# DIGITALE SIGNAALVERWERKING

## INLEIDING

Een nieuw onderwerp in de exameneisen voor de F-vergunning is digitale signaal verwerking (DSP van Digital Signal Processing). Dit stencil beoogt een inleiding in dit onderwerp te verzorgen.

Signaalverwerking is een bezigheid die we in de radiotechniek veelvuldig beoefenen. Bij een SSB-QSO zetten we aan de zenderzijde een geluidssignaal in diverse stappen om in een hf-signaal met een flink vermogen dat door de antenne kan worden uitgestraald. Aan ontvangstzijde vindt een hele reeks bewerkingen plaats op het door de ontvangstantenne opgepikt hf-signaal met als eindresultaat een min of meer gelijkende copie van het oorspronkelijke geluidssignaal dat ten gehore wordt gebracht door een luidspreker.

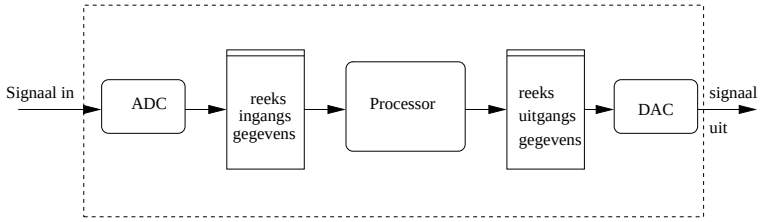
Traditioneel werden al deze stappen uitgevoerd in analoge techniek met behulp van versterker- en filtertrappen met transistoren en/of buizen, LC-kringen, etc. Tegenwoordig is het goed mogelijk een aantal van deze bewerkingen door een computer te laten uitvoeren. In plaats van een signaalbewerking uit te voeren met een schakeling van elektronische componenten, benutten we dan een **computerprogramma** dat die functie vervult.

Moderne tranceivers voeren steeds meer van hun taken uit in computerprogramma's (software). Dat biedt grote voordelen:

- de reproduceerbaarheid van de functies is perfect, die van schakelingen zijn sterk afhankelijk van de toleranties van onderdelen
- het maakt de apparatuur compact - een computerchip neemt in het algemeen minder plaats in dan spoelen, condensatoren en weerstanden
- filters verlopen niet door veroudering van componenten en er zijn geen temperatuursinvloeden
- het maakt de apparatuur flexibel - een schakeling veranderen kost meer moeite dan het aanpassen van een computerprogramma
- de mogelijkheden zijn groter en de kosten zijn lager - sommige functies, met name speciale filterbewerkingen zijn met schakelingen nauwelijks of slechts tegen hoge kosten te realiseren

Computers werken met getallen. Om dus een bepaalde bewerking op een signaal met een computer te kunnen uitvoeren, zal dat signaal eerst moeten

worden omgezet in een reeks getallen. Trouwens daar komt het woord **digitaal** vandaan (digit= vinger waarmee je kunt tellen). Dit proces van het omzetten van een signaal in getallen noemt men Analoog/Digitaal Conversie (=omzetting), afgekort **ADC**. De bewerking van het gedigitaliseerde signaal levert als resultaat een nieuwe reeks getallen, het uitgangssignaal in digitale vorm. Om dit eindresultaat ten gehore te kunnen brengen zullen die getallen weer in een elektrisch signaal moeten worden omgezet. Dit proces heet Digitaal/Analoog Conversie en gebeurt met een zogeheten Digitaal/Analoog Converter (**DAC**). Een blokschema van digitale signaalverwerking ziet er dan ook uit als getekend in figuur 1. Na de ADC en voor de DAC is een




Figuur 1: Blokschema DSP

blokje getekend dat aangeeft dat er een aantal gegevens een tijdje bewaard moet worden. Voor veel digitale signaalverwerkingsmethoden is het noodzakelijk om te kunnen beschikken over meerdere waarden van het signaal op verschillende tijdstippen. In DSP wordt dat gerealiseerd door een aantal metingen van het signaal te bewaren in computer**geheugen**. Het geheel in het gestippelde blok van figuur 1 kan ondergebracht zijn in één chip, maar kan evengoed een volledige PC met geluidskaart zijn.

