft een begintijdstip, t_1, en een eindtijdstip, t_2, waarbij altijd $t_2 > t_1$. Het label hoort bij het interval. Voorbeelden hiervan kunnen zijn woordlabels en foneemlabels.
- Tijdstip labels. Een label dat hoort bij een tijdstip. Een voorbeeld is een label bij het begin of einde van iets of bij het "midden van een klinker".

De nivo's waarop men kan labelen noemt men *tiers*, alle labels zijn georganiseerd in tiers. Een `TextGrid`-object kan meerdere tiers bevatten; elke tier bevat óf alleen interval-labels óf alleen tijdstip-labels.

9.2 Attributen van een TextGrid

9.3 Sound: To TextGrid

De snelste manier om een `TextGrid`-object te maken. De begin- en eindtijd van het nieuwe `TextGrid`-object wordt gecopiëerd van het `Sound`-object. In het formulier dat verschijnt als `To Textgrid...` gekozen is, moeten alle namen van de gewenste tiers ingevuld worden, ook die van de gewenste punttiers. De standaard tiers zijn intervaltiers. Als je geen punttiers wilt dan maak je de regel bij punttiers leeg.

Hoofdstuk 10

Klasse PointProcess

10.1 Inleiding

10.2 Attributen van een PointProcess

Hoofdstuk 11

Klasse MFCC

11.1 Inleiding

11.2 Attributen van een MFCC

Hoofdstuk 12

Databases

12.1 Inleiding

Databases kunnen in principe gezien worden als een tabel. In deze tabel hebben rijen en kolommen verschillende functies. Alle gegevens in een kolom moeten van het zelfde type zijn: een kolom bevat óf alleen namen, óf alleen getallen, óf alleen datums óf alleen ... Een rij in een tabel heet ook wel een *record*. Wanneer je merkt dat in een tabel bij twee of meer kolommen heel vaak dezelfde waardes staan, dat wil zeggen dat er heel vaak dezelfde relaties tussen kolommen voorkomen, dan kun je besluiten om deze kolommen in een aparte, kleinere, tabel onder te brengen en in de oorspronkelijk tabel een verwijzing naar het goede record in de nieuwe tabel. We hebben nu een *relationele database* gecreëerd. Het voordeel van een relationele database is dat hij gemakkelijker te beheren is, het nadeel is dat de formulering van zoekopdrachten (queries) ingewikkelder kan zijn. Aan de hand van een fictief voorbeeldje kunnen we dit illustreren: Stel we willen een internationaal adressenbestand maken met daarin de *naam*, *straat*, *postcode*, *plaats*, *land*. Stel dat we ook geïnteresseerd zijn in het aantal inwoners van dit land en hiervoor een aparte kolom willen opnemen met naam *inwoners*. We kunnen dan om te beginnen een tabel maken met 7 kolommen. Als we deze tabel hebben gevuld met ons adressenbestand dan zou hij er als volgt kunnen uitzien als in tabel 12.1.

Tabel 12.1: Adressenbestand.

naam	straat	...	land	inwoners
jan	oracle 1		NL	15000000
willem	oracle 2		NL	15000000
marie	db 2		NL	15000000
...	...		NL	15000000
claude	postgres 14		FR	61000000
...	...		FR	61000000
john	ingres 3		US	250000000
bill	SQL server 0		US	250000000
...	...		...	...

Bijlage A

De Fourier-transformatie

A.1 Inleiding

De Fourier-transformatie is uitgevonden door, en vernoemd naar, de Franse wiskundige Jean Baptiste Joseph de Fourier (1768–1830). Deze transformatie is één van de allerbelangrijkste hulpmiddelen binnen de signaalanalyse. Bij de Fourier-transformatie onderscheidt men de Fourier-analyse en de Fourier-synthese. Deze worden uitgedrukt door de volgende twee formules:

$$H(f) = \int_{-\infty}^{+\infty} h(t) e^{-2\pi i f t}\, dt \qquad \text{"Analyse"} \tag{A.1}$$

$$h(t) = \int_{-\infty}^{+\infty} H(f) e^{+2\pi i f t}\, df \qquad \text{"Synthese"} \tag{A.2}$$

In deze formules is $h(t)$ een functie van de tijd t, en $H(f)$ een functie van de frequentie f. Als de eenheid van tijd de seconde (s) is, dan is de eenheid van frequentie de Hertz (Hz). De functie $H(f)$ wordt ook wel het *spectrum* van $h(t)$ genoemd. In het spectrum zijn zowel positieve als ook negatieve frequenties vertegenwoordigd.

> In de literatuur over de Fourier-transformatie kun je bovenstaande formules ook tegenkomen met het teken van de exponent verwisseld. Dit is de conventie in de fysisch geörienteerde literatuur, zie bijvoorbeeld in Press et al. (1996) en Wolfram (1996). In de signaalanalyse kom je meestal bovenstaande conventie tegen, zie bijvoorbeeld van den Enden & Verhoeckx (1987), Parsons (1987), Furui (1989), Deller et al. (2000) en Papoulis (1980, 1988). *Praat* hanteert vanaf versie 4.0.17 de bovenstaande *signaalanalytische* conventie. Eerdere versies van *praat* gebruiken nog de fysische conventie.

De complexe exponent is een verkorte schrijfwijze om in één keer een cosinus en een sinus te kunnen gebruiken:

$$e^{2\pi ift} = \cos 2\pi ft + i \sin 2\pi ft \tag{A.3}$$

Formule A.1 beschrijft Fourier-analyse. Met behulp van de Fourier-analyse probeert men een signaal te ontbinden in basisfuncties die bestaan uit sinussen en cosinussen van verschillende frequenties. De sterkte waarmee basisfuncties met verschillende frequenties f in het signaal zijn vertegenwoordigd wordt gegeven door de grootte van $H(f)$.

Formule A.2 beschrijft Fourier-synthese, sinussen en cosinussen worden opgeteld om het tijdsignaal $h(t)$ te construeren.

De formules samen beschrijven twee verschillende *representaties* van hetzelfde signaal: een beschrijving $h(t)$ in het *tijd-domein* en een beschrijving $H(f)$ in het *frequentie-domein*. Bovenstaande Fourier-transformatie-paar beschrijft dan hoe je van de ene naar de andere representatie komt. Beide functies, $h(t)$ en $H(f)$, kunnen in principe complexwaardige functies zijn. In onze praktijk is echter $h(t)$ altijd een reële functie en alleen $H(f)$ is dan complexwaardig.

Sinussen en cosinussen blijken niet alleen mooie wiskundige eigenschappen te hebben maar ook ”basisgeluiden” voor ons oor te zijn. We zullen in dit hoofdstuk proberen te verklaren waarom bij de Fourier-analyse

- sinussen én cosinussen nodig zijn.
- bij periodieke signalen de frequenties van de basisfuncties een harmonische relatie hebben.

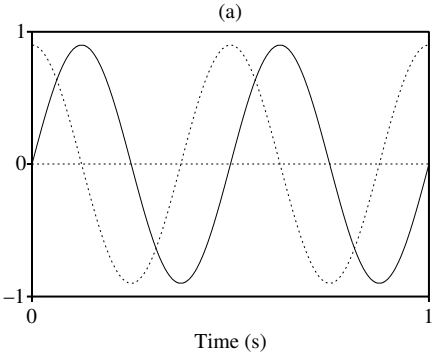
Voordat we hieraan beginnen een paar inleidende begrippen. Een functie $h(t)$ noemen we *even* als $h(t) = h(-t)$ en *oneven* als $h(t) = -h(-t)$. Verder hebben we *reële, complexe* en *imaginaire* functies.

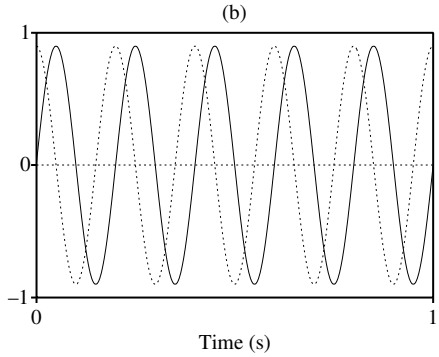
A.2 De basisfuncties

Wij schrijven onze basisfuncties als $\sin 2\pi ft$ en $\cos 2\pi ft$, waarbij f de frequentie en t de tijd is[1]. Meestal is bij de laatste manier van schrijven de variable de tijd t en is de factor $2\pi f$ een ”constante”. Eén periode van $\sin x$ wordt doorlopen als x gaat van 0 tot 2π. Eén periode van $\sin 2\pi ft$ wordt doorlopen als t gaat van 0 tot $1/f$.

In figuur A.1 is een voorbeeld getekend van deze basissignaaltjes. De $\sin$ is een *oneven* en de $\cos$ een *even* functie. Verder zien we duidelijk dat $\sin$ en $\cos$

[1] Als we algemene eigenschappen van sinus en cosinus willen aanduiden schrijven we $\cos x$ of $\sin x$.


(a)
1
0
–1
0
1
Time (s)


(b)
1
0
–1
0
1
Time (s)

eigenlijk in de tijd verschoven versies van elkaar zijn. Eén mogelijke formulering hiervan is:

$$\sin(x) = \cos(x - \pi/2),$$

waarbij we voor x ook $2\pi ft$ mogen denken. Het rechter plaatje laat echter zien dat we niet algemeen genoeg zijn geweest: als we de $\cos$ nog 2π verder schuiven dan valt hij weer over de $\sin$. En nog weer verder schuiven over 2π maakt ze ook weer gelijk. Onderstaande formule vat de relatie tussen de $\sin$ en $\cos$ samen:

$$\sin(x) = \cos(x - \pi/2 \pm 2k\pi),$$

waarbij k een geheel getal is uit de reeks (0, 1, 2, 3 ...). De $\cos$ en de $\sin$ zijn complementair:

$$\cos^2(x) + \sin^2(x) = 1$$

Een andere eigenschap die we gaan gebruiken is de volgende:

$$s(x) = c\sin(x + \phi) = a\cos(x) + b\sin(x),$$

waarbij $a = c\cos\phi$ en $b = c\sin\phi$, of, omgekeerd, $c = \sqrt{(a^2 + b^2)}$ en $\phi = \tan^{-1}\frac{b}{a}$. Elke $\sin$ die, als $x = 0$, niet met amplitude nul "begint", kan geschreven worden als een lineaire kombinatie van een $\sin$ en een $\cos$ die wel op nul beginnen. Uit de "ontbinding" van $s(x)$ in een $\cos$- en een $\sin$-functie kunnen we de fase ϕ van het oorspronkelijke signaal reconstrueren. De coëfficiënten a en b geven aan hoe sterk $s(x)$ lijkt op een $\cos$ en op een $\sin$. Als a veel groter is dan b dan lijkt $s(x)$ meer op een $\cos$, als b veel groter is dan a dan lijkt $s(x)$ meer op een $\sin$. Een numeriek voorbeeld:

$$s(x) = sin(x + \pi/3) \longrightarrow c = 1, \phi = \pi/3, a = 0.5, b = 0.866 \qquad \text{(A.4)}$$

De vraag is nu: hoe komen we in het algemeen te weten hoeveel twee signalen op elkaar lijken? Een antwoord is: je legt ze over elkaar heen en kijkt hoeveel ze elkaar overlappen. Hoe meer ze overlappen, hoe meer ze op elkaar lijken. Je mag eventueel één van de twee signalen nog spiegelen om een betere overlap te krijgen. De wiskundige formulering voor dit *overlappen* is:

- Vermenigvuldig de twee signalen met elkaar.
- Bepaal van de resulterende functie de oppervlakte boven en onder de nul-as.
- Trek deze twee oppervlaktes van elkaar af.
- Vermenigvuldig de uitkomst met een schaalfactor die afhangt van de twee te vergelijken signalen.

Als de uitkomst heel erg klein is dan lijken de signalen niet op elkaar, als de uitkomst groot is dan lijken ze veel op elkaar. In feite bepaalt formule A.1 in één keer hoeveel het signaal $h(t)$ lijkt op de basisfuncties $\sin$ en $\cos$ (het complexe getal $H(f)$ bestaat immers uit twee getallen!).

Tabel A.1: Symmetrieën van de Fourier-transformatie.

Als $h(t)\ldots$	dan is$\ldots$
reëel is	$H(-f) = H^*(f)^2$
imaginair is	$H(-f) = -H^*(f)$
even is	$H(-f) = H(f)$
oneven is	$H(-f) = -H(f)$
reëel en even is	$H(f)$ reëel en even
reëel en oneven is	$H(f)$ reëel en oneven
imaginair en even is	$H(f)$ imaginair en even
imaginair en oneven is	$H(f)$ imaginair en oneven

A.3 Eigenschappen van de Fourier-transformatie

Uit formules A.1 en A.2 kunnen we tabel A.1 afleiden die een aantal symmetrieën geeft tussen het tijd-domein en het frequentie-domein. Veel informatie in deze paragraaf is afkomstig uit Press et al. (1996). Als $h(t)$ en $H(f)$ een Fourier-transformatie-paar zijn, wat we noteren als $h(t) \Longleftrightarrow H(f)$, dan zijn de volgende paren dit ook:

$$h(at) \Longleftrightarrow \frac{1}{|a|}H(\frac{f}{a}) \qquad \text{"tijd-schaling"} \tag{A.5}$$

$$\frac{1}{|b|}h(\frac{t}{b}) \Longleftrightarrow H(bf) \qquad \text{"frequentie-schaling"} \tag{A.6}$$

$$h(t - t_0) \Longleftrightarrow H(f)e^{2\pi i f t_0} \qquad \text{"tijd-verschuiving"} \tag{A.7}$$

$$h(t)e^{-2\pi i f_0 t} \Longleftrightarrow H(f - f_0) \qquad \text{"frequentie-verschuiving"} \tag{A.8}$$

Fourier-paar A.5 drukt uit dat als we de tijd-as uitrekken met een factor a, de frequentie-as met dezelfde factor in elkaar gedrukt wordt. De vorm van het signaal blijft precies hetzelfde, alleen de tijd wordt uitgerekt (als $a > 1$) of in elkaargedrukt (als $a < 1$). Als we bijvoorbeeld een sinusvormig signaal van 1 Hz, waarvan één periode 1 s duurt, met een factor twee uitrekken, zodat die ene periode nu 2 s duurt, dan wordt hierdoor de frequentie gehalveerd. De frequentie-schaling in A.6 is dan analoog.

Wanneer we een signaal gaan verschuiven in de tijd zoals in A.7, dan veranderen daardoor natuurlijk de frequenties in het signaal niet. Het enige effect van deze verschuiving is dat de sinus- en cosinuscomponenten bij de nul niet meer precies op 0 en 1 beginnen. Ze hebben een andere fase gekregen. Dit wordt uitgedrukt door de factor $exp(2\pi i f t_0)$.

Wanneer we een signaal vermenigvuldigen met een sinus of een cosinus, zoals bij A.8, dan krijgen we som en verschil frequenties. Immers als $h(t) = \cos(2\pi ft)$ en we vermenigvuldigen dit met een sinus, dan geldt altijd dat:

$$\cos(2\pi ft)\sin(2\pi f_0 t) = \frac{1}{2}\left\{\sin(2\pi(f+f_0)t) - \sin(2\pi(f-f_0)t)\right\}.$$

Door simpelweg te vermenigvuldigen met een sinus kunnen we frequenties omhoog en omlaag transformeren. Deze truc wordt vaak toegepast bij het *multiplexen* van signalen. Een eenvoudig voorbeeld gaat met het telefoonsignaal. De bandbreedte die ons met een analoge telefoon ter beschikking staat is ongeveer 3 kHz. Als de kabel, waarover het gesprek gaat, een bandbreedte heeft van 6 kHz dan zouden er eigenlijk twee telefoongesprekken tegelijkertijd over deze kabel kunnen. Het eerste gesprek gebruikt de frequenties van 0-3000 Hz. Het tweede gesprek zou het gebied van 3000-6000 Hz kunnen gebruiken. Maar, het tweede gesprek bevat ook, net zo als het eerste, alleen frequenties van 0-3000 Hz. Door nu het tweede gesprek te vermenigvuldigen met een frequentie van 3000 Hz, worden alle frequenties in het gesprek met 3000 Hz opgehoogd, en komen daardoor in het gewenste bereik te liggen van 3000-6000 Hz. Nadat we de verschilfrequenties hebben weggefilterd, tellen we het gemodificeerde tweede signaal gewoon op bij het signaal van het eerste gesprek. Aan de ontvangende kant moeten we nu het tweede gesprek weer terug transformeren naar het gebied 0-3000 Hz. Dit kan in principe door weer te vermenigvuldigen met een frequentievan 3000 Hz en daarna alle frequenties boven de 3000 Hz weg te filteren.