In het hierna volgende zullen we een aantal van de functieblokken in een digitale signaalprocessor onder de loep nemen.

# ANALOOG/DIGITAAL CONVERSIE

De Analoog Digitaal Convertor (ADC) meet een spanning en zet die om in een getal voor de computer. Zijn belangrijkste karakteristieken zijn:

- spanningsbereik
- nauwkeurigheid

- conversietijd
- openingstijd

Het **spanningsbereik** moet passen bij de amplitude van het te meten signaal, net als bij een gewone voltmeter zijn schaal zó gekozen moet worden dat er een redelijke uitslag is die binnen de schaal blijft. Het meten van het signaal wordt wel het **bemonsteren** (Eng. sampling) van het signaal genoemd. Uiteraard moeten de hoogste en de laagste signaalwaarden binnen het bereik vallen. Toch moet het bereik ook weer niet te groot worden gekozen omdat anders de nauwkeurigheid daaronder lijdt. Ook de ingangsimpedantie moet voldoende hoog zijn om het signaal niet te beïnvloeden, maar meestal is dat voor de gangbare ADC-chips wel in orde.

De **nauwkeurigheid** wordt bepaald door het aantal **bits** van de ADC. Computers rekenen met zogeheten **binaire** getallen (in het tweetrallig stelsel). Als de ADC n bits heeft dan is het grootste getal dat hij kan produceren gelijk aan $2^n$-1. Dus een 8-bits ADC kan als grootste getal $2^8$-1 = 256-1= 255 leveren. Een 10-bits ADC $2^{10}$-1=1023, etc. De spanning die hoort bij een bepaald getal kunnen we uitrekenen met de formule:

$$U_{ADC} = \frac{ADCwaarde}{2^n - 1} * Bereik(V) \tag{1}$$

Een voorbeeld. Stel we hebben een 10-bits ADC met een bereik van 1 Volt en de ADC geeft een meting met het getal 200, dan is de spanning
(200/($2^{10}$ - 1) * 1 = (200/1023) * 1 = 0,195 V.

Een ADC geeft aan de computer als resultaat alleen gehele getallen tussen 0 en $2^n$-1. Tussengelegen waarden van de bijbehorende spanningen worden dus naar boven of naar beneden afgerond. De **resolutie**, de spanningswaarde tussen twee opeenvolgende getallen van de ADC, is het bereik gedeeld door de grootste waarden van de ADC. Gemiddeld zal hierdoor een meting de helft van dit bedrag fout zitten. Deze fout heet de **kwantiseringsfout**

$$Kwantiseringsfout(V) = \frac{1}{2^n - 1} * \frac{Bereik}{2} \tag{2}$$

Een voorbeeld: voor een 10-bits ADC met een bereik van 10 Volt is de kwantiseringsfout (1/1023)*5 = 0,005 V of 5mV.

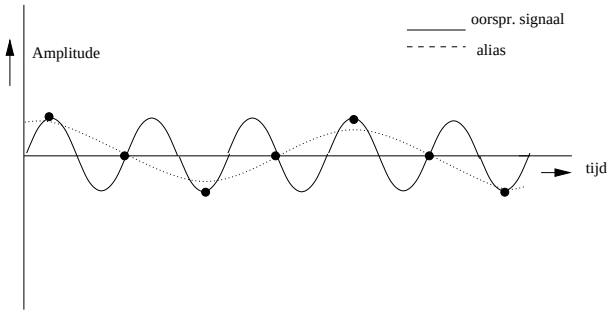
De ADC heeft een zekere tijd nodig (zijn **conversietijd**) om de te meten spanning om te zetten in een getal. Hoe sneller hij dat doet, des te sneller

kan hij achter elkaar door meten. Uiteraard is dat van belang als je snel wisselende signalen (met een hoge frequentie) wilt bemeten.

Ook van belang is de zogeheten **openingstijd**, de tijd dat de ADC naar het signaal kijkt (iets anders dan de conversietijd !). Ook hiervoor geldt hoe korter hoe beter omdat de geproduceerde getalswaarde een gemiddelde is over die openingstijd. Is die openingstijd erg groot, dan is de kans groot dat het signaal tijdens het meten is verlopen.

## Het meten met een ADC

In een DSP-programma wordt een signaal met vaste tussenpozen met een ADC bemonsterd. De keuze van de bemonsteringsfrequentie (= 1/ tijd tussen twee metingen) hangt af van de aard van het signaal. Nyquist heeft aangetoond dat die bemonsteringsfrequentie **twee maal zo hoog moet zijn als de hoogste frequentie die in het signaal voorkomt** omdat anders in het resultaat valse signalen of **aliassen** ontstaan. Figuur 2 laat dat zien voor de bemonstering van een sinusvormig signaal met frequentie f dat met een frequentie $f_s = 1,33f$ bemonsterd wordt. Omdat signalen altijd




Figuur 2: Alias t.g.v. te lage bemonsteringsfrequentie

ruis bevatten met ook hoge frequenties daarin moeten we er voor zorgen dat het aan de ADC aangeboden signaal geen hogere frequenties dan de helft van de bemonsteringsfrequentie bevat. Dat doen we met een **laagdoorlaat** filter dat frequenties boven $f_s/2$ tot een te verwaarlozen amplitude verzwakt. Zo'n filter wordt een **anti-aliasfilter** genoemd. en moet dus **voor** de ADC worden geschakeld.

# DIGITAAL/ANALOOG CONVERSIE

De digitaal/analoog omzetter of converter heeft als taak getallen die de computer levert om te zetten (te converteren) naar spanningen. De meest belangrijke eigenschappen zijn:

- spanningsbereik
- nauwkeurigheid
- insteltijd (Eng. settling time)

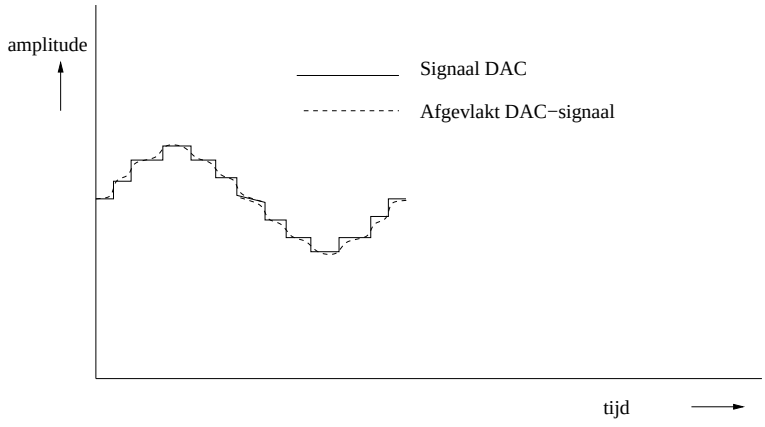
Het spanningsbereik wordt bepaald door de laagste en hoogste spanning die de DAC kan leveren. Sommige DAC's zijn **unipolair** (laagste spanning = 0V), sommige zijn **bipolair** (laagste spanning negatief, hoogste spanning positief). De nauwkeurigheid wordt net als bij de ADC bepaald door het aantal bits waarmee de DAC werkt. Dat aantal bits bepaalt het grootste getal dat de computer naar de DAC kan sturen (= $2^n$-1 met n het aantal bits). Dit grootste getal komt overeen met de hoogste spanning die de DAC kan leveren, het getal 0 met de laagste spanning. Omdat de DAC met gehele getallen werkt, is de kleinste spanningsverandering die de DAC kan realiseren de spanning die overeenkomt met het getal 1:

$$kleinste\_spanningsstap\_DAC = \frac{Spanningsbereik}{2^n - 1} \tag{3}$$

Een voorbeeld: een 12-bits DAC met een spanningsbereik van 0 to 5 V heeft als kleinste stap $5/(2^{12} - 1) = 5/4095 = 0{,}00122$ V = 1,2 mV.