A.3.1 De Fourier-transformatie van twee functies

Met twee Fourier-transformatie-paren $h(t) \Longleftrightarrow H(f)$ en $g(t) \Longleftrightarrow G(f)$ zijn een aantal interessante mogelijkheden te maken. Allereerst geldt dat de Fouriertransformatie een *lineaire* transformatie is. Hiermee wordt bedoeld dat "de Fouriertransformatie van de som van twee functies gelijk is aan de som van de afzonderlijke transformaties". De volgende formules beschrijven de lineaire transformatie:

$$\begin{aligned} h(t) + g(t) &\Longleftrightarrow H(f) + G(f) \\ ah(t) &\Longleftrightarrow aH(f) \end{aligned} \tag{A.9}$$

Een hele belangrijke combinatie van twee functies is de *convolutie* die gedefiniëerd is als

$$(g * h)(t) \equiv \int_{-\infty}^{+\infty} g(\tau)h(t-\tau)\,d\tau \tag{A.10}$$

Er geldt dat $g * h = h * g$. De interpretatie van bovenstaande formule is de volgende. Er geldt dat $g * h$ een functie is van de tijd en dat we de berekening van de waarde op een willekeurig tijdstip t_1 op de volgende manier kunnen visualizeren:

- Klap $h(\tau)$ om in de tijd: $h(\tau)$ staat nu "achterstevoren" en is $h(-\tau)$ geworden.
- Verschuif $h(-\tau)$, de omgeklapte $h(\tau)$, over de afstand t_1. Nu is het de functie $h(t_1 - \tau)$ geworden.
- Vermenigvuldig het omgeklapte en verschoven signaal $h(t_1 - \tau)$ hierna met $g(t)$ zodat we krijgen $g(\tau)h(t_1 - \tau)$.
- Bepaal de oppervlakte onder deze product-functie, waarbij delen boven en onder de nullijn tegen elkaar wegvallen. We hebben nu $\int g(\tau)h(t_1 - \tau)\, d\tau$ bepaald.

Resumerend: voor elk tijdstip t_1 waar we de waarde van de convolutie-functie willen bepalen moeten we bovenstaande berekeningen verrichten. Gelukkig ziet de Fourier-transformatie van de convolutie er veel eenvoudiger uit:

$$g * h \Longleftrightarrow G(f)H(f) \qquad \text{"Convolutie theorema"} \tag{A.11}$$

De Fourier-transformatie van de convolutie van g en h in het tijd-domein is gelijk aan het product van de Fourier-transformaties van g en f. Het product $G(f)H(f)$ kunnen we weer interpreteren als de beschrijving van ***filteren*** in het spectrale domein: het is resultaat van een signaal met spectrale representatie $G(f)$ dat door een filter met spectrale representatie $H(f)$ gevoerd wordt. In het tijddomein zeggen we dan: het resultaat van het filteren van signaal $g(t)$ door een filter met *impulsrespons* $h(t)$ is het signaal dat beschreven wordt door de *convolutie* $g * h$.

Een andere belangrijke combinatie is de correlatie die we aangeven met $Corr(g, h)$ en gedefinieerd is als:

$$Corr(g, h)(t) \equiv \int_{-\infty}^{+\infty} g(\tau + t)h(\tau)\, d\tau \tag{A.12}$$

Als g en h reële functies zijn dan hebben we het volgende Fourier-transformatiepaar:

$$Corr(g, h) \Longleftrightarrow G(f)H^*(f) \qquad \text{"Correlatie theorema"} \tag{A.13}$$

Met de de correlatieformule A.12 bepalen we in feite hoeveel twee signalen g en h op elkaar lijken. Dit op elkaar lijken is gedefiniëerd als: leg de signalen over elkaar heen, vermenigvuldig ze met elkaar en bepaal de oppervlakte[3]. Het op-elkaar-lijken kun je dan een functie van de tijd maken door het ene signaal

[3]Zie ook paragraaf A.2 waarin lijken-op behandeld wordt.

telkens een beetje te verschuiven in de tijd en dan weer te kijken hoeveel ze op elkaar lijken. $Corr(g, h)$ is dus een functie van de verschuivings-tijd. Deze tijd wordt in het Engels *lag* genoemd.

Wanneer de twee functies die we correleren *dezelfde* functies zijn dan spreken we over de *autocorrelatie*. Het transform-paar wordt dan:

$$Auto(g) = Corr(g, g) \Longleftrightarrow |G(f)|^2 \qquad \text{"Wiener-Khinchin Theorema"} \quad \text{(A.14)}$$

Omdat de bepaling van de autocorrelatiefunctie ook weer gaat met de standaard overlap-methode van paragraaf A.2, biedt zij bijvoorbeeld de mogelijkheid om periodiciteit in een signaal te meten. Immers, de autocorrelatiefunctie geeft aan hoeveel een (verschoven) versie van het signaal lijkt op de niet verschoven versie. Wanneer het (periodieke) signaal over de periodetijd T verschoven is en je legt dit over de niet verschoven versie, dan overlappen deze natuurlijk veel. De autocorrelatiefunctie zal daarom een maximum vertonen op bij de verschuivingstijd T (en veelvouden hiervan).

A.3.2 Vermogen

Het totale vermogen van een signaal moet onafhankelijk zijn van het feit of we in het tijd-domein dan wel in het frequentie-domein werken. *Parseval's theorema* drukt dit uit:

$$Vermogen \equiv \int_{-\infty}^{+\infty} |h(t)|^2 \, dt = \int_{-\infty}^{+\infty} |H(f)|^2 \, df \qquad \text{(A.15)}$$

Het is belangrijk te weten dat we niet kunnen spreken over hoeveel vermogen er bij een bepaalde frequentie zit, we zullen het altijd moeten hebben over het vermogen in een frequentie-interval of frequentie-band. Hierbij maakt men dan geen onderscheid tussen positieve en negatieve frequenties en beschouwt men alleen positieve frequenties. De Engelse term hiervoor is *one-sided spectral density* en deze is gedefiniëerd als

$$P_h(f) \equiv |H(f)|^2 + |H(-f)|^2 \qquad 0 \leq f < \infty \qquad \text{(A.16)}$$

Voor onze signalen geldt dat ze reëel zijn zodat $H(f) = H(-f)$ en bovenstaande vergelijking resulteert dan in

$$P_h(f) = 2|H(f)|^2 \qquad \text{(A.17)}$$

Volgens Press et al. (1996) ziet men in de literatuur formule A.17 ook wel zonder de factor twee. Zonder deze factor twee komt bij het tekenen het *hele* amplitudespectrum 3 dB *lager* te liggen.

A.4 Het uitrekenen van integralen

Met behulp van bemonstering kunnen we het uitrekenen van integralen benaderen. Een integraal berekent de oppervlakte onder een curve, waarbij de delen die onder en boven de nullijn liggen tegen elkaar wegvallen. De meest gebruikte benadering van een integraal is via de zogenaamde *Riemann*-som: de oppervlakte onder de curve wordt benaderd door de oppervlakte van een aantal (N) rechthoekjes op te tellen. Hoe smaller de rechthoekjes worden, des te beter wordt de benadering van de oppervlakte. We schrijven dan:

$$\int_0^T s(x)\,dx \approx \sum_{k=1}^{N} s(k \cdot dx)\,dx.$$

Omdat we de breedte van de rechthoekje, $dx = T/N$, overal hetzelfde maken, kunnen we de som herschrijven als $dx \cdot \sum_{k=1}^{N} s(k \cdot dx)$. Het verdelen in rechthoekjes van een analoog signaal kunnen we vergelijken met het *bemonsteren* van een signaal. De gemiddelde waarde $\bar{s}$ van een bemonsterd signaal $s(x)$ is gedefiniëerd als $\bar{s} = 1/N \cdot \sum_{k=1}^{N} s(k \cdot dx)$. Als we dit gebruiken dan houden we de volgende benadering over:

$$\int_0^T s(x)\,dx \approx T \cdot \bar{s}.$$

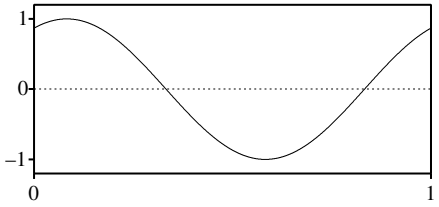
Hier staat dat de functie $s(x)$, geïntegreerd over een interval, op een schalingsfactor na, gelijk is aan het gemiddelde van dit signaal over dit interval. Omdat we in *praat* het gemiddelde van een `Sound`-object over een tijdsinterval op kunnen vragen via `Get mean...` betekent dit dat we nu integralen van functies kunnen uitrekenen. Door een `Sound`-object te maken via een formule bemonsteren we de analoge functie. Het gemiddelde van dit bemonsterde signaal is dan een benadering van de integraal op een factor na. Om de werkelijke oppervlakte te krijgen moeten we met de intervalduur vermenigvuldigen.

A.5 Synthese

A.6 Analyse

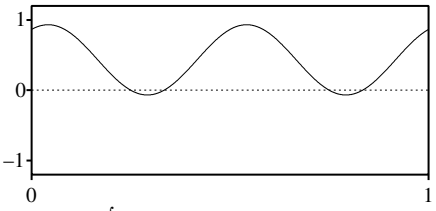
Gewapend met de kennis hoe we integralen uitrekenen gaan we in deze paragraaf een Fourier-analyse uitvoeren ”met de hand”. We zullen stap voor stap een signaal $s(x)$ gaan ontleden in zijn sinus- en cosinus-componenten. Dit proces wordt aanschouwelijk gemaakt in figuur A.2. We starten met het eenvoudige signaal

(a)



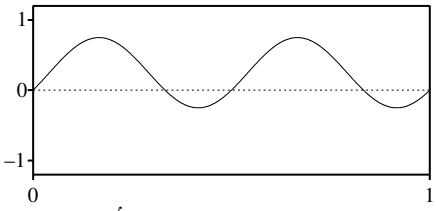
$s(x)=\sin(2\pi\ 1\ x+\pi/3)$

(b)



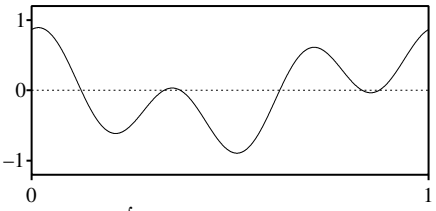
$\int s(x)\cdot\cos(2\pi\ 1\cdot 1\ x) \rightarrow 0.433$

(c)



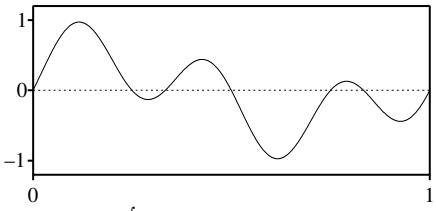
$\int s(x)\cdot\sin(2\pi\ 1\cdot 1\ x) \rightarrow 0.25$

(d)



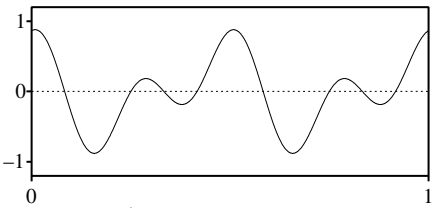
$\int s(x)\cdot\cos(2\pi\ 2\cdot 1\ x) \rightarrow 0$

(e)



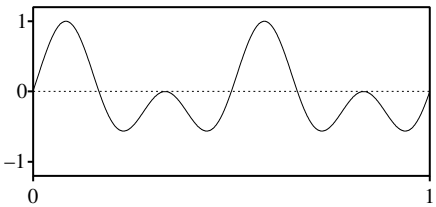
$\int s(x)\cdot\sin(2\pi\ 2\cdot 1\ x) \rightarrow 0$

(f)



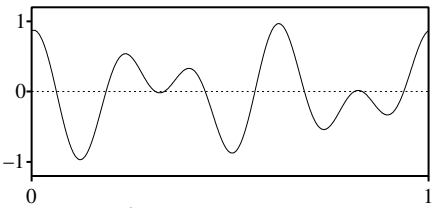
$\int s(x)\cdot\cos(2\pi\ 3\cdot 1\ x) \rightarrow 0$

(g)



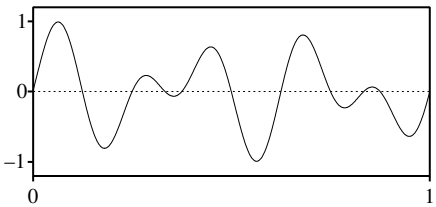
$\int s(x)\cdot\sin(2\pi\ 3\cdot 1\ x) \rightarrow 0$

(h)



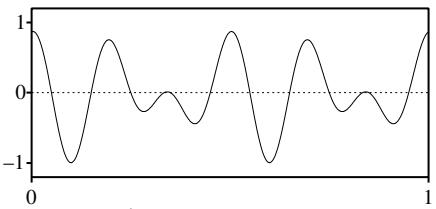
$\int s(x)\cdot\cos(2\pi\ 4\cdot 1\ x) \rightarrow 0$

(i)



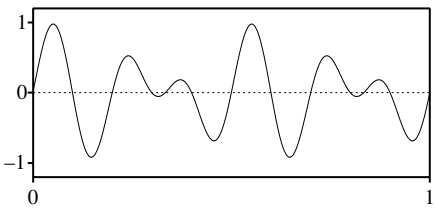
$\int s(x)\cdot\sin(2\pi\ 4\cdot 1\ x) \rightarrow 0$

(j)



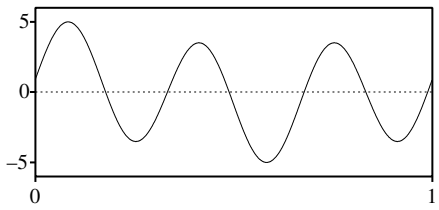
$\int s(x)\cdot\cos(2\pi\ 5\cdot 1\ x) \rightarrow 0$

(k)



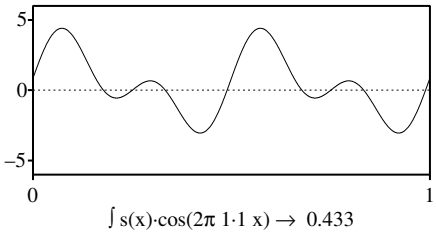
$\int s(x)\cdot\sin(2\pi\ 5\cdot 1\ x) \rightarrow 0$

(a)



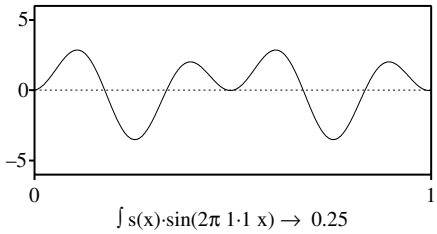
$s(x)=\sin(2\pi\, x+\pi/3)+4\,\sin(2\pi 3x)$

(b)



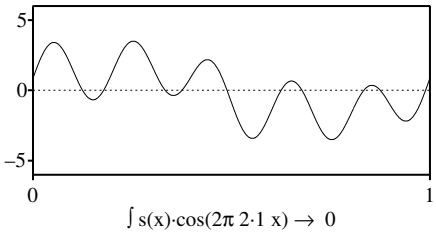
$\int s(x)\cdot\cos(2\pi\, 1\cdot 1\, x) \rightarrow 0.433$

(c)



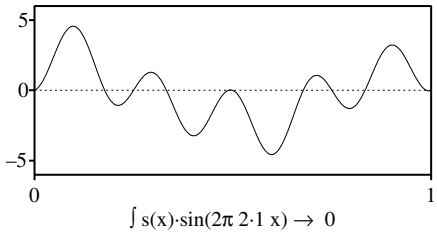
$\int s(x)\cdot\sin(2\pi\, 1\cdot 1\, x) \rightarrow 0.25$

(d)



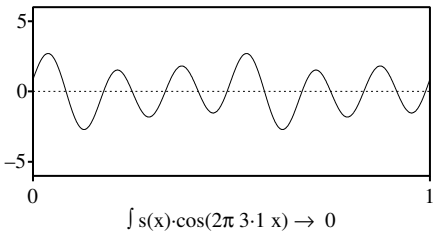
$\int s(x)\cdot\cos(2\pi\, 2\cdot 1\, x) \rightarrow 0$

(e)



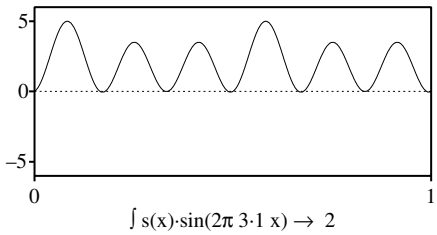
$\int s(x)\cdot\sin(2\pi\, 2\cdot 1\, x) \rightarrow 0$

(f)



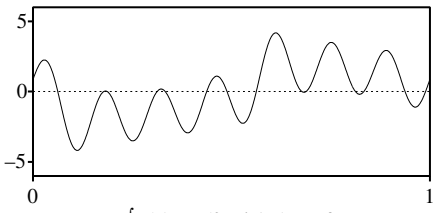
$\int s(x)\cdot\cos(2\pi\, 3\cdot 1\, x) \rightarrow 0$

(g)



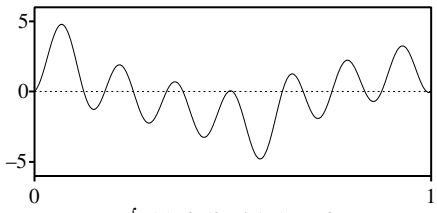
$\int s(x)\cdot\sin(2\pi\, 3\cdot 1\, x) \rightarrow 2$

(h)



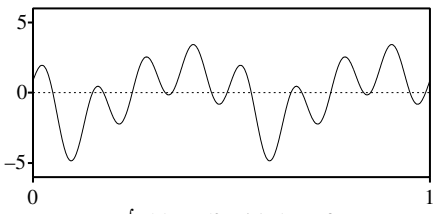
$\int s(x)\cdot\cos(2\pi\, 4\cdot 1\, x) \rightarrow 0$

(i)



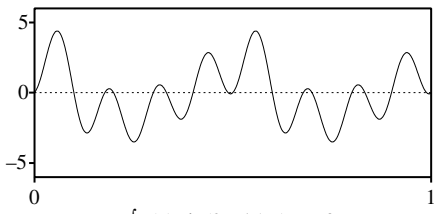
$\int s(x)\cdot\sin(2\pi\, 4\cdot 1\, x) \rightarrow 0$

(j)



$\int s(x)\cdot\cos(2\pi\, 5\cdot 1\, x) \rightarrow 0$

(k)



$\int s(x)\cdot\sin(2\pi\, 5\cdot 1\, x) \rightarrow 0$

$s(x) = sin(2\pi 1x + \pi/3)$, een sinus die $\pi/3$ in fase verschoven is. Om te bepalen hoe sterk een component in $s(x)$ aanwezig is moeten we de procedure volgen zoals beschreven op paragraaf 20. Dit betekent vermenigvuldigen van $s(x)$ met de component en daarna de oppervlakte bepalen. Als componenten moeten we hier sinussen en cosinussen nemen waarvan de frequenties harmonischen zijn van f_0. Deze f_0 is gerelateerd aan de duur van het signaal dat we willen analyseren. Stel dat het signaal T secondes lang is, dan is $f_0 = \frac{1}{T}$. In figuur A.2 hebben we een signaal van 1 s, daarom nemen we als f_0 hier 1 Hz. De analyse gaat als volgt:

1. Begin met $k = 1$
2. Vermenigvuldig $s(x)$ met $\cos(2\pi\, k \cdot f_0\, x)$. Dit resulteert in een signaal $p_c(x) = s(x)\cos(2\pi\, k \cdot f_0\, x)$.
3. Bepaal de oppervlakte onder $p_c(x)$ door de truc met `Get mean...`
4. Vermenigvuldig $s(x)$ met $\sin(2\pi\, k \cdot f_0\, x)$. Dit resulteert in een signaal $p_s(x) = s(x)\sin(2\pi\, k \cdot f_0 f_0\, x)$.
5. Bepaal de oppervlakte onder $p_s(x)$ door de truc met `Get mean...`
6. Hoog k één op: $k = k + 1$.
7. Stop als k groter is dan 5, anders begin weer bij 2.

In figuur A.2.b zien we het signaal $p_c(x) = s(x)cos(2\pi\, k \cdot f_0 f_0\, x)$. Het is duidelijk te zien dat de curve $p_c(x)$ veel meer boven de nullijn ligt dan eronder en dat daarom de "oppervlakte" groter dan nul zal zijn. De uitkomst van `Get mean...` geeft het getal 0.433. Dit is, op een factor 2 na, gelijk aan de uitkomst van het numerieke voorbeeld in formule A.4. In figuur A.2.c zien we het resultaat van de vermenigvuldiging van $s(x)$ met een sinus van 1 Hz. Ook hier is de oppervlakte van het deel van de curve boven de nullijn veel groter dan van het deel onder de nullijn. De totale oppervlakte boven de nullijn is echter kleiner dan in figuur A.2.b. Dit resulteert dan ook in een kleiner "gemiddelde": 0.25, in overeenstemming met het numerieke voorbeeld A.4.

In Figuur A.2.d en A.2.e zijn de componenten met een frequentie van $2 \cdot f_0$ aan de beurt. We zien hier dat de oppervlaktes boven en onder de nullijn niet veel verschillen. Ze blijken precies gelijk te zijn zodat de integraal nul oplevert. deze termen geven dus geen bijdrage: er zit geen sinus-component en ook geen cosinus-component met frequentie $2 \cdot f_0$ in $s(x)$. Als we verder gaan, met nog hogere frequentie-componenten, dan blijken deze allemaal geen bijdrage meer te leveren, zoals in de plaatjes (f) tot en met (k) ook duidelijk wordt. Er blijken maar

twee basisfuncties aanwezig te zijn. Als we de schaalfactor 2 verdisconteren dan is $s(x)$ dus ontbonden op de volgende manier:

$$s(x) \approx 0.866\cos(2\pi 1 \cdot f_0 x) + 0.5\sin(2\pi 1 \cdot f_0 x)$$

We hadden dit resultaat natuurlijk ook meteen kunnen voorspellen uit de manier waarop we $s(x)$ gemaakt hebben. Het is mooi dat de analyse onze berekening in voorbeeld A.4 bevestigt.

In figuur A.3 is eenzelfde soort voorbeeld uitgewerkt als hierboven maar nu is $s(x)$ uitgebreid met een extra sinus-component van een hogere frequentie:

$$s(x) = sin(2\pi 1x + \pi/3) + 4\sin(2\pi 3x).$$

De figuur maakt duidelijk dat we hier alleen bijdrages hebben in de plaatjes (b), (c) en (g). De analyse geeft, als we de factor 2 weer verdisconteren de volgende oplossing:

$$s(x) = 0.866\cos(2\pi 1 \cdot f_0 x) + 0.5\sin(2\pi 1 \cdot f_0 x) + 4\sin(2\pi 3 \cdot f_0 x)$$

Het toevoegen van de extra sinus-component met frequentie $3 \cdot f_0$ heeft geen invloed op de grootte van de bijdrages voor de $1 \cdot f_0$ componenten.

A.7 De discrete Fourier-transformatie

De discrete Fourier-transformatie, afgekort tot DFT, is de Fourier-transformatie toegepast op bemonsterde signalen. Bemonsterde signalen kunnen we, zoals we weten, weergeven als een reeks getallen, de monsterwaardes. In het algemene geval kun je deze monsterwaardes weergeven als:

$$h_k = h(kT) \qquad k = \ldots, -3, -2, -1, 0, 1, 2, 3, \ldots \tag{A.18}$$

In deze formule is T de bemonsteringstijd, dit is de tijd tussen twee opeenvolgende bemonstereringen en k een geheel getal. De bemonsteringstijd is de inverse van de bemonsteringsfrequentie. De representatie van een bemonsterd signaal in *praat* is iets aangepast:

$$h_k = h(T_0 + (k-1)T) \qquad k = 1, 2, 3, \ldots$$

In *praat* liggen natuurlijk de monster ook op afstanden T van elkaar, maar de eerste monsterwaarde kennen we toe aan tijdstip T_0. In *praat* kennen we de monsterwaarde toe aan het *midden* van een tijdsinterval van lengte T. Het allereerste interval loopt meestal van $t = 0$ tot $t = T$, het midden van dit interval ligt dus op

$T/2$. Wij wijzen er met nadruk op dat dit alleen een *interpretatiekwestie* is en dat het geen invloed heeft op de rekenregels die we hierna zullen ontwikkelen. Een bemonsterd signaal blijft voor de meeste rekentrucs gewoon een rijtje van N getallen. Om deze N getallen te indexeren kiest men in de analyseliteratuur meestal een index die van 0 tot $N-1$ loopt in plaats van 1 tot N. Voor de uitkomsten is dit irrelevant[4].

Omdat we geen informatie uit het niets tevoorschijn kunnen toveren, kan het spectrum $H(f)$ van een bemonsterd signaal bestaande uit N monsterwaardes ook niet meer dan uit N onafhankelijke getallen bestaan. Het heeft daarom ook geen zin om naar meer dan N getallen te zoeken. Omdat negatieve frequenties wel degelijk meedoen, verdelen we het frequentie-interval tussen de negatieve Nyquist-frequentie, $-f_c$, tot de positieve Nyquist-frequentie, $+f_c$, in N stukjes en zoeken de waardes van het spectrum op de dicrete frequentie-punten:

$$f_k = \frac{k}{NT}, \qquad k = \frac{-N}{2}, \ldots, \frac{N}{2} \tag{A.19}$$

Strikt genomen zijn dit $N+1$ punten i.p.v. N, maar de waardes van het spectrum op de twee extreme frequentie-punten $f_{\pm\frac{N}{2}}$ zijn niet onafhankelijk van elkaar maar gelijk zoals we verderop zullen zien.

Om het spectrum op deze discrete frequentie-punten uit te rekenen uit het bemonsterde signaal gaan we de integraal in vergelijking A.1 benaderen via rechthoekjes[5]:

$$H(f_n) = \int h(t) e^{-2\pi i f_n t}\, dt \approx \sum_{k=0}^{N-1} h_k T e^{-2\pi i f_n t_k} = T \sum_{k=0}^{N-1} h_k e^{-2\pi i k n/N} \tag{A.20}$$

In de laatste stap van bovenstaande vergelijking is gebruik gemaakt van vergelijkingen A.18 en A.19. We definiëren nu H_n, de *discrete Fourier-transformatie* van de punten h_k als volgt:

$$H_n = \sum_{k=0}^{N-1} h_k e^{-2\pi i k n/N} \tag{A.21}$$

Ook hier geldt weer de opmerking die we ook gemaakt hebben bij vergelijkingen A.1 en A.2 over het teken van de exponent. Vergelijking A.21 beschrijft simpelweg de afbeelding van N (mogelijk complexe) getallen h_k naar N andere complexe getallen, de H_n's. Deze afbeelding is *onafhankelijk* van de tijd tussen twee bemonsteringen.

[4] In de algorithmes in *praat* worden rijtjes meestal geïndexeerd van 1 tot N. Het eerste monster in een Sound-object heeft dan ook index 1.

[5] Zie ook paragraaf A.4 over het uitrekenen van integralen waar we ook zo iets doen.

Tabel A.2: Symmetrieën van de Fourier-transformatie.

Als $h_h\ldots$	dan is…
reëel is	$H_{N-n} = H_n^*$
imaginair is	$H_{N-n} = -H_n^*$
even is	$H_{N-n} = H_n$
oneven is	$H_{N-n} = -H_n$
reëel en even is	H_n reëel en even
reëel en oneven is	H_n reëel en oneven
imaginair en even is	H_n imaginair en even
imaginair en oneven is	H_n imaginair en oneven

De formule om uit de spectrale waardes weer het bemonsterde signaal te reconstrueren wordt de *inverse* DFT genoemd:

$$h_k = \frac{1}{N}\sum_{n=0}^{N-1} H_n e^{+2\pi i k n/N} \tag{A.22}$$

De formules A.21 en A.22 vormen samen, analoog aan de continue Fourier-transformatie, een DFT-paar[6].

De symmetrie-eigenschappen in tabel A.1 van de continue Fourier-transformatie gelden ook voor de DFT. Bij bemonsterde signalen vertalen we de term ”even” functie door $h_k = h_{N-k}$ en ”oneven” functie door $h_k = -h_{N-k}$. De symmetrie-eigenschappen van de discrete Fourier-transformatie staan vermeld in tabel A.2. Ook bij de discrete —FT geldt het Parseval-theorema:

$$\sum_{k=0}^{N-1} |h_k|^2 = \frac{1}{N}\sum_{n=0}^{N-1} |H_n|^2 \tag{A.23}$$

Discrete versies van convolutie en correlatie bestaan natuurlijk ook. Behandeling hiervan stellen we uit tot na de paragraaf A.8 waarin we een snel algoritme behandelen om de DFT uit te rekenen.

[6]De index n in A.21 zou volgens A.19 lopen van $-N/2$ tot $N/2$. Uit formule A.21 is duidelijk te zien dat H_n periodiek is in n, met periode N. Dit betekent dat $H_{-n} = H_{N-n}$ voor $n = 1, 2, \ldots$. We kunnen daarom gerust onze indexering van 0 tot $N-1$ laten lopen door als het ware het spectrum $N/2$ waardes op te schuiven. Dit heeft tot gevolg dat frequentie nul overeenkomt met index $n = 0$ en dat de positieve frequenties, $0 < f < f_c$, corresponderen met de indexen $1 \le n \le N/2 - 1$. De negatieve frequenties, $-f_c < f < 0$, corresponderen dan met de indexen $N/2 + 1 \le N - 1$. De waarde bij index $N/2$ correspondeert dan met zowel de frequenties $f = f_c$ als ook met $f = -f_c$.

A.8 De FFT

In formule A.21 staat het voorschrift om uitgaande van N monsterwaardes h_k de N frequentie-componenten H_n te berekenen. Met formule A.22 kun je uit de N frequentie-componenten H_n verliesloos de N monsterwaardes reconstrueren. We merken op het enige wezenlijke verschil tussen de twee formules het teken van de exponent is, als we even afzien van de schalingsfactor $1/n$ in A.22. Dit betekent dat een algoritme dat de DFT berekent volgens voorschrift A.21 met één simpele wijziging ook de inverse DFT zou kunnen berekenen.

We stellen nu de vraag hoeveel berekeningen een algoritme dat een DFT uitrekent moet maken. Om het, tot in de zestiger jaren, standaard antwoord hierop te kunnen geven gaan we eerst even formule A.21 iets anders schrijven en definiëren

$$W \equiv e^{-2\pi i/N} \tag{A.24}$$

Dan kunnen we A.21 schrijven als:

$$H_n = \sum_{k=0}^{N-1} W^{nk} h_k \tag{A.25}$$

We kunnen deze vergelijking nu interpreteren als een matrix-vergelijking: aan de rechterkant staat dan de vermenigvuldiging van een $N \times N$ matrix met een N-dimensionale vector. Het element op n-de rij en de k-de kolom wordt gegeven door W tot de macht $n \times k$, het k-de element van de vector is h_k. Het uitrekenen van 1 getalletje H_n kost dan N vermenigvuldigingen en optellingen. Het uitrekenen van de N spectrale waardes van H kost dan minstens $N \times N$ berekeningen. We zeggen daaron dat het uitrekenen van de Fourier-transformatie via formule A.25 een proces is van orde N^2 en noteren dit als $O(N^2)$. Een berekening is van $O(N^2)$ als het aantal berekeningen kwadratisch toeneemt met de grootte van de invoer. Dit betekent dat elke ver*dubbel*ing van het aantal getallen waarop men een berekening wil doen een ver*vier*voudiging van de rekenlast betekent.

Efficiënte algoritmes waren tot begin zestiger jaren geen gemeengoed. Pas toen Cooley en Tukey hun algoritme voor het sneller uitrekenen van de DFT publiceerden kwam de signaalverwerking in een stroomversnelling. Dit algoritme heet de FFT, het staat voor *Fast Fourier Transform*, en het is $O(Nlog_2N)$.

Het verschil tussen $O(N^2)$ en $O(Nlog_2N)$ is gigantisch. Stel we hebben een computer die 1 μs per operatie nodig heeft[7] en we willen de Fourier-transformatie

[7] Een miljoen operaties per seconde betekent, anno 2000, dat het om een hele langzame computer gaat. Het aantal operaties van een standaard buromodel ligt dichter nu dichter bij de honderdmiljoen instructies per seconde. Anno 1965 lag het aantal operaties per seconde op de allersnelste computer dichter bij de honderdduizend. Het uitrekenen van de FFT van een Sound met precies 2^{20} monsters, op mijn standaard thuiscomputer met een 350 Mhz processor en 64MB geheugen duurt 10 s.

van 10^6 monsters uitrekenen. Als we een orde N^2 algoritme gebruiken dan hebben we aan tijd nodig: $10^6 \times 10^6 \times 1\mu s = 10^6 s$. Er gaan $24(uur) \times 60(minuten) \times 60(s) = 86400$ secondes in een dag. We hebben met dit algoritme een rekenklus van ruim 11.5 dag. Het $O(Nlog_2N)$-algoritme doet het in $10^6 \times 20 \times 1\mu s$, nog geen 20 secondes!

Om de werking van het algoritme te verklaren laten we eerst zien dat je elke DFT van lengte N kunt herschrijven als de som van twee DFT's van de halve lengte. Eén van de twee maak je uit de monsters met even index en de andere van de monsters met oneven index op de volgende manier. We beginnen met de DFT-formule A.21

$$H_n = \sum_{k=0}^{N-1} h_k e^{-2\pi ikn/N}$$

Nu nemen we de $N/2$ even termen en de $N/2$ oneven termen bij elkaar en krijgen dan

$$H_n = \sum_{k=0}^{N/2-1} h_{2k} e^{-2\pi i(2k)n/N} + \sum_{k=0}^{N/2-1} h_{2k+1} e^{-2\pi i(2k+1)n/N}$$

In de tweede term kunnen we uit de exponent de factor $e^{2\pi i1n/N}$, die immers niet afhankelijk is van de sommatie-index k, buiten het som-teken halen. We herschrijven deze term met behulp van formule A.24 als W^n:

$$H_n = \sum_{k=0}^{N/2-1} h_{2k} e^{-2\pi i(2k)n/N} + W^n \sum_{k=0}^{N/2-1} h_{2k+1} e^{-2\pi i(2k)n/N}$$

Een eerste cosmetische truc, de term $(2k)/N$ in de beide exponenten schrijven we als $k/(N/2)$:

$$H_n = \sum_{k=0}^{N/2-1} h_{2k} e^{-2\pi ikn/(N/2)} + W^n \sum_{k=0}^{N/2-1} h_{2k+1} e^{-2\pi ikn/(N/2)}$$

Nog wat cosmetica, we schrijven $N' = N/2$ en definiëren $e_k = h_{2k}$ en $o_k = h_{2k+1}$ en dan:

$$H_n = \sum_{k=0}^{N'-1} e_k e^{-2\pi ikn/N'} + W^n \sum_{k=0}^{N'-1} o_k e^{-2\pi ikn/N'}$$

De twee termen die we overhouden lijken precies op vergelijking A.21, de term waarmee we zijn begonnnen. We kunnen niet anders dan concluderen dat deze twee termen beide een DFT beschrijven en we verkrijgen het *Danielson Lanczos Lemma* dat laat zien dat je het probleem van het uitrekenen van één grote DFT

kunt reduceren tot het uitrekenen van twee kleinere DFT's telkens op een andere helft van de monsters.

$$H_n = H_n^e + W^k H_n^o, \tag{A.26}$$

waarbij H_n^e de DFT is van lengte $N/2$ van de even componenten van de originele h_k's en H_n^o de DFT is van lengte $N/2$ van de oneven componenten. De index k loopt van 0 tot N, terwijl de H_n^e en H_n^o DFT's zijn met een periode van $N/2$ in k. Omdat het uitrekenen van de twee termen in refeq:danielson-lanczos beide ongeveer orde $O((\frac{N}{2})^2)$ operaties vergt, is voor het uitrekenen van het rechter deel van deze vergelijking maar ongeveer de ***helft*** van het aantal operaties nodig als die voor het uitrekenen via vergelijking A.21

Het mooie van bovenstaande reductie A.26 is dat we het recursief kunnen toepassen en daardoor elke keer bijna een factor twee in het aantal operaties besparen. We kunnen de $N/4$ even en de $N/4$ oneven indexen van H_n^e bij elkaar nemen om de H_n^{ee} en de H_n^{eo} te bepalen. Evenzo goed kunnen we de oneven-even en de oneven-oneven componenten H_n^{oe} en H_n^{oo} bepalen uit de $N/4$ even en oneven indexen van H_n^o. De resulterende vier kunnen we weer stuk voor stuk behandelen volgens bovenstaand reductie-principe en zo voorts, net zo lang totdat we, $\log_2 N$ stappen diep, uitkomen op transformaties van lengte 1. Deze laatste operatie, de Fourier-transformatie van lengte 1 is gelijk aan de monsterwaarde.

$$H_n^{eooeo\cdots oee} = h_k \tag{A.27}$$

Om uit te vinden welke index k correspondeert met een gegeven patroon van e's en o's in A.27 gebruiken we de volgende truc: keer het patroon van e's en o's in A.27 om, substitueer $e = 0$ en $o = 1$ en het resultaat is de index k in ***binaire*** vorm.