De uitgangsspanningen die door een DAC worden geproduceerd zullen dus nooit een vloeiend verloop hebben omdat hij altijd in stapjes werkt. Als die DAC audio moet produceren, moeten die steile trapjes met een laag doorlaatfilter worden afgevlakt (zie figuur 3). Dat kan met een geschikt laagdoorlaat filter.

Soms worden er meer waarden naar de DAC gestuurd dan er in de verwerking zijn uitgerekend. Tussen de in de voorgaande stap berekende waarden wordt dan een extra tussengelegen waarde ingevoegd. Dat proces noemt men **interpolatie**. Het resultaat na het laagdoorlaatfilter is dan gladder. De **insteltijd** (Eng. settling time) van de DAC is de tijd die hij nodig heeft om van het door de computer aangeboden getal een stabiele spanning te maken. Die tijd moet in ieder geval korter zijn dan de conversietijd van de ADC in het systeem om niet de beperkende factor te zijn.




Figuur 3: Afvlakken DAC-signaal

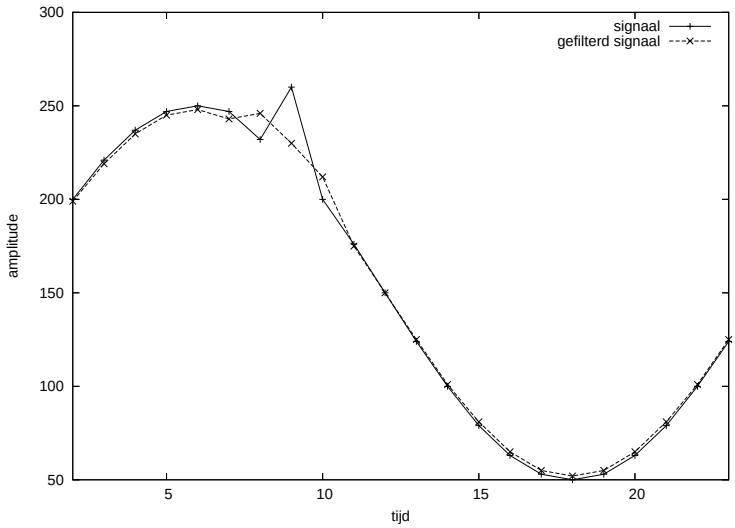
# DIGITAAL FILTEREN

De meest belangrijke toepassing van DSP's is ongetwijfeld het filteren van signalen. Ter herinnering: signaalfilteren is het opknappen van signalen door het verzwakken van ongewenste signaalcomponenten en/of het bevorderen van gewenste componenten. Het volgende voorbeeld wil laten zien hoe met een rekenprogramma een signaal kan worden opgeknapt. Stel we hebben met een ADC een sinusvormig signaal bemonsterd terwijl er op dat sinusvormige signaal een uitschieter zit ten gevolge van een storing (zie de "puist" in figuur 4).

Op het oog is direct te zien dat de "puist"net na de positive top van de sinus een ongerechtigheid is die niet in het signaal thuishoort, maar van een storing komt. Hoe kunnen we het signaal nu zó bewerken dat het "gladder" wordt?