De structuur van een FFT-algoritme bestaat uit twee delen.

1. Sorteer de monsters in ***bit-omkeer*** volgorde. De indexen worden "omgezet naar binaire code" en de bits worden achterstevoren gezet. Bijvoorbeeld als $N = 16$ dan wordt index $k = 1$,binair is dit 0001, omgezet naar binair 1000, wat overeenkomt met $k = 8$. Uit dit voorbeeld zien we al dat de bit-omkeer volgorde eenvoudig uit de oorspronkelijke volgorde af te leiden is via het paarsgewijs verwisselen van monsterswaardes.

2. Het berekenen van transformaties van orde 2, 4, 8,..., N.

De FFT werkt alleen als het aantal monsterwaardes een macht van 2 is. Als het aantal monsters dit niet is, dan is een eenvoudig recept om de dichtstbijzijnde hogere macht van 2 te pakken, en het signaal aan te vullen met nullen.

A.9 De Fourier-transformatievan een blokfunctie

We willen de Fourier-transformatie uitrekenen van de rechthoekige blokfunctie gedefiniëerd als

$$h(x) = \begin{cases} 1 & -c \leq x \leq c \\ 0 & x < -c \text{ of } x > c \end{cases} \tag{A.28}$$

Wanneer we dit invullen in vergelijking A.1 krijgen we:

$$\begin{aligned} H(f) &= \int_{-\infty}^{+\infty} h(x) e^{-2\pi i f x}\, dx \\ &= \int_{-c}^{+c} h(x) e^{-2\pi i f x}\, dx \\ &= \frac{1}{-2\pi i f} \left[e^{-2\pi i f x}\right]_{-c}^{+c} \\ &= \frac{1}{-2\pi i f} [e^{-2\pi i f c} - e^{+2\pi i f c}] \\ &= \frac{sin(2\pi f c)}{\pi f} \qquad \text{(A.29)} \\ &= 2c\, \frac{sin(2\pi f c)}{2\pi f c} \qquad \text{(A.30)} \end{aligned}$$

Uit vergelijking A.29 zien we dat de nulpunten van de functie $H(f)$ gelijk zijn aan de nulpunten van de $\sin(2\pi f c)$ in de teller. We kunnen derhalve stellen dat $2\pi f c = k\pi$, zodat de nulpunten van $H(f)$ liggen op $f = k/(2c)$, voor $k = \pm 1, \pm 2, \ldots$. De afstand tussen de eerste nulpunten, links en rechts van de nul, losjesweg ook wel de ”breedte” van de piek genoemd, is $1/c$. De blokfunctie en zijn spectrum zijn weergegeven in figuur A.4. Omdat onze blokfunctie in het tijddomein een *even* functie is, is het spectrum $H(f)$ reëel en bestaat alleen uit cosinus-termen. We zien een grote bijdrage van frequentie-termen in het centrum rond de 0 Hz, deze bijdrage neemt bij groter wordende frequenties snel af met een factor $1/f$.

De typische vorm van het spectrum wordt een ”sinus-x-over-x” vorm genoemd vanwege zijn langzaam kleiner wordende sinusvormige amplitude.

Er is een inverse relatie tussen de *breedte* van de blokfunctie in het tijddomein (c) en de ”breedte” van de ”sinus-x-over-x” in het spectrale domein ($1/c$). Figuur A.5 laat zien hoe de ”sinus-x-over-x” verandert als de breedte van de blokgolf verandert. In feite representeert deze figuur het effect van een *eindige* meettijd op het spectrum. Om een frequentie heel goed te meten hebben we een lange

meettijd nodig: als we c heel groot maken dat gaat de "sinus-x-over-x" steeds meer op een piek lijken. Als de meettijd kort wordt, c wordt klein, dan smeert de "sinus-x-over-x" zijn frequenties steeds verder uit over het spectrum.

A.10 Het bemonsteringstheorema van Shannon

Als een continue signaal $h(t)$ bandgelimiteerd is, dat wil zeggen er komen geen frequenties groter dan een zekere f_c in voor ($H(f) = 0$ voor alle $f \geq f_c$), dan kan het continue signaal $h(t)$ volledig bepaald worden uit het bemonsterde signaal (als de minimale bemonsteringsfrequentie maar groter is dan $2f_c$). De formule om $h(t)$ uit de monsterwaardes h_n te reconstrueren is:

$$h(t) = T \sum_{n=-\infty}^{n=-\infty} h_n \frac{sin[2\pi f_c(t - nT)]}{\pi(t - nT)} \tag{A.31}$$

We kunnen de waarde van $h(t)$ op elk tijdstip t vinden uit de monsterwaardes door naburige monsterwaardes te interpoleren met een functie die weer van de vorm "sinus-x-over-x" is. Een bandgelimiteerd signaal kunnen we dus zo bemonsteren dat alle *informatie* behouden blijft. Door het bandgelimiteerde signaal alleen te representeren op discrete tijdstippen is toch *alle* informatie behouden gebleven. Formule A.31 geeft het interpolatievoorschrift om uit de representatie op de discrete tijdstippen de waarde op elk ander tijdstip te bepalen.

Een van de toepassingen van deze formule is herbemonstering met een andere bemonsteringsfrequentie. Wanneer de nieuwe bemonsteringstijd T' is dan kunnen we met behulp van formule A.31 de monsterwaardes op de nieuwe tijdstippen berekenen:

$$h(kT') = T \sum_{n=-\infty}^{n=-\infty} h_n \frac{sin[2\pi f_c(kT' - nT)]}{\pi(kT' - nT)} \tag{A.32}$$

Als $T' > T$, de nieuwe bemonsteringsfrequentie is lager dan de oude, moeten we om vouwvervorming te voorkomen eerst filteren om de bandbreedte van het signaal beperken. Als de nieuwe bemonsteringsfrequentie hoger is dan de oude hoeft er niet eerst gefilterd te worden.

A.11 De Fourier-transformatie als filterbank

We kunnen de Fourier-transformatie ook beschouwen als een filterbank. Een filterbank is een verzameling filters die parallel staan en allemaal hetzelfde invoersignaal aangeboden krijgen. Elk filter is van het type banddoorlaat en afgesteld

op een ander frequentie-bandje[8]. Deze afstelling, de filterfunctie, heeft weer een "sinus-x-over-x" karakteristiek (zie paragraaf A.9). Stel we maken een Fourier-transformatie van een stukje van lengte T van een sinus met frequentie f. Het spectrum hiervan is van de vorm "sinus-x-over-x", gecentreerd rond de frequentie f met nulpunten op afstanden $\Delta f = 1/2T$. Wanneer we dit spectrum nu gaan interpreteren als de uitvoer van een filterbank, dan moeten we constateren dat behalve het filter dat staat afgesteld op het frequentie-bandje gecentreerd rond de frequentie f ook andere filters uitvoer geven. Geïnterpreteerd als filterbank blijken alle filters een "sinus-x-over-x" karakteristiek te hebben. Zij zijn dus niet alleen gevoelig in het gebiedje $[f - \Delta f, f + \Delta f]$, maar ook voor frequenties die buiten dit gebied liggen. Het zal nu niemand meer verwonderen dat deze filterfunctie ook weer een "sinus-x-over-x" vorm heeft.

A.12 Vensteren

Het bepalen van het spectrum van signalen die abrupt beginnen en eindigen introduceert ongewenste frequenties in het spectrum (zie bijvoorbeeld paragraaf 4.5). Meestal wanneer we we analyses doen van opeenvolgende, al dan niet overlappende (korte) stukjes uit een spraaksignaal dan zullen deze stukjes abrupt eindigen en beginnen. Om de ongewenste effecten die met het abrupte beginnen en eindigen wat te temperen gebruiken we in deze gevallen een zogenaamd *vensterfunctie*. Een andere reden om te vensteren is dat we niet tevreden zijn met de filterfunctie van het rechthoekige venster. De vensterfunctie wordt meestal gedefiniëerd op een tijdinterval van lengte T. De functie is meestal symmetrisch om het midden van het interval, waar hij zijn maximale waarde bereikt. Vanaf het midden loopt hij dan meestal geleidelijk af naar de beide extreme punten van het interval. Tabel A.3 geeft een overzicht van de meest gebruikte vensterfuncties in *praat*. In figuur A.6 staan deze meest gebruikte vensterfuncties in *praat* afgebeeld, samen met hun spectra. Om een duidelijkere (symmetrische) afbeelding te krijgen is elk spectrum gecentreerd rond de 1000 Hz. Dit kan heel eenvoudig door de vensterfunctie te vermenigvuldigen met een sinus van 1000 Hz.

A.13 Opgaves

1. Fourier-analyse.

In paragraaf A.4 hebben we laten zien hoe we integralen kunnen uitrekenen.

[8] In tegenstelling tot een "echte" filterbank, waarbij een vast aantal filters met een vaste gegeven bandbreedte aanwezig zijn, hangen bij deze interpretatie het aantal filters en hierdoor ook de bandbreedte van elk filter af van de meettijd.

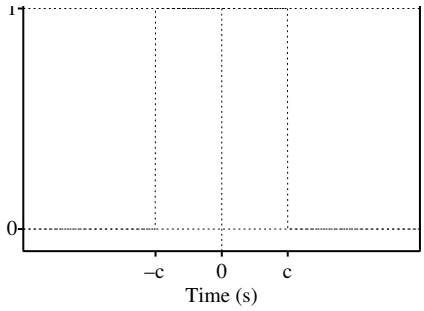
Tabel A.3: Een aantal gangbare vensterfuncties gedefinieerd op het interval $[0, T]$. Zie ook figuur A.6 voor een grafische voorstelling.

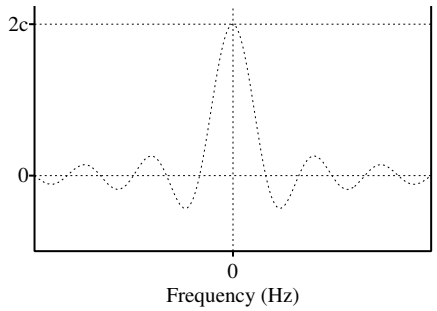
Naam	$f(t)$	-3dB breedte	sidelobe (dB)
Rechthoekig	1		-14
Hamming	$0.54 + 0.46\cos(2\pi * (t/T - 1/2))$		-42
Bartlett	$1 - \lvert 2t/T - 1\rvert$		-26
Welch	$1 - (2t/T - 1)^2$		-21
Hanning	$\frac{1}{2}(1 + \cos(2\pi * (t/T - 1/2)))$		-31
Gaussisch	$\frac{e^{-\left(\frac{t-T/2}{T/4}\right)^2} - e^{-4}}{1-e^{-4}}$	$\frac{2\sqrt{2\ln 6}}{\pi T}$	n.v.t.

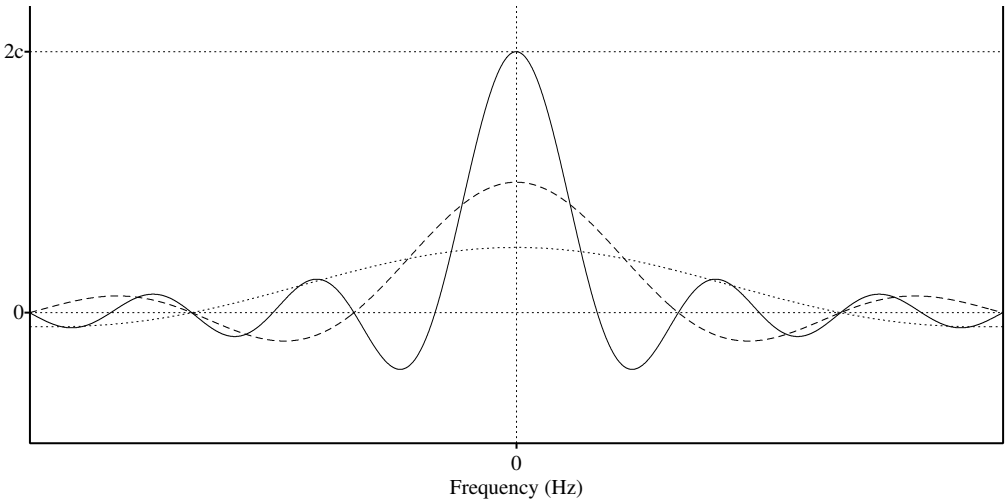
Maak het volgende signaal $s(x)$ (duur 1 s, bemonsteringsfrequentie is 22050 Hz):

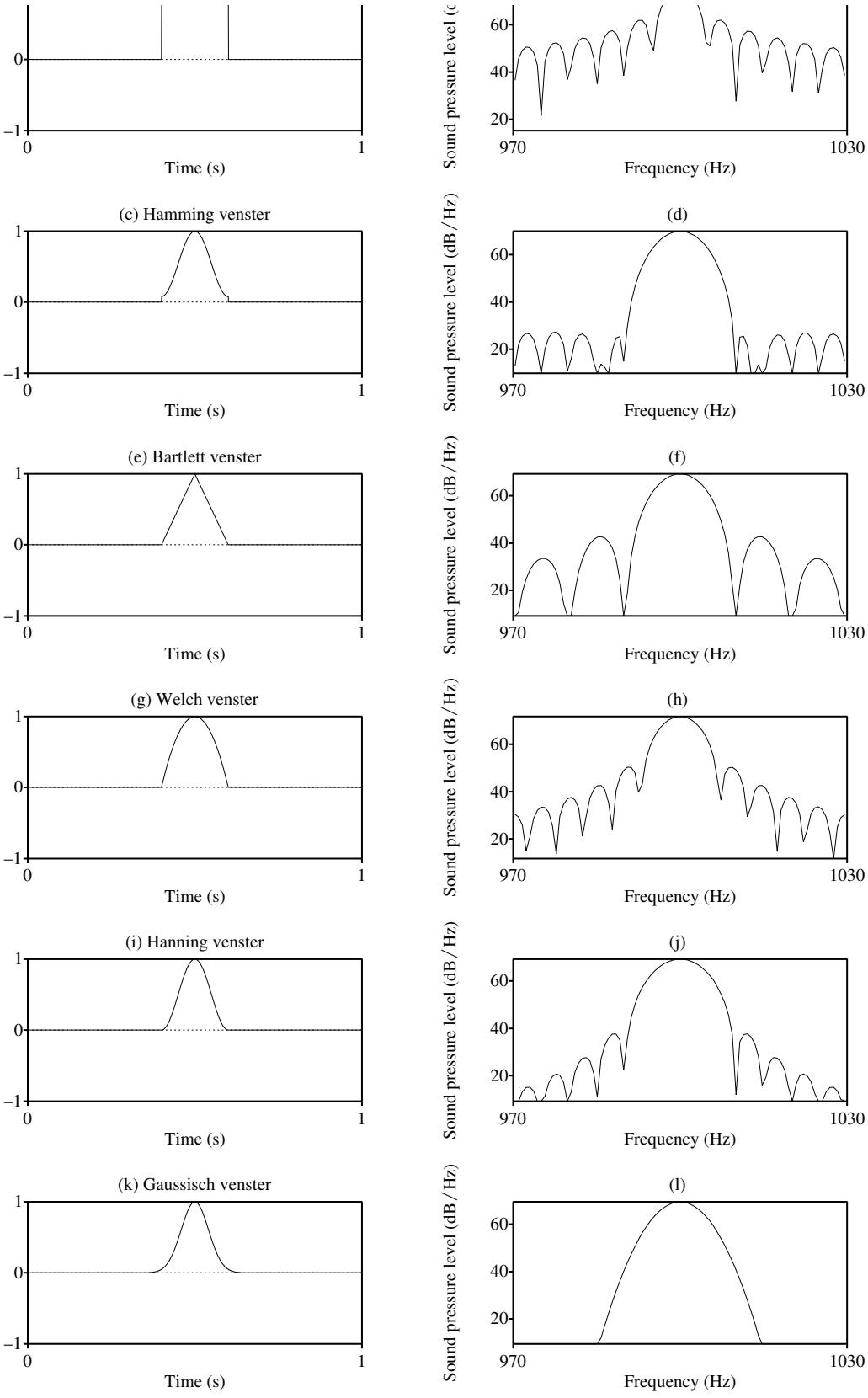
$$s(x) = sin(2\pi 2x) + sin(2\pi 3x + \pi/3)/3 + sin(2\pi 4x)/4$$

Ontleed dit signaal in zijn componenten met behulp van Fourier-analyse: vermenigvuldig hiervoor $s(x)$ met basisfuncties $\cos 2\pi kx$ en $\sin 2\pi kx$ en noteer de sterktes. Kies $k = 1, 2, 3, 4, 5$ Maak voor het te ontleden signaal $s(x)$ plaatjes met $s(x) \cdot \cos 2\pi kx$ en $s(x) \cdot \sin 2\pi kx$ naast elkaar. Zet het plaatje met het te analyseren signaal $s(x)$ helemaal bovenaan. Zorg dat alle plaatjes op één A4 passen.