Eén mogelijkheid daartoe is de waarde van elk meetpunt te middelen met zijn naaste buren. Daarmee gaan we plotselinge grote veranderingen tegen. Kijken we eens naar de tabel van de meetwaarden van de ADC (tabel 1) behorende bij figuur 4. In de laatste kolom staan de over 3 meetwaarden berekende gemiddelde waarden. Voor het eerste en laatste meetpunt kunnen we die niet berekenen omdat die resp. geen linker- en geen rechterbuur hebben. Als we die waarden samen met de oorspronkelijke waarden in een figuur zetten (figuur 4) zien we dat die uitschieter behoorlijk "in het gelid" is gekomen. Maar we zien ook dat er effect is op de "goede" punten; de toppen worden wat lager.

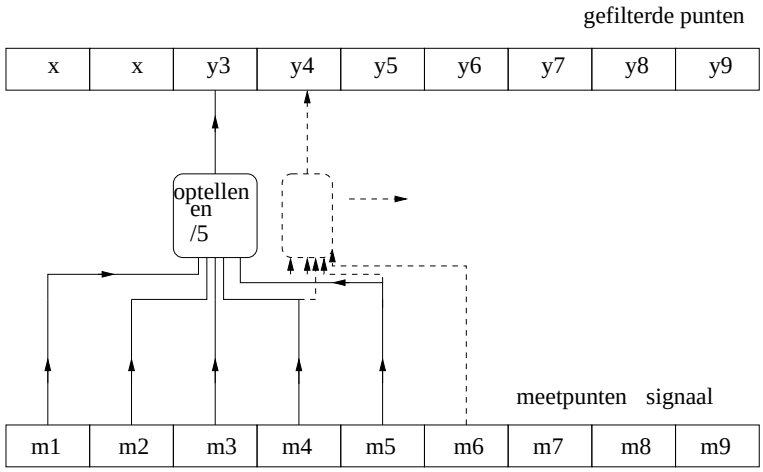
We zouden het effect van de uitschieter nog verder kunnen wegwerken




Figuur 4: "puist" op sinussignaal

door te middelen met meer buren, bijvoorbeeld met twee punten links en rechts. De curve wordt dan nog gladder, maar de toppen worden nog meer afgeplat. Deze zogeheten **voortschrijdend of glijdend gemiddelde** filters hebben een laagdoorlaat werking. Hun werking kan dus worden aangepast door de keuze van het aantal punten van het oorspronkelijke signaal dat in de bewerking wordt meegenomen. Bekijken we het hart van de rekenprocedure wat nader voor wat betreft de eerste stap dan zien we (figuur 5) dat de waarde van één punt van het gefilterde signaal tot stand komt door een berekening uit te voeren op de waarden van een blok van het oorspronkelijke signaal met een oneven aantal punten. Het tijdstip behorende bij het berekende gefilterde signaalpunt ($y3$) komt overeen met het tijdstip waarop het centrale punt m3 van het signaal werd gemeten.

Door het blok van de te bewerken meetpunten van het oorspronkelijke signaal telkens één positie naar rechts op te schuiven (naar 1 tijdstip later) kunnen we zo successievelijk steeds latere punten van het gefilterde signaal berekenen. Alleen aan het begin en het einde kunnen we een paar gefilterde punten niet berekenen omdat we aan het begin een paar voorafgaande metingen missen en omdat aan het einde een paar meetpunten nog niet voorhanden zijn (die liggen nog in de toekomst). In de praktijk is dat geen bezwaar. We nemen genoegen met het feit dat het filter even in moet lopen en aan het eind gaan we pas weer verder als er nieuwe meetpunten binnen zijn. Het betekent