1
0
–c
0
c
Time (s)


2c
0
0
Frequency (Hz)


2c
0
0
Frequency (Hz)


0
–1
0
1
Time (s)
Sound pressure level (
60
40
20
970
1030
Frequency (Hz)
(c) Hamming venster
(d)
(e) Bartlett venster
(f)
(g) Welch venster
(h)
(i) Hanning venster
(j)
(k) Gaussisch venster
(l)
Sound pressure level (dB / Hz)

Bijlage B

Digitale filters

B.1 De Z-transformatie

Als we een rijtje monsterwaardes $\{s_n\}$ hebben met bemonsteringstijd T, dan definiëren we de Z-transformatie $S(z)$ van dit rijtje als:

$$S(z) = \sum_{n=-\infty}^{+\infty} s_n z^{-n}, \qquad z = e^{2\pi i f T} \tag{B.1}$$

Voor bemonsterde signalen is dit een handige transformatie. Wil de transformatie zinvol zijn, dat wil zeggen voor elke z een eindig getal opleveren dan moeten we eisen dat $|z| \leq 1$. Voor $|z| \geq 1$ zullen immers de termen z^n een steeds grotere bijdrage geven als n groter wordt. Als $\{s_n\} \Longleftrightarrow S(z)$ een transformatiepaar zijn, dan gelden de volgende dingen:

$$\begin{array}{rrll}
a\{x_n\} + b\{y_n\} & \Longleftrightarrow & aX(z) + bY(z) & \text{"Lineair"} \\
\{x_{n-k}\} & \Longleftrightarrow & z^{-k}X(z) & \text{"Tijdtranslatie"} \\
\{x_{-n}\} & \Longleftrightarrow & X(z^{-1}) & \text{"Tijdinversie"} \\
\{x_n\} * \{y_n\} & \Longleftrightarrow & X(z)Y(z) & \text{"Convolutie"} \\
\{x_n\} - \{x_{n-1}\} & \Longleftrightarrow & (1 - z^{-1})X(z) & \text{"Differentie"}
\end{array} \tag{B.2}$$

Voor het gemak noemen we $S(z)$ het spectrum van het rijtje $\{s_n\}$ (formule B.1 lijkt natuurlijk heel veel op de DFT van een bemonsterd signaal).

B.2 Waarom is deze transformatie zo handig?

Stel we willen de frequentierespons, het spectrum, weten van een digitaal filter. In de meest algemene vorm kan een digitaal filter geschreven worden als:

$$y_n = \sum_{k=0}^{M} c_k x_{n-k} + \sum_{j=1}^{N} d_j y_{n-j}. \tag{B.3}$$

De $M+1$ coëfficiënten c_k en de N coëfficiënten d_j zijn vast en definiëren de filterrespons. Dit filter produceert elke nieuwe uitvoerwaarde y_n met behulp van de huidige *invoer*waarde (x_n), de M vorige invoerwaardes (x_{n-k}) en de N vorige *uitvoer*waardes (y_{n-j}). De uitvoer van het filter is een lineaire combinatie van $M+1$ invoeren x_{n-k} en N vorige uitvoeren y_{n-j}. Voor de signalen waar wij mee werken geldt altijd dat de $c_k \in \mathbf{R}$ en de $d_j \in \mathbf{R}$. We kunnen twee filtercategoriën onderscheiden: de recursieve en de niet-recursieve. We nemen de Z-transform links en rechts van het isgelijk-teken in B.3:

$$\sum_{n=-\infty}^{\infty} y_n z^{-n} = \sum_{k=0}^{M} c_k \sum_{n=-\infty}^{\infty} x_{n-k} z^{-n} + \sum_{j=1}^{N} d_j \sum_{n=-\infty}^{\infty} y_{n-j} z^{-n}$$

We maken gebruik van definitie B.1 en eigenschap B.2:

$$Y(z) = \sum_{k=0}^{M} c_k z^{-k} X(z) + \sum_{j=1}^{N} d_j z^{-j} Y(z)$$

Verzamel gelijke termen

$$Y(z)\left(1 - \sum_{j=1}^{N} d_j z^{-j}\right) = X(z) \sum_{k=0}^{M} c_k z^{-k}$$

Dit geeft als filterrespons

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} c_k z^{-k}}{1 - \sum_{j=1}^{N} d_j z^{-j}} \tag{B.4}$$

Dit is natuurlijk een geweldig resultaat: het spectrum $H(z)$ van een digitaal filter is op een "simpele" manier alleen afhankelijk van zijn coëfficiënten c_k en d_j. Het spectrum $H(z)$ is het quotiënt van twee polynomen in z. Het polynoom in de teller is alleen afhankelijk van de coëfficiënten c_k van de invoer en het polynoom in de noemer alleen van de recursieve coëfficiënten d_j.

B.3 Het middelingsfilter

Een simpel voorbeeld van een digitaal filter is het middelingsfilter dat elke twee opeenvolgende waardes middelt:

$$y_n = (x_n + x_{n-1})/2 \tag{B.5}$$

Dit filter heeft geen recursie, alle coëfficiënten d_j zijn nul, het filter is van het type *Moving Average*. De frequentierespons volgens formule B.4 is, met $c_0 = c_1 = \frac{1}{2}$:

$$H(z) = \frac{1 + z^{-1}}{2} \tag{B.6}$$

We willen graag het amplitude-spectrum($|H(f)|$) berekenen, dit gaat als volgt:

$$\begin{aligned} |H(f)| = |H(z)| = |H(e^{2\pi ifT})| &= \left|\frac{1 + e^{-2\pi ifT}}{2}\right| \\ &= \left|e^{-\pi ifT}\frac{e^{\pi ifT} + e^{-\pi ifT}}{2}\right| \\ &= \cos(\pi fT), \qquad \text{voor} \qquad 0 \le f \le \frac{1}{2T} \end{aligned} \tag{B.7}$$

Wanneer f loopt van 0 tot $1/2T$ dan loopt het argument van de cosinus van o tot $\pi/2$ en de cosinus van 1 naar 0. Dit is een *laagdoorlaat*-karakteristiek.

B.4 Het pre-emfase filter

Een veel gebruikte voorbewerking bij de spraaksignaalanalyse is de pre-emfase. Dit is een filter van de vorm:

$$y_n = x_n - ax_{n-1}, \qquad \text{voor} \qquad 0 < a \le 1. \tag{B.8}$$

Dit is een heel eenvoudige bewerking, trek van elke monsterwaardes een fractie van de vorige af. De frequentie-karakteristiek van dit filter is, met $c_0 = 1, c_1 = -1$:

$$H(z) = 1 - az^{-1} \tag{B.9}$$

Het amplitude-spectrum is dan:

$$\begin{aligned} |H(f)| &= \sqrt{H(f)H^*(f)} \\ &= \sqrt{(1 - ae^{-2\pi ifT})(1 - ae^{+2\pi ifT})} \\ &= \sqrt{1 + a^2 - a(e^{+2\pi ifT} + e^{-2\pi ifT})} \\ &= \sqrt{1 + a^2 - 2a\cos 2\pi fT} \qquad \text{voor} \qquad 0 \le f \le \frac{1}{2T} \end{aligned} \tag{B.10}$$

In het spectrum van een klinkerachtig geluid zijn de hogere frequenties over het algemeen minder sterk aanwezig dan de lagere frequenties. De helling van zo'n klinkerspectrum is ongeveer -6dB/octaaf. Omdat over het algemeen analysemethodes, zoals lineaire predictie (zie C), sterkere pieken (*formanten*) beter beschrijven dan zwakkere zouden zonder correctie de hogere formanten slecht benaderd worden[1]. Door toepassing van een pre-emfasefilter wordt ruwweg de helling van het spectrum met een factor +6dB/octaaf gecorrigeerd. Hierdoor krijgen de formanten weer ongeveer allemaal dezelfde sterkte.

B.5 Het formant filter

De beide voorgaande filters hebben een filterkarakteristiek die min of meer vast ligt: een top (dal) bij de laagste frequentie en een dal (top) bij de hoogste frequentie. Aan de positie van de top valt niet te sleutelen, deze ligt vast. Wat we nu willen is een filter waarmee we een piek op een willekeurige frequentie kunnen leggen én de breedte van de piek kunnen variëren. Uit formule B.35 blijkt dat we het beste een piek kunnen modeleren met *polen*, in filter termen betekent dit dat we dan met een recursief filter te doen hebben. Omdat we *twee* parameters willen modeleren zal het filter ook minstens twee parameters moeten hebben. We gebruiken het volgende digitale formantfilter:

$$y_n = -py_{n-1} - qy_{n-2} + x_n \tag{B.11}$$

Dit is een recursief tweede orde filter met frequentierespons:

$$H(z) = \frac{1}{1 + pz^{-1} + qz^{-2}} = \frac{z^2}{z^2 + pz + q} \tag{B.12}$$

De noemer is een tweede graads polynoom in z en heeft derhalve twee nulpunten. Deze zijn:

$$z_{1,2} = -\frac{p}{2} \pm \sqrt{\frac{p^2}{4} - q} \tag{B.13}$$

We kunnen de noemer dan ontbinden in factoren zodat de filterfunctie dan wordt:

$$H(z) = \frac{1}{(z - z_1)(z - z_2)} \tag{B.14}$$

De amplituderespons krijgen we door het nemen van absolute waardes:

$$|H(z)| = \frac{1}{|(z - z_1)||(z - z_2)|} \tag{B.15}$$

[1]De amplitude van de hogere formanten is immers klein, dat wil zeggen dat fouten die in dit gebied gemaakt worden ook klein zijn. Dit betreft zowel fouten bij de bepaling van de ligging van de pieken als bij de bepaling van de sterkte van de piek.

Wanneer frequentie f varieert van 0 Hz tot $1/2T$, de Nyquist-frequentie, dan loopt z ($= e^{2\pi fT}$) een halve bovencircel in het complexe vlak tegen de wijzers van de klok in, startend bij het punt (1,0). De amplituderespons bij elke frequentie f in het interval [0, Nyquist-frequentie] kun je dan vinden door de inverse van het product van twee afstanden uit te rekenen. Deze afstanden zijn die van het punt z tot elk van de twee nulpunten z_1 en z_2.

Het is te bewijzen dat om een *stabiel* filter te krijgen de polen in de eenheidscircel moeten liggen[2].

In figuur B.1 hebben we in het linker plaatje de positie van de polen getekent voor een filter van de vorm B.11 met $p = 0.9$ en $q = 0.95$. Invullen van p en q in formule B.13 geeft de nulpunten $z_{1,2} \approx -0.45 \pm 0.865i$. Het volgende script fragment draagt bij aan dit resultaat:

```
1  p = 0.9
2  q = 0.95
3  Create Polynomial... f -1 1 'q' 'p' 1
4  To Roots
5  To Spectrum... 5000 1025
6  Copy... fi
7  Formula... (if row=1 then 1 else -1 fi) * 10^-3
8          ... * self / (Spectrum_f[1,col]^2 + Spectrum_f[2,col]^2)
9  # tekenen
10 fontSize = 10
11 Font size... fontSize
12 # vx is de breedte en vy is de hoogte van het viewport.
13 vx = ...
14 vy = vx - 2.8 * fontSize / 72
15 Viewport... 0 vx 0 vy
16 ...
```

In regel 3 wordt het polynoom in de teller van formule B.12 gecreëerd. De nulpunten worden berekend in regel 4. In regel 5 wordt het spectrum berekend. Er worden 1025 frequentiepunten genomen met een Nyquist-frequentie van 5000 Hz. In regel 7 en 8 berekenen we tenslotte het inverse spectrum. Bij het tekenen met *praat* worden verschillende horizontale en verticale marges gehanteerd. Deze marges hangen af van de grootte van het lettertype (de *fontSize*). Om er bij het tekenen voor te zorgen dat een vierkant een vierkant wordt en geen rechthoek, dat een circel een circel blijft en geen ellips wordt, moeten we corrigeren voor deze verschillende marges. Dit gebeurt in de regels 10 t/m 14 van het script.

[2]Een systeem is stabiel als een eindige invoer onder alle omstandigheden ook altijd een eindige uitvoer genereert. Dit laat zich dan vertalen in de positie van de nulpunten die binnen de eenheidscircel moeten liggen.

De afstanden $|z - z_1|$ en $|z - z_2|$ zijn voor een "willekeurige" frequentie f getekend. In de rechter figuur is het amplitude-spectrum getekend dat verkregen wordt als z linksom over de bovenrand van de circel loopt. De frequentie loopt dan van 0 Hz tot de Nyquist-frequentie. We zien in het spectrum een duidelijke piek. Uit de figuur kunnen we afleiden dat als z over de bovenrand van de circel loopt dit voor de afstand tot het tweede nulpunt $|z - z_2|$ geen grote gevolgen heeft: de verhouding tussen de maximale afstand (ongeveer de positie van z in de figuur) en de minimale afstand (z op positie (-1,0)) is minder dan een factor 3 (kleiner dan 9 dB). Als z daarentegen steeds dichter in de buurt komt van het eerste nulpunt z_1, dan wordt de afstand $|z - z_1|$ erg veel kleiner. De verhouding tussen grootste afstand ($z = (1, 0)$) en kleinste afstand (z op het snijpunt van de circel met de lijn door z_1 vanuit de oorsprong) wordt dan:

$$\frac{\sqrt{(1-(-0.45))^2 + (0-0.865)^2}}{1 - \sqrt{(-0.45)^2 + 0.865^2}} = 67.7 (\approx 37 dB)$$

We kunnen hieruit afleiden dat hoe dichter een pool bij de eenheidscircel ligt, hoe geprononceerder zijn piek in het spectrum zal zijn. In het algemeen geldt dat de scherpte en bandbreedte van een piek een inverse relatie hebben. Hoe scherper een piek hoe kleiner zijn bandbreedte zal zijn.

We kunnen met deze globale methode om het spectrum te bepalen ook gelijk zien dat als de nulpunten reëel zijn dit geen "interessante" mogelijkheden oplevert. Nulpunten die in de buurt liggen van de ± 1 geven het spectrum alleen maar een helling doordat het maximum van de piek bij de 0 Hz of bij de Nyquist-frequentie ligt.

De polen zijn complex als in vergelijking B.13 de term onder het wortelteken negatief wordt:

$$\frac{p^2}{4} - q < 0, \tag{B.16}$$

zodat de "interessante" nulpunten dan gegeven worden door:

$$z_{1,2} = -\frac{p}{2} \pm i\sqrt{q - \frac{p^2}{4}} \tag{B.17}$$

De twee nulpunten zijn elkaars complex toegevoegde en liggen aan weerszijden van de reële as. Het filter is stabiel als z_1 en z_2 in de eenheidscircel liggen. Dit kunnen we vertalen in de eis $|z_1| < 1$[3]. Dit kunnen we verder uitwerken:

$$|z_1| = \sqrt{\left(-\frac{p}{2}\right)^2 + (q - \frac{p^2}{4})} = \sqrt{q} < 1 \tag{B.18}$$

[3]Als $z_2 = z_1^*$ dan geldt ook automatisch $|z_2| < 1$.