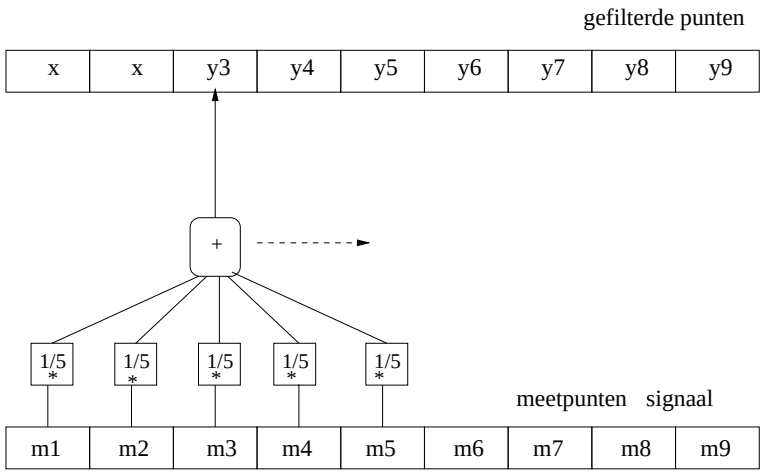



Figuur 5: voortschrijdend gemiddelde over 5 signaalpunten

wel dat het gefilterde signaal altijd **achterloopt** op de werkelijkheid van het oorspronkelijke signaal, ook al zijn we helemaal bij met de berekening. Die vertraging is groter naarmate we het blok van de te bewerken punten groter maken.

Met een kleine aanpassing van deze procedure blijkt het mogelijk om filters met betere doorlaatkarakteristieken te maken en niet alleen laagdoorlaat, ook hoogdoorlaat. Figuur 6 laat zien hoe dat in zijn werk gaat. We blijven als voorbeeld bij een voortschrijdend gemiddelde filter van 5 punten. In plaats van eerst de meetwaarden van het centrale punt en zijn twee linker- en rechterburen op te tellen en vervolgens deze som door 5 te delen om het gemiddelde te krijgen, berekenen we dit gemiddelde door eerst elk van de vijf meetpunten te vermenigvuldigen met een factor 1/5 en daarna tellen we die producten op en zetten het resultaat op de plaats van het gefilterde signaal tegenover het centrale punt van het blok. Dit resultaat is weer gewoon het gemiddelde over die 5 punten.

Ook bij deze procedure schuiven we dit blok met wat nu de filtercoëfficiënten heten stap voor stap langs het hele oorspronkelijke signaal en berekenen zo het gefilterde signaal. Het resultaat is dus weer het zelfde als bij de eerdere berekening van het voortschrijdende gemiddelde. Maar, en dat is de kracht van deze methode die **convolutie** heet, we hoeven deze factoren niet aan



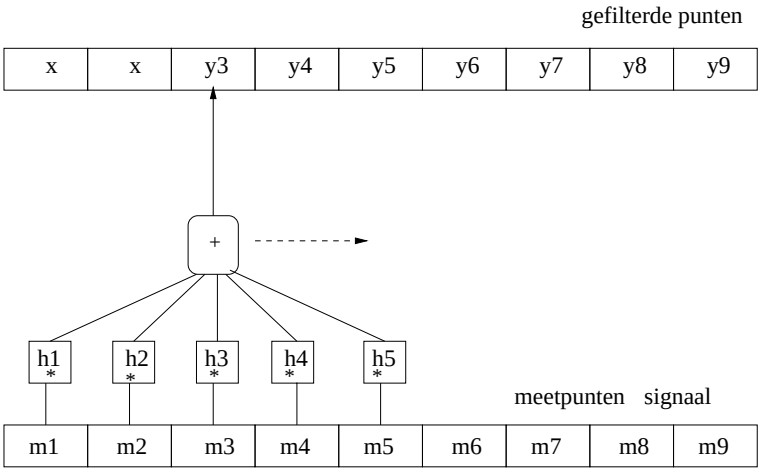

Figuur 6: voortschrijdend gemiddelde m.b.v. eerst vermenigvuldigen met factor, dan optellen

elkaar gelijk te kiezen (zoals hier alle coëfficiënten wel gelijk zijn aan 1/5). Door andere coëfficiënten te kiezen kunnen we een totaal andere filterkarakteristiek verkrijgen, bijvoorbeeld een hoogdoorlaat karakteristiek. Figuur 7 laat deze methode in algemene zin zien waarbij $h1, h2, h3, ...$ staan voor een reeks filtercoëfficiënten die horen bij het gewenste filter, qua grootte en aantal.