B.5.1 De frequentie bij de pool

De frequentie die bij z_1 hoort kunnen we uitrekenen door eerst z_1 in cartesische vorm $re^{i\phi}$ te brengen en daarna de frequentie te bepalen als

$$F = \frac{\phi}{\pi}\frac{1}{2T}. \tag{B.19}$$

Met behulp van formule F.18 in paragraaf F.4 rekenen we eerst de lengte r uit van z_1:

$$r = |z_1| = \sqrt{\left(\frac{-p}{2}\right)^2 + (q - \frac{p^2}{4})} = \sqrt{q} \tag{B.20}$$

De afstand van z_1 tot de oorsprong blijkt alleen afhankelijk te zijn van de coëfficiënt q en niet van p. Omdat de afstand tot de oorsprong gerelateerd is aan de scherpte van de piek betekent dit dat ook de bandbreedte alleen maar een functie van q kan zijn. Om de hoek ϕ te vinden kunnen we natuurlijk altijd formule F.19 toepassen, maar dit geeft een argument van de arctg met een wortel er in. In dit geval is het eenvoudiger om gebruik te maken van het feit dat het reële deel van z_1 $(-p/2)$ gelijk is aan $r\cos\phi$, zodat dan:

$$\cos\phi = -\frac{p}{2\sqrt{q}}, \qquad \text{zodat} \qquad \phi = \arccos -\frac{p}{2\sqrt{q}}$$

De frequentie F die hoort bij het nulpunt z_1 is dan:

$$F = \frac{1}{2\pi T}\arccos -\frac{p}{2\sqrt{q}} \tag{B.21}$$

Deze frequentie noemen we de *formantfrequentie*. De *bandbreedte* B die bij deze frequentie hoort is:

$$B = -\frac{1}{\pi T}\ln\sqrt{q} \tag{B.22}$$

Omgekeerd als we F en B kennen, dan kunnen we p en q hieruit bereken volgens:

$$q = e^{-2\pi BT} \tag{B.23}$$

$$p = -2\sqrt{q}\cos(2\pi FT) \tag{B.24}$$

B.5.2 De frequentie met maximale amplituderespons

We kunnen de frequentie met maximale amplituderespons ook uitrekenen uit de filterrespons. We beginnen met vergelijking B.15:

$$\begin{aligned}|H(z)| &= \frac{1}{|(z-z_1)(z-z_2)|} \\ &= \frac{1}{\sqrt{(z-z_1)(z-z_2)(z^*-z_1^*)(z^*-z_2^*)}} \\ &= \frac{1}{\sqrt{(z-z_1)(z^*-z_1^*)(z-z_2)(z^*-z_2^*)}}\end{aligned}$$

We vermenigvuldigen de eerste twee termen met elkaar en de laatste twee termen en krijgen dan:

$$= \frac{1}{\sqrt{(zz^*-zz_1^*-z_1z^*+z_1z_1^*)(zz^*-zz_2^*-z_2z^*+z_2z_1^*)}}$$

Met $z = e^{2\pi ifT}$ en $z_1 = re^{i\phi}$ en $z_2 = re^{-i\phi}$ worden $zz^* = 1$ en $z_1z_1^* = z_2z_2^* = r^2$ en dit resulteert dan in:

$$|H(f)| = \frac{1}{\sqrt{(1+r^2-2r\cos(2\pi fT-\phi))(1+r^2-2r\cos(2\pi fT+\phi))}} \tag{B.25}$$

De extreme waardes van $|H(f)|$ vinden we door de afgeleide te nemen naar f en deze gelijk aan nul te stellen. Deze afgeleide resulteert in een quotiënt waarbij de term in de teller de afgeleide is van de term onder het wortelteken in B.25.

$$\begin{aligned}2r2\pi T\sin(2\pi fT-\phi)(1+r^2-2r\cos(2\pi fT+\phi)) \\ + 2r2\pi T\sin(2\pi fT+\phi)(1+r^2-2r\cos(2\pi fT-\phi)) = 0\end{aligned}$$

Dit kunnen we vereenvoudigen tot:

$$\begin{aligned}\sin(2\pi fT-\phi)(\frac{1+r^2}{2r}-\cos(2\pi fT+\phi)) \\ + \sin(2\pi fT+\phi)(\frac{1+r^2}{2r}-\cos(2\pi fT-\phi)) = 0\end{aligned}$$

We splitsen:

$$\begin{aligned}&\frac{1+r^2}{2r}(\sin(2\pi fT-\phi)+\sin(2\pi fT+\phi)) \\ &-(\sin(2\pi fT-\phi)\cos(2\pi fT+\phi)+\sin(2\pi fT+\phi)\cos(2\pi fT-\phi)) = 0\end{aligned}$$

We gaan nu eerst de laatste term vereenvoudigen met behulp van formules F.15 en F.12:

$$\frac{1+r^2}{2r}(\sin(2\pi fT-\phi)+\sin(2\pi fT+\phi)) \\ -(\frac{1}{2}\{\sin 2(2\pi fT)-\sin 2\phi\}+\frac{1}{2}\{\sin 2(2\pi fT)+\sin 2\phi\})=0$$

$$\frac{1+r^2}{2r}(\sin(2\pi fT-\phi)+\sin(2\pi fT+\phi))-\sin 2(2\pi fT) = 0$$

Het eerste deel tussen haken kunnen we vereenvoudigen met formules F.9 en F.7:

$$\frac{1+r^2}{r}\sin 2\pi fT\cos\phi-\sin 2(2\pi fT)=0$$

De sinus van de dubbele hoek gaan we anders schrijven (F.10):

$$\frac{1+r^2}{r}\sin 2\pi fT\cos\phi-2\sin 2\pi fT\cos 2\pi fT \\ =\sin 2\pi fT\left\{\frac{1+r^2}{r}\cos\phi-2\cos 2\pi fT\right\}=0 \tag{B.26}$$

Het "interessante" nulpunt van deze vergelijking ligt bij een frequentie f waarvoor:

$$\cos 2\pi fT=\frac{1+r^2}{2r}\cos\phi \tag{B.27}$$

Dit geeft als oplossing voor f:

$$f=\frac{1}{2\pi T}\arccos\left(\frac{1+r^2}{2r}\cos\phi\right)$$

Als $r\approx 1$ dan is $(1+r^2)/2r\approx 1$ en krijgen we hetzelfde resultaat als bij formule B.21. Dit resultaat kunnen we interpreteren als: hoe dichter een pool bij de eenheidscircel ligt hoe meer de positie van het lokale maximum alleen bepaald wordt door de frequentie van de pool. Als de pool verder weg ligt van de eenheidscircel dan verschuift de positie van het maximum.

B.5.3 Het maximum van de amplituderespons

We herhalen hier formule B.25:

$$|H(f)|=\frac{1}{\sqrt{(1+r^2-2r\cos(2\pi fT-\phi))(1+r^2-2r\cos(2\pi fT+\phi))}} \\ =\frac{1}{\sqrt{(1+r^2)^2-2r(1+r^2)A(f)+4r^2B(f)}} \tag{B.28}$$

waarbij:

$$A(f) = \cos(2\pi fT - \phi) + \cos(2\pi fT + \phi)$$
$$B(f) = \cos(2\pi fT - \phi)\cos(2\pi fT + \phi)$$

We maken gebruik van de trigonometrische formules F.8 en F.6 om $A(f)$ te vereenvoudigen:

$$\begin{aligned} A(f) &= \cos(2\pi fT)\cos\phi + \sin(2\pi fT)\sin\phi + \cos(2\pi fT)\cos\phi - \sin(2\pi fT)\sin\phi \\ &= 2\cos(2\pi fT)\cos\phi \end{aligned}$$

De reductie van $B(f)$ gaat via formule F.13:

$$B(f) = \frac{1}{2}\left\{\cos(2(2\pi fT)) + \cos 2\phi\right\}$$

We willen $B(f)$ in termen van enkele hoeken en gebruiken daarom formule F.11:

$$\begin{aligned} &= \frac{1}{2}\left\{2\cos^2(2\pi fT) - 1 + 2\cos^2\phi - 1\right\} \\ &= \cos^2(2\pi fT) + \cos^2\phi - 1 \end{aligned}$$

We substitueren nu de gereduceerde $A(f)$ en $B(f)$ in B.28

$$|H(f)| = \frac{1}{\sqrt{(1+r^2)^2 - 2(1+r^2)2r\cos(2\pi fT)\cos\phi + (2r\cos(2\pi fT))^2 + 4r^2\cos^2\phi - 4r^2}} \tag{B.29}$$

Bij het maximum geldt volgens B.27 dat $2r\cos 2\pi fT = (1+r^2)\cos\phi$:

$$|H(f_{max})| = \frac{1}{\sqrt{(1+r^2)^2 - 2(1+r^2)^2\cos^2\phi + (1+r^2)^2\cos^2\phi + 4r^2\cos^2\phi - 4r^2}} \tag{B.30}$$

Dit kan gereduceerd worden tot:

$$= \frac{1}{(1-r^2)\sin\phi} \tag{B.31}$$

B.5.4 De bandbreedte van het formantfilter

De bandbreedte is gedefiniëerd als het verschil tussen de frequenties waarbij de amplitude $1/\sqrt{2}$ van het maximum is (of het vermogen de helft). We kunnen de frequenties f die bij deze amplitude horen bepalen door uit te rekenen:

$$|H(f)| = \frac{1}{\sqrt{2}}\,|H(f_{max})|$$

We gebruiken nu B.29 en B.31:

$$\frac{1}{\sqrt{(1+r^2)^2 - 2(1+r^2)2r\cos(2\pi fT)\cos\phi + (2r\cos(2\pi fT))^2 + 4r^2\cos^2\phi - 4r^2}} = \frac{1}{\sqrt{2}(1-r^2)\sin\phi}$$

Kwadrateren geeft:

$$\frac{1}{(1+r^2)^2 - 2(1+r^2)\cos\phi(2r\cos(2\pi fT)) + (2r\cos(2\pi fT))^2 + 4r^2\cos^2\phi - 4r^2} = \frac{1}{2(1-r^2)^2\sin^2\phi}$$

Dit geeft:

$$(2r\cos(2\pi fT))^2 - 2(1+r^2)\cos\phi(2r\cos(2\pi fT)) + (1+r^2)^2 + 4r^2\cos^2\phi - 4r^2 - 2(1-r^2)^2\sin^2\phi = 0$$

Dit kan nog gereduceerd worden tot:

$$(2r\cos(2\pi fT))^2 - 2(1+r^2)\cos\phi(2r\cos(2\pi fT)) + 2(1+r^4)\cos^2\phi - (1-r^2)^2 = 0 \quad \text{(B.32)}$$

Dit is een kwadratische vergelijking in $2r\cos(2\pi fT)$ en kan dus simpel opgelost worden met de abc-formule. We rekenen eerst de discriminant uit:

$$\begin{aligned} b^2 - 4ac &= 4(1+r^2)^2\cos^2\phi - 4\cdot 1\cdot(2(1+r^4)\cos^2\phi - (1-r^2)^2) \\ &= 4\left\{(1+r^2)^2\cos^2\phi - 2(1+r^4)\cos^2\phi - (1-r^2)^2\right\} \\ &= 4\left\{(1-r^2)^2(1-\cos^2\phi)\right\} \\ &= 4(1-r^2)^2\sin^2\phi \end{aligned} \quad \text{(B.33)}$$

De oplossing is dan:

$$2r\cos(2\pi fT) = (1+r^2)\cos\phi \pm (1-r^2)\sin\phi \quad \text{(B.34)}$$

B.6 De algemene filterrespons

Om een indruk te krijgen van hoe het spectrum van de algemene filterrespons van formule B.4 er uitziet maken we gebruik van de stelling dat een polynoom van orde n ook n nulpunten heeft. Een uitbreiding van deze stelling laat zien dat als de coëfficiënten reële getallen zijn, de nulpunten reëel zijn of in paren komen die elkaars complex toegevoegde zijn[4]. Dit betekent dat we elk polynoom kunnen schrijven als een product van "nulpunten" en B.4 kunnen schrijven als:

$$H(z) = sz^{M-N}\frac{\prod_{k=0}^{M}(z - z_k)}{\prod_{j=1}^{N}(z - z_j)}, \tag{B.35}$$

waarbij z_k en z_j nulpunten zijn en s een schaalfactor. Als geen enkel nulpunt of pool reëel is dan is nog een "vereenvoudiging" mogelijk:

$$H(z) = sz^{M-N}\frac{\prod_{k=0}^{M/2}(z - z_k)(z - z_k^*)}{\prod_{j=1}^{N/2}(z - z_j)(z - z_j^*)}, \tag{B.36}$$

Van een zich goed gedragend digitaal filter geldt voor de nulpunten van de teller en de nulpunten van de noemer dat $|z_i| \leq 1$: de nulpunten liggen in het (complexe) vlak binnen een circel met straal 1, de eenheidscircel. Nulpunten in de noemer worden ook wel *polen* genoemd. Het spectrum kunnen we berekenen door z over de bovenrand van de circel te laten lopen en B.35 uit te rekenen. Voor elke waarde van z staan dan in de teller en in de noemer allemaal producten van termen die de afstand van het punt op de rand, z, tot deze nulpunten zijn[5].

Als z in de buurt komt van een nulpunt z_i dan wordt de term $(z - z_i)$ klein en dientengevolge $H(z)$ ook. Als daarentegen de z_i een *pool* is dan zal $H(z)$ juist groot zijn. In het algemeen zal $H(z)$ de vorm hebben van een berg-en-dal landschap: de bergen treden op als z in de buurt komt van een nulpunt in de noemer. De dalen treden op als z niet in de buurt van een pool ligt of als z in de buurt van een nulpunt in de teller ligt.

B.7 Opgaves

1. Formantsynthese met één formant.

Maak een Sound met één formant met formantfrequentie $F = 500$ Hz en bandbreedte $B = 50$ Hz via de recursieve relatie B.11 met een bemonsteringsfrequentie van 16000 Hz.

[4]Neem als voorbeelden $x^2 - 3x - 2 = (x - 2)(x - 1)$ en $x^2 - 2x + 3 = (x - i\sqrt{2})(x + i\sqrt{2})$

[5]In paragraaf B.5 hebben we dit ook gebruikt voor het bepalen van de respons van het formantfilter.

Neem voor x_n een puls `if col=1 then 1 else 0 endif` en bereken de uitvoer van dit filter.

Schaal de amplitude van het signaal y_n zodat het te beluisteren is en maak dan het spectrum.

Probeer nu op twee manieren de frequentie en bandbreedte te meten van het gegenereerde signaal y_n.

1. Uit het amplitude-spectrum.
2. Uit het signaal y_n zelf. Bedenk dat dit een gedempte sinus is van de vorm $y(nT) = e^{-\pi BkT} * \sin(2\pi FkT + \phi)$. Dit betekent dat we uit de relatie tussen de amplitudes op twee tijdstippen t_1 en t_2 van dit signaal F en B kunnen bepalen.

We gaan nu onderzoeken hoe stabiel de verschillende representaties van een formant zijn. Gebruik de waardes van p en q zoals boven gegeven voor het genereren van een formant met $F = 500$ Hz en $B = 50$ Hz met een kleine verandering: maak p 1% groter. Wat is dan de formantfrequentie en bandbreedte van het nieuwe signaal? Hoeveel % bedraagt de afwijking? Welke van de twee representaties is het stabielst?

2. De invloed van bandbreedte.

Maak een script voor het genereren van één formant met een frequentie van 800 Hz waarvan de bandbreedte variabel is (bemonsteringsfrequentie is 16000 Hz).

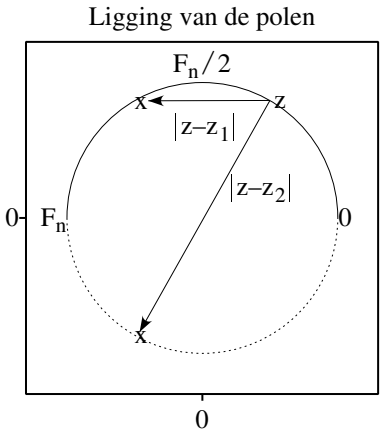
Onderzoek de invloed van de bandbreedte door in één plaatje de spectra te tekenen van de verschillende bandbreedtes. Neem als bandbreedtes bijvoorbeeld 5%, 10%, 20%, 50%, 100% en 200% van de formantfrequentie.

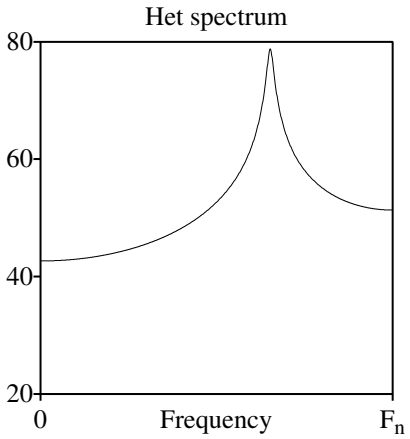
3. Formantsynthese met twee formanten.

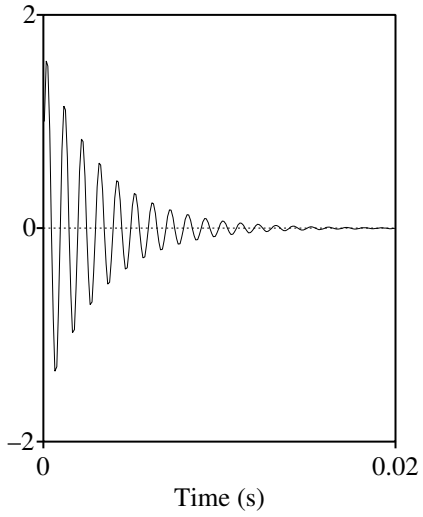
De bemonsteringsfrequentie is 16000 Hz. Maak Sound s_1 met een formantfrequentie van 800 Hz en een bandbreedte van 80 Hz.

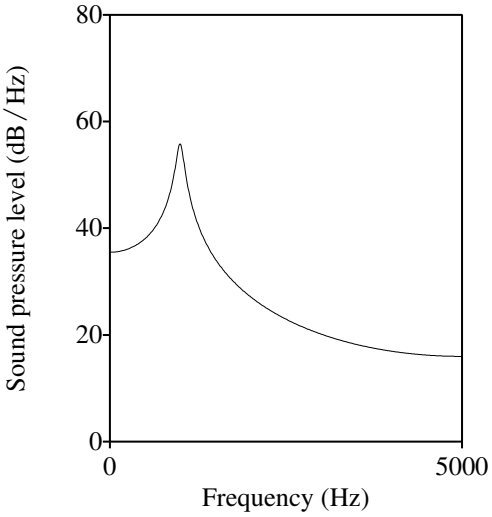
Maak een tweede Sound s_2 met $F = 1300$ Hz en $B = 130$ Hz. We kunnen nu op drie manieren een geluidje met twee formanten maken:

1. Een puls dient als invoer voor filter F_1 en de uitvoer hiervan (s_1) dient als invoer voor filter F_2.
2. Een puls dient als invoer voor filter F_2 en de uitvoer hiervan (s_2) dient als invoer voor filter F_1.
3. De uitvoer is de som van F_1 en F_2.


Ligging van de polen
$F_n/2$
x
z
$|z-z_1|$
$|z-z_2|$
0
F_n
0
x
0


Het spectrum
80
60
40
20
0
Frequency
F_n


2
0
–2
0
0.02
Time (s)


80
60
40
20
0
Sound pressure level (dB / Hz)
0
5000
Frequency (Hz)

Bijlage C

Lineaire predictie

C.1 Inleiding

Linear Predictive Coding (LPC) is de meest gebruikte analyse techniek binnen de wereld van de spraakanalyse. Deze techniek wordt in de spraakwereld al gebruikt sinds het einde van de jaren zestig (van de vorige eeuw). De basisprincipes dateren van nog weer eerder. LPC is een analyse van het tijdsignaal en is al zodanig in veel gebieden toepasbaar. Eén van de basisboeken over LPC is van Markel & Gray (1976). Een goede Nederlandstalige inleiding kan gevonden worden in Vogten (1987).

C.2 Model

Gegeven een bemonsterd signaal met monsterwaardes s_n, dan is elk monster te schrijven als een lineaire combinatie van p voorgaande monsterwaardes en een invoer:

$$s_n = -\sum_{k=1}^{p} a_k s_{n-k} + u_n \tag{C.1}$$

Dit is een digitale filter vergelijking: een invoer u_n wordt gefilterd door een recursief filter met coëfficiënten a_k en geeft dan een uitvoer s_n. Wij willen graag de coëfficiënten a_k van het filter bepalen. In de werkelijkheid, bij de analyse van het spraaksignaal bijvoorbeeld, hebben we te maken met een situatie waarbij we behalve de filtercoëfficiënten a_k ook de invoer, u_n, niet kennen. De enige schatting voor s_n die we dan kunnen maken is dat we ons alleen baseren op de dingen die we kennen, de monsterwaardes en voorlopig de u_n maar moeten vergeten.

Daarom definiëren we $\hat{s}_n$, de schatting voor s_n als volgt:

$$\hat{s}_n = -\sum_{k=1}^{p} a_k s_{n-k} \tag{C.2}$$

Het verschil tussen de geschatte waarde, $\hat{s}_n$ en de werkelijke waarde, s_n, noemen we de fout, e_n:

$$e_n = s_n - \hat{s}_n \tag{C.3}$$

Wanneer we nu formule C.2 invullen in bovenstaande formule C.3 dan krijgen we:

$$e_n = s_n + \sum_{k=1}^{p} a_k s_{n-k} \tag{C.4}$$

Een herschrijving levert dan op:

$$s_n = -\sum_{k=1}^{p} a_k s_{n-k} + e_n \tag{C.5}$$

Een vergelijking van C.1 en C.5 levert dus een schatting op van het bronsignaal u_n. De beste keuze voor onze filtercoëfficiënten a_k zijn die waardes die ervoor zorgen dat de geschatte waardes voor $\hat{s}_n$ zo veel mogelijk lijken op de originele s_n. Als ons analyseframe uit N monsterwaardes bestaat, dan proberen we het verschil tussen $\hat{s}_n$ en s_n zo klein mogelijk te maken voor alle N monsterwaardes. We definiëren daarom de kwadratische fout E als:

$$E = \sum_{n=1}^{N} e_n^2 \tag{C.6}$$

De E in deze formule is een functie van p variabelen, de a_k's. We zoeken waardes voor de a_k's zodat E zo klein mogelijk wordt. De standaardtruc voor dit soort gevallen is differentiëren naar de a_k's en de afgeleides gelijk nul stellen. Dit geeft p vergelijkingen in de a_k's. Deze vergelijkingen zijn oplosbaar zoals onderstaande afleiding zal laten zien.

$$\frac{\partial E}{\partial a_i} = \frac{\partial}{\partial a_i} \sum_{n=1}^{N} e_n^2, \qquad 1 \leq i \leq p \tag{C.7}$$

Dit zijn p vergelijkingen, voor elke a_i één. We gaan ze uitwerken:

$$\frac{\partial E}{\partial a_i} = \frac{\partial}{\partial a_i} \sum_{n=1}^{N} e_n^2 = \sum_{n=1}^{N} 2 e_n \frac{\partial e_n}{\partial a_i} \tag{C.8}$$

Uit vergelijking C.4 volgt

$$\frac{\partial e_n}{\partial a_i} = \frac{\partial e_n}{\partial a_i}\left(s_n + \sum_{k=1}^{p} a_k s_{n-k}\right) = s_{n-i} \tag{C.9}$$

Invullen van C.4 en C.9 in C.8 geeft:

$$\begin{aligned}\frac{\partial E}{\partial a_i} &= \sum_{n=1}^{N} 2\left(s_n + \sum_{k=1}^{p} a_k s_{n-k}\right) s_{n-i} \\ &= 2\sum_{n=1}^{N}\left(s_n s_{n-i} + \sum_{k=1}^{p} a_k s_{n-k} s_{n-i}\right)\end{aligned} \tag{C.10}$$

We stellen nu de partiële afgeleide gelijk nul en kunnen dan de laatste term herschrijven (na door 2 gedeeld te hebben):

$$\sum_{k=1}^{p} a_k \sum_{n=1}^{N} s_{n-k} s_{n-i} = -\sum_{n=1}^{N} s_n s_{n-i} \tag{C.11}$$

In de termen links en rechts van het isgelijk-teken staan indexeringen van monsterwaardes buiten het analysevenster met indices van 1 tot N. We moeten kiezen hoe we de adressering buiten het analysevenster aanpakken. Wij maken de aanname dat de monsterwaardes buiten het analysevenser gelijk nul zijn. De vergelijkingen die we overhouden zijn dan gebaseerd op emphautocorrelaties, deze methode heet dan ook de autocorrelatiemethode. De s_{n-k} is, ten opzichte van de index n, verschoven over k monsters en de s_{n-i} over i monsters. Netto is dit een verschuiving over $|i-k|$ monsters en we kunnen C.11 dan ook schrijven als:

$$\sum_{k=1}^{p} a_k \sum_{n=1}^{N} s_n s_{n+i-k} = -\sum_{n=1}^{N} s_n s_{n-i} \tag{C.12}$$

De autocorrelaties R_i zijn gedefiniëerd als:

$$R_i = \sum_{n=-\infty}^{n=+\infty} s_n s_{n+|i|}, \qquad R_i = R_{-i} \tag{C.13}$$

We kunnen dan vergelijkingen C.12 in termen van autocorrelaties definiëren:

$$\sum_{k=1}^{p} a_k R_{|i-k|} = -R_i \qquad\qquad 1 \le i \le p, \tag{C.14}$$

waarbij:

$$R_i = \sum_{n=1}^{N-i} s_n s_{n+i}$$

We kunnen deze p vergelijkingen ook in matrix vorm schrijven:

$$\begin{pmatrix} R_0 & R_1 & \dots & R_{p-1} \\ R_1 & R_0 & \dots & R_{p-2} \\ \dots & \dots & \dots & \dots \\ R_{p-1} & R_{p-2} & \dots & R_0 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_p \end{pmatrix} = - \begin{pmatrix} R_1 \\ R_2 \\ \dots \\ R_p \end{pmatrix} \tag{C.15}$$

Of nog korter als:

$$\mathbf{R}\mathbf{a} = -\mathbf{r}, \tag{C.16}$$

waarbij $\mathbf{R}$ een $p \times p$ matrix is en $\mathbf{a}$ en $\mathbf{r}$ kolomvectoren zijn. De oplossing $\mathbf{a}$ van bovenstaande matrixvergelijking kan dan verkregen worden door links en rechts met de inverse van matrix $\mathbf{R}$ te vermenigvuldigen:

$$\mathbf{a} = -\mathbf{R}^{-1}\mathbf{r} \tag{C.17}$$

Zoals blijkt uit vergelijking C.16 heeft de matrix $\mathbf{R}$ een speciale symmetrie, het is een zogenaamde *Toeplitz* matrix. Er is een speciaal algoritme ontwikkeld om een vergelijking van het type C.16 met een Toeplitz matrix op te losssen, het *Levinson algoritme*. Dit algoritme maakt gebruik van de speciale symmetrie en kost slechts $O(p^2)$ bewerkingen. Het uitrekenen van de coëfficiënten via vergelijking C.17 is van orde $O(p^3)$ door de bepaling van de inverse $\mathbf{R}^{-1}$.

We kunnen vergelijking C.1 ook interpreteren als een filter met invoer u_n en uitvoer s_n, en (recursieve) filtercoëfficiënten a_k. Bij de autocorrelatiemethode is het filter "leeg" als we beginnen: de monsterwaardes s_0, s_{-1}, ..., s_{1-p} zijn namelijk allemaal nul. Het blijkt dat de autocorrelatiemethode altijd een stabiel filter oplevert.

De autocorrelatiemethode is maar één van de vele in gebruik zijnde methodes om de coëfficiënten a_k uit te rekenen. Bij de *covariantie*methode bijvoorbeeld is het filter niet "leeg" bij het begin maar bevat al p waardes. De minimalisatie vindt hier plaats over $N - p$ monsters in plaats van N. De methodes van Marple (1980) en Burg (zie Press et al., 1996) maken zelfs gebruik van voorwaartse- en achterwaardse predictie. Bij deze laatsgenoemde predictiemethode schrijft men in plaats van vergelijking C.2 voor de schatting $\hat{s}_n$ de volgende:

$$\hat{s}_n = -\sum_{k=1}^{p} a_k s_{n-k} - \sum_{m=1}^{p} b_m s_{n+m}$$

C.3 Het frequentiedomein

We zijn begonnen met vergelijking C.1:

$$s_n = -\sum_{k=1}^{p} a_k s_{n-k} + u_n$$

Deze vergelijking beschrijft een ”standaard” recursief digitaal filter: een ingangssignaal u_n wordt recursief gefilterd en levert een uitgangssignaal s_n. We kunnen hiervan de Z-transform uitrekenen en krijgen dan:

$$S(z) = H(z)U(z), \tag{C.18}$$

waarbij $S(z)$ en $U(z)$ de transform zijn van respectievelijk het uitgangs- en het ingangssignaal. $H(z)$ is het filter dat gegeven wordt door:

$$H(z) = \frac{1}{1 + \sum_{k=1}^{p} a_k z^{-k}} \tag{C.19}$$

Omdat we het bronsignaal u_n niet kenden zijn we overgegaan naar de formulering van formule C.4

$$e_n = s_n + \sum_{k=1}^{p} a_k s_{n-k}$$

De Z-transform hiervan is:

$$E(z) = A(z)S(z), \tag{C.20}$$

waarbij dan

$$A(z) = \frac{1}{H(z)} = 1 + \sum_{k=1}^{p} a_k z^{-k} \tag{C.21}$$

Dit is de zogenaamde *inverse-filter* formulering. In plaats van het bronsignaal te filteren om het uitvoersignaal te krijgen zoals bij C.18, filteren we het uitvoersignaal (invers) om het bronsignaal te krijgen. Het filter $A(z)$ wordt het *analysefilter* genoemd, $H(z)$ het *synthesefilter*.

We associëren $U(z)$ met de *bron*, de stembanden, en het filter $H(z)$ met het *spraakkanaal*, de mond-keelholte. Voor het analyseren van het spraaksignaal via een lpc-analyse maken we stilletjes de volgende aannames:

- Spraak is semi-stationair. Binnen één analyseframe is het stationair[1].
- De stembanden kunnen gemodelleerd worden door óf een pulstrein óf door witte ruis: de pulstrein voor de stemhebbende stukken, de ruis voor de stemloze segmenten. Dit betekent dat stemhebbende fricatieven problemen kunnen opleveren.
- Het gekombineerde effect van de glottis, de mon-keelholte en de uitstraling bij de mond kan gemodelleerd worden als een lineaire *all-pole* filter (AR, is Auto Regressief).
- De bron en het filter zijn onafhankelijk.

C.4 Andere representaties van het filter

De beschrijving van het spectrum via het synthesefilter C.19 of het analysefilter C.21 heeft een aantal praktische bezwaren. We hebben gezien dat bij dergelijke filters de polen/nulpunten informatie geven over de vorm van het spectrum en de plaats van de pieken. Hierbij kunnen kleine afwijkingen of verstoringen in de coefficieënten a_k grote afwijkingen op de posities (en bandbreedtes) van de pieken hebben. Dit probleem speelt vooral als we spraak via lpc willen comprimeren. Hierbij kunnen dan kwantisatie- en transmissiefouten tot een instabiel filter leiden. Daarom worden uit het p-de orde filter vaak andere representaties afgeleid die stabieler zijn en daardoor minder gevoelig voor deze fouten. Twee voorbeelden van andere representaties zijn de *cascade van formantfilters* en het *ladderfilter*.

C.4.1 Cascade van formantfilters

We kunnen de representatie C.21 van $H(z)$ ontbinden als een cascade van tweede orde recursieve filters. In plaats van $H(z)$ te ontbinden kunnen we natuurlijk ook $A(z)$ ontbinden op de volgende manier:

$$A(z) = 1 + \sum_{k=1}^{M} a_k z^{-k} = \prod_{k=1}^{M/2} (1 + p_k z^{-1} + q_k z^{-2})$$

Hierin zijn p_k en q_k de filtercoëfficiënten van het formantfilter van paragraaf B.5. Van dit filter weten we de relatie tussen de pq-parametrisatie en de FB-parametrisatie, ze wordt gegeven door de formules B.23 en B.24. Voor de FB-parametrisatie geldt dat ze een stuk robuuster is dan de pq-parametrisatie (zie ook opgave 1 op bladzijde D-FILTERS–13).

[1] Stationair wil zeggen dat de gemiddelde statistische eigenschappen niet veranderen.

C.4.2 Ladderfilter

Een ladderfilter kan opgevat worden als het electrische analogon van een verliesloze akoestische buis die uit p cilindervormige segmenten bestaat. Elk segment heeft een andere doorsnee en de buis als geheel vormt een sterk vereenvoudigde benadering van de vorm van het spraakkanaal. Op elke overgang tussen twee segmenten wordt de akoestische golf gedeeltelijk doorgelaten en gedeeltelijk gereflecteerd. De mate van reflectie, die een functie is van de doorsnedes van de twee segmenten, wordt aangegeven door de *reflectiecoëfficiënt*. In het de berekening van de lpc-coëfficiënten volgens de autocorrelatie-methode komen de reflectiecoëfficiënten al voor als intermediaire representatie. De relatie tussen de reflectiecoëfficiënt r_i van de overgang tussen het segment $i-1$ met oppervlakte A_{i-1} en het volgende segment i met oppervlakte A_i is als volgt Markel & Gray (1976):

$$r_i = \frac{A_{i-1} - A_i}{A_{i-1} + A_i} \tag{C.22}$$

De aanname die altijd gemaakt wordt is dat de telling begint bij de lippen. A_0 representeert dan de buis met een (oneindige) oppervlakte voor de lippen, A_p de oppervlakte achter de glottis. Uit vergelijking C.22 zien we dat als we alle oppervlaktes met dezelfde faktor vermeningvuldigen, de reflectiecoëfficiënten hierdoor niet veranderen. Het is duidelijk dat we dus alleen relatieve oppervlaktes kunnen berekenen. De schalingsfactor kunnen we zelf kiezen, de keuze die meestal gemaakt wordt is dat we de glottisoppervlakte gelijk 2 maken: $A_p = 1$.

Bijlage D

Bandfilteranalyse

D.1 Inleiding

Een bandfilteranalyse meet de energie in filterbanden als functie van de tijd. Het is één van de vele methodes om een spectrale representatie van het spraaksignaal te krijgen. Er zijn een aantal redenen te noemen waarom men een bandfilteranalyse gebruikt:

- De analyse toont grote verwantschap met de manier waarop ons oor een spectrale analyse uitvoert.
- Het is een *objectieve* analyse, er is niet, zoals bij een formantanalyse, voorkennis vereist voor de interpretatie.
- Zij kan automatisch uitgevoerd worden.
- Zij kan zowel in hardware als in software worden uitgevoerd.

Grofweg kan men zeggen dat de keuze van het type bandfilteranalyse dat men wil uitvoeren vanuit twee extreme richtingen bepaald wordt: vanuit de *perceptie* en vanuit de *automatische spraakherkenning*. Bij de perceptie georiënteerde bandfilteranalyse wil men zo nauwkeurig mogelijk de menselijke auditieve periferie nabootsen en onderzoeken. Bij de spraakherkenning wil men een zo efficiënt mogelijke representatie van het signaal voor optimale herkenning en is perceptieve relevantie geen criterium.

Vanuit welk perspectief men de bandfilteranalyse ook gebruikt, er zullen altijd de volgende keuzes gemaakt moeten worden over:

- De filterfunctie in termen van bijvoorbeeld bandbreedte, helling van de functie bij de lage en de hoge frequenties, modelering van neurofysiologische of psychofysische afstemcurves.

- Wat zijn de centrale frequenties van de filters?
- Wat is de versterkingsfacor van elk filter?
- Welk frequentiegebied wil men overdekken.

Bij een bandfilteranalyse maakt men gebruik van een filterbank, dit is een verzameling filters waarbij elk filter is afgestemd op een andere frequentieband. Meestal hebben de centrale frequenties een vaste afstand tot elkaar op de één of andere frequentieschaal (mel, bark, log).

D.2 Problemen met bandfilteranalyse

Invloed van de grondtoon.

D.3 Mel Frequency Cepstral Coefficients analyse

D.4 Opgaves

1. Analyse
signaal 500 Hz tot 0.213, 3500 Hz vanaf 0.867 en lineair stijgen er tussen in. Daarna bandfilteranalyse Ff (B=100 Hz), Mel en Barkf.

Bijlage E

Instantane frequentie

De instantane frequentie $f(t)$ van een signaal $s(t) = \sin \phi(t)$ is gedefiniëerd als:

$$f(t) = \frac{1}{2\pi} \frac{d\phi(t)}{dt}$$

Bijlage F

Rekentrucs

F.1 Euler

De belangrijkste relaties tussen complexe getallen en sinussen en cosinussen is de volgende:

$$e^{i\phi} = \cos\phi + i\sin\phi \qquad ''Euler'' \tag{F.1}$$

Uit bovenstaande relatie kunnen we die afleiden met negatieve hoek:

$$e^{-i\phi} = \cos\phi - i\sin\phi \tag{F.2}$$

We willen nu de $\cos\phi$ en de $\sin\phi$ uitdrukken in de exponenten. We beginnen met bovenstaande vergelijkingen F.1 en F.2 op te tellen:

$$e^{i\phi} + e^{-i\phi} = 2\cos\phi$$

Even herschrijven:

$$\cos\phi = \frac{e^{i\phi} + e^{-i\phi}}{2} \tag{F.3}$$

Nu hetzelfde verhaal voor de $\sin\phi$ door vergelijking F.2 van F.1 af te trekken:

$$e^{i\phi} - e^{-i\phi} = 2i\sin\phi$$

We krijgen nu eenzelfde soort uitdrukking als F.3.

$$\sin\phi = \frac{e^{i\phi} - e^{-i\phi}}{2i} \tag{F.4}$$

F.2 Euler en samengestelde argumenten

We gaan hier trigonometrische functies waarvan de argumenten samengesteld zijn (zoals $sin(\alpha - \beta)$) analyseren. We beginnen met de formule van Euler van een samengestelde vorm en proberen hieruit alles af te leiden:

$$e^{i(\alpha+\beta} = \cos(\alpha + \beta) + i \sin(\alpha + \beta) \tag{F.5}$$

We gaan het deel voor het isgelijk-teken ontbinden:

$$\begin{aligned} &= e^{i\alpha} e^{i\beta} = (\cos\alpha + i\sin\alpha)(\cos\beta + i\sin\beta) \\ &= \cos\alpha\cos\beta - \sin\alpha\sin\beta + i(\cos\alpha\sin\beta + \sin\alpha\cos\beta) \end{aligned}$$

De reële delen van de twee bovenstaande vergelijkingen moeten aan elkaar gelijk zijn:

$$\cos(\alpha + \beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta \tag{F.6}$$

Hetzelfde geldt voor de imaginaire delen:

$$\sin(\alpha + \beta) = \cos\alpha\sin\beta + \sin\alpha\cos\beta \tag{F.7}$$

Door substitutie $-\beta$ krijgen we:

$$\cos(\alpha - \beta) = \cos\alpha\cos\beta + \sin\alpha\sin\beta \tag{F.8}$$

$$\sin(\alpha - \beta) = -\cos\alpha\sin\beta + \sin\alpha\cos\beta \tag{F.9}$$

Speciale gevallen zijn die voor de dubbele hoek, waarbij $\alpha = \beta$:

$$\sin 2\alpha = 2\sin\alpha\cos\alpha \tag{F.10}$$

$$\begin{aligned} \cos 2\alpha &= \cos^2\alpha - \sin^2\alpha \qquad \text{óf,} \\ &= 1 - 2\sin^2\alpha \qquad \text{óf,} \\ &= 2\cos^2\alpha - 1 \end{aligned} \tag{F.11}$$

F.3 Euler en de som van twee sinussen

We gebruiken relatie F.4 om $\sin\alpha + \sin\beta$ te herschrijven:

$$\sin\alpha + \sin\beta = \frac{e^{i\alpha} - e^{-i\alpha} + e^{i\beta} - e^{-i\beta}}{2i}.$$

Nu eerst hergroeperen:

$$= \frac{1}{2i}\left\{(e^{i\alpha} + e^{i\beta}) - (e^{-i\alpha} + e^{-i\beta})\right\}$$

De som van de halve hoeken afsplitsen geeft:

$$= \frac{1}{2i}\left\{e^{i\frac{\alpha+\beta}{2}}(e^{i\frac{\alpha-\beta}{2}} + e^{-i\frac{\alpha-\beta}{2}}) - e^{-i\frac{\alpha+\beta}{2}}(e^{-i\frac{\alpha-\beta}{2}} + e^{i\frac{\alpha-\beta}{2}})\right\}.$$

De termen tussen haakjes kunnen we afsplitsen:

$$= \frac{1}{2i}(e^{i\frac{\alpha+\beta}{2}} - e^{-i\frac{\alpha+\beta}{2}})(e^{i\frac{\alpha-\beta}{2}} + e^{-i\frac{\alpha-\beta}{2}})$$

Nu gebruiken we F.4 en F.3:

$$= \frac{1}{2i}(2i\sin\frac{\alpha+\beta}{2})(2\cos\frac{\alpha-\beta}{2})/2i$$

Ons resultaat:

$$\sin\alpha + \sin\beta = 2\sin\frac{\alpha+\beta}{2}\cos\frac{\alpha-\beta}{2} \tag{F.12}$$

De som en verschillen van twee cosinussen kunnen we dan afleiden uit F.12 door de volgende translatiesymmetrieën tussen sinus en cosinus te gebruiken:

$$\begin{aligned}\cos\gamma &= \sin(\gamma + \frac{\pi}{2})\\ \sin\delta &= \cos(\delta - \frac{\pi}{2})\\ -\cos\epsilon &= \cos(\epsilon + \pi)\end{aligned}$$

We krijgen dan bijvoorbeeld:

$$\begin{aligned}\cos\alpha + \cos\beta &= \sin(\alpha + \frac{\pi}{2}) + sin(\beta + \frac{\pi}{2})\\ &= 2\sin(\frac{\alpha+\beta}{2} + \frac{\pi}{2})\cos\frac{\alpha-\beta}{2}\\ &= 2\cos\frac{\alpha+\beta}{2}\cos\frac{\alpha-\beta}{2}\end{aligned} \tag{F.13}$$

En nog deze:

$$\cos\alpha - \cos\beta = -2\sin\frac{\alpha+\beta}{2}\sin\frac{\alpha-\beta}{2} \tag{F.14}$$

$$\sin\alpha - \sin\beta = 2\sin\frac{\alpha-\beta}{2}\cos\frac{\alpha+\beta}{2} \tag{F.15}$$

F.4 Van Euclidisch naar Cartesisch en terug

Het komt vaak voor dat we een complex getal moeten omzetten naar een andere representatie. Stel we hebben een punt in het vlak met coördinaten (x, y) zoals in figuur F.1. Dit is de *Euclidische* representatie: de coördinaten zijn de projecties op twee onderling loodrechte richtingen. Stel we willen een andere tweedimensionale representatie: de afstand r tot de oorsprong en de hoek ϕ die gemaakt wordt met de x-as. Dit is de *Cartesische* representatie.

Van Cartesisch *naar* Euclidisch betekent van r en ϕ naar x en y.
Uit de figuur zien we dat:

$$x = r\cos\phi \tag{F.16}$$
$$y = r\sin\phi \tag{F.17}$$

Van Euclidisch *naar* Cartesisch betekent van x en y naar r en ϕ. Kwadrateren van vergelijkingen F.16 en F.17 geeft:

$$x^2 = r^2\cos^2\phi$$
$$y^2 = r^2\sin^2\phi$$

Optellen van de kwadraten geeft:

$$x^2 + y^2 = r^2\cos^2\phi + r^2\sin^2\phi = r^2(\cos^2\phi + \sin^2\phi) = r^2$$

Dit geeft voor r:

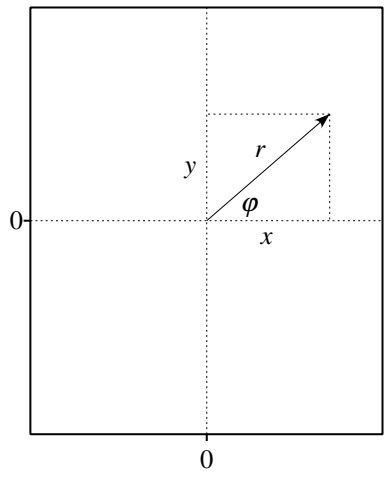
$$r = \sqrt{x^2 + y^2} \tag{F.18}$$

Om ϕ op te lossen gaan we vergelijking F.17 delen door F.16:

$$\frac{y}{x} = \frac{\sin\phi}{\cos\phi} = \tan\phi,$$

zodat ϕ wordt:

$$\phi = \tan^{-1}\frac{y}{x} = \arctan\frac{y}{x} \tag{F.19}$$


r
y
φ
0
x
0

Bijlage G

Rekentrucs-privé

$$
\begin{aligned}
G(f) &= \sum_{k=-N}^{N} e^{2\pi i f k t_0} \\
&= \sum_{k=1}^{N} e^{-2\pi i f k t_0} + 1 + \sum_{k=1}^{N} e^{2\pi i f k t_0} \\
&= \frac{e^{2\pi i f t_0} - e^{2\pi i f (N+1) t_0}}{1 - e^{2\pi i f t_0}} + 1 + \frac{e^{-2\pi i f t_0} - e^{-2\pi i f (N+1) t_0}}{1 - e^{-2\pi i f t_0}} \\
&= 1 + \frac{(1 - e^{-2\pi i f t_0})(e^{2\pi i f t_0} - e^{2\pi i f (N+1) t_0}) + (1 - e^{2\pi i f t_0})(e^{-2\pi i f t_0} - e^{-2\pi i f (N+1) t_0})}{(1 - e^{2\pi i f t_0})(1 - e^{-2\pi i f t_0})} \\
&= 1 + \frac{e^{2\pi i f t_0} + e^{-2\pi i f t_0} - e^{2\pi i f (N+1) t_0} - e^{-2\pi i f (N+1) t_0} - 2 + e^{2\pi i f N t_0} + e^{-2\pi i f N t_0}}{2 - e^{2\pi i f t_0} - e^{-2\pi i f t_0}} \\
&= 1 + \frac{2\cos(2\pi f t_0) - 2\cos(2\pi f (N+1) t_0) - 2 + 2\cos(2\pi f N t_0)}{2 - 2\cos(2\pi f t_0)} \\
&= \frac{\cos(2\pi f N t_0) - \cos(2\pi f (N+1) t_0)}{1 - \cos(2\pi f t_0)} \qquad \text{(G.1)}
\end{aligned}
$$

We hebben hierbij gebruik gemaakt van:

$$
\sum_{k=1}^{N} r^N = \frac{r - r^{N+1}}{1 - r} \qquad \text{(G.2)}
$$

De DFT van een blokgolf:

$$
H_n = \sum_{k=0}^{N-1} e^{2\pi i k n / N}
$$

De blokfunctie is 1 tot index $M-1$, dan

$$H_n = \sum_{k=0}^{M-1} e^{2\pi ikn/N}$$

Stel $r = e^{2\pi in/N}$, dan

$$\begin{aligned} H_n &= \sum_{k=0}^{M-1} r^k \\ &= \frac{1-r^M}{1-r} \end{aligned}$$

We substitueren weer $r = e^{2\pi in/N}$, dan

$$H_n = \frac{1-e^{2\pi inM/N}}{1-e^{2\pi in/N}}$$

Teller en noemer maal $1 + e^{-2\pi in/N}$

$$\begin{aligned} &= \frac{(1-e^{2\pi inM/N})(1+e^{-2\pi in/N})}{-2i\sin(2\pi n/N)} \\ &= \frac{1+e^{-2\pi in/N}-e^{2\pi inM/N}-e^{2\pi in(M-1)/N}}{-2i\sin(2\pi n/N)} \\ &= \frac{i+ie^{-2\pi in/N}-ie^{2\pi inM/N}-ie^{2\pi in(M-1)/N}}{2\sin(2\pi n/N)} \end{aligned}$$

Scheiding in reëel en imaginair deel:

$$\begin{aligned} Re[H_n] &= \frac{-\sin(2\pi n/N)+\sin(2\pi nM/N)+\sin(2\pi n(M-1)/N)}{2\sin(2\pi n/N)} \\ Im[H_n] &= \frac{1+\cos(2\pi n/N)-\cos(2\pi nM/N)-\cos(2\pi n(M-1)/N)}{2\sin(2\pi n/N)} \end{aligned} \quad \text{(G.3)}$$

Nulpunten zoeken in het amplitude spectrum: Er geldt dat: $|a| = \sqrt{aa^*}$ en dat $|\frac{a}{b}| = \sqrt{\frac{aa^*}{bb^*}}$
We maken gebruik van

$$\begin{aligned} |1-e^{i\phi}| &= \sqrt{(1-e^{i\phi})(1-e^{-i\phi})} \\ &= \sqrt{1-e^{-i\phi}-e^{i\phi}+1} \\ &= \sqrt{2-(e^{i\phi}+e^{-i\phi})} \\ &= \sqrt{2-2\cos\phi} \end{aligned}$$

De nulpunten in het amplitude-spectrum:

$$\begin{aligned} |H_n| &= \left| \frac{1 - e^{2\pi i n M/N}}{1 - e^{2\pi i n/N}} \right| \\ &= \sqrt{\frac{2 - 2\cos(2\pi n M/N)}{2 - 2\cos(2\pi n/N)}} \\ &= \sqrt{\frac{1 - \cos(2\pi n M/N)}{1 - \cos(2\pi n/N)}} \end{aligned}$$

De nulpunten liggen waar de teller nul is:

$$\cos(2\pi n M/N) = 1,$$

$$2\pi n M/N = (k - \frac{1}{2})\pi$$

De nulpunten liggen op afstanden $\frac{N}{2M}$.

Bijlage H

Verklarende woordenlijst

ADC **A**naloog **D**igitaal **C**onverter. Een apparaat dat een analoog signaal omzet in een digitaal signaal.

ADPCM **A**daptive **P**ulse **C**ode **M**odulation.

bemonsteringsfrequentie Het aantal keren per seconde dat een signaal bemonsterd is.

blokgolf Een signaal dat afwisselend twee verschillende waardes aanneemt.

DAC **D**igitaal **A**naloog **C**onverter. Een apparaat dat een digitaal signaal omzet in een analoog signaal.

DAT-recorder **D**igitale **A**udio **T**ape recorder. Maakt geluidopnames op tape met een bemonsteringsfrequentie van 48000 Hz.

DFT **D**iscrete **F**ourier **T**ransformatie

FFT **F**ast **F**ourier **T**ransform. Een algoritme om in $O(N \log_2 N)$ tijd de Fourier-transformatie van N monsterwaardes te kunnen bepalen.

JND **J**ust **N**oticible **D**ifference.

Nyquist-frequentie De bandbreedte van een bemonsterd signaal. Gelijk aan de helft van de bemonsteringsfrequentie. Wanneer, zoals meestal het geval is bij spraakopnames, het bemonsterde signaal een continu spectraal gebied bestrijkt dat bij 0 Hertz begint, dan is de Nyquist-frequentie ook de hoogste frequentie die nog gerepresenteerd kan worden in hety bemonsterde signaal.

PCM **P**ulse **C**ode **M**odulation.

H.1 wiskundige hulpmiddelen

Beginnersnivo van Raaij (1991b) samen met van Raaij (1991a). Gevorderden Ayres (1972) en Spiegel (1988).

Bibliography

Ayres, F. (1972): *Differential and Integral Calculus*, Schaum Outline series, McGraw-Hill.

Boersma, P. P. (1998): *Functional Phonology: Formalizing the interactions between articulatory and perceptual drives*, PhD dissertation, University of Amsterdam.

Boersma, P. P. G. (1993): "Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound", *Proceedings of the Institute of Phonetic Sciences University of Amsterdam* **17**: 97–110.

Deller, J. R., J. H. Hansen & J. G. Proakis (2000): *Discrete-Time processing of Speech Signals*, IEEE Press.

Furui, S. (1989): *Digital Speech Processing, Synthesis, and Recognition*, Marcel Dekker, Inc.

Hess, W. J. (1992): "Pitch and voicing determination", in S. Furui & M. M. Sondhi (eds.), *Advances in Speech Signal Processing*, chap. 2, 3–48, Marcel Dekker, Inc.

Markel, J. (1972): "Digital inverse filtering: A new tool for formant trajectory estimation", *IEEE Trans. on Audio Electroacoust.* 129–137.

Markel, J. & A. Gray, Jr. (1976): *Linear prediction of speech*, Springer Verlag, Berlin.

Marple, L. (1980): "A new autoregressive spectrum analysis program", *IEEE Trans. on ASSP* 441–451.

O'Shaughnessey, D. (1987): *Speech communication*, Addison-Wesley.

Papoulis, A. (1980): *Circuits and Sytems: A modern approach*, Series in Electrical and Computer Engineering, Holt, Rinehart and Winston, Inc.

Papoulis, A. (1988): *Signal analysis*, McGraw-Hill.

Parsons, T. W. (1987): *Voice and Speech Processing*, McGraw-Hill.

Press, W. H., S. A. Teukolsky, W. T. Vetterling & B. P. Flannery (1996): *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, 2nd edn.

Spiegel, M. R. (1988): *Advanced Calculus*, Schaum Outline series, McGraw-Hill.

van den Enden, A. & N. Verhoeckx (1987): *Digitale signaalbewerking*, Delta

Press BV.

van Raaij, F. (1991a): *Antwoorden bij Geprogrammeerde instructie moderne wiskunde*, Delta Press.

van Raaij, F. (1991b): *Geprogrammeerde instructie moderne wiskunde*, Delta Press.

Vogten, L. L. (1987): "Basis techniek in de signaalanalyse: Lpc", in L. F. ten Bosch & D. J. M. Weenink (eds.), *Reader colloquium signaalanalyse en spraak*, 3–32.

Wolfram, S. (1996): *The Mathematica Book*, Cambridge University Press, 3rd edn.