De algemene procedure is nu als volgt. Leg het blok met de zogeheten filtercoëfficiënten met zijn eerste coëfficiënt gelijk met de eerste meetwaarde. Vermenigvuldig elke meetwaarde van het signaal binnen het bereik van het blok met zijn tegenoverliggende filtercoëfficiënt en tel deze producten bij elkaar op. Het resultaat is nu de gefilterde waarde tegenover het centrale punt van het blok. Stuur deze waarde naar de uitgang, schuif het blok 1 positie op en herhaal deze bewerkingen.

Een digitaal filter dat op deze manier is gerealiseerd, heet een **FIR-filter** waarbij **FIR** staat voor Finite Impulse Response. Op deze term wordt hier niet verder ingegaan, maar wel is van belang te weten dat deze filters onvoorwaardelijk stabiel zijn, d.w.z. het resultaat is bruikbaar ongeacht de aard van het aangeboden signaal.

De tegenhanger van een FIR-filter is een **IIR-filter** ( Infinite Impulse response). Zo'n filter gedraagt zich alleen maar netjes als ook het ingangssignaal




Figuur 7: Algemene vorm convolutieberekening

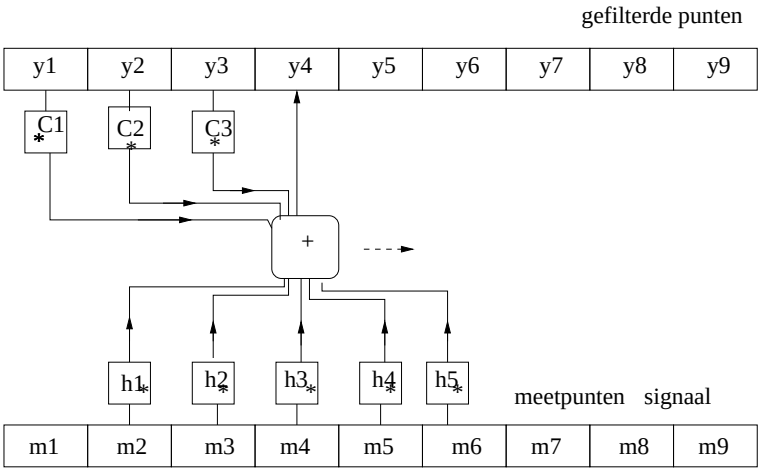
zich niet al te vreemd gedraagt.

Het kenmerkende verschil tussen een IIR-filter en een FIR-filter is dat een IIR-filter terugkoppelingen kent en een FIR-filter niet. Die terugkoppelingen kunnen de oorzaak zijn van "wild" gedrag van het filter (bijvoorbeeld het gaan oscilleren). Figuur 8 laat zien hoe dat kan worden gerealiseerd. Een gefilterd resultaat wordt nu dus niet alleen berekend door gemeten signaalwaarden met een filterfactor te vermenigvuldigen en de resultaten bij elkaar op te tellen, voor een IIR-filter worden ook eerder berekende resultaten (voorafgaande y-waarden) met een filterfactor vermenigvuldigd en opgeteld om bij te dragen aan het resultaat. In figuur 8 zijn dat de filter factoren c1, c2, c3, ...

Met FIR- zowel als IIR-filters kunnen heel fraaie filterkarakteristieken worden gerealiseerd, waarbij IIR-filters vaak met veel minder coëfficiënten voor een zelfde prestatie toe kunnen. Maar zoals reeds gezegd, het kan zijn dat ze instabiel worden en dan heb je er niets aan.

## Digitaal filteren in het frequentiedomein

De hierboven beschreven FIR- en IIR-filters werken in het zogeheten **tijdsdomein**, dat wil zeggen ze werken direct met een reeks op verschillende tijdstippen gemeten waarden van het signaal. Als je een grafiek zou maken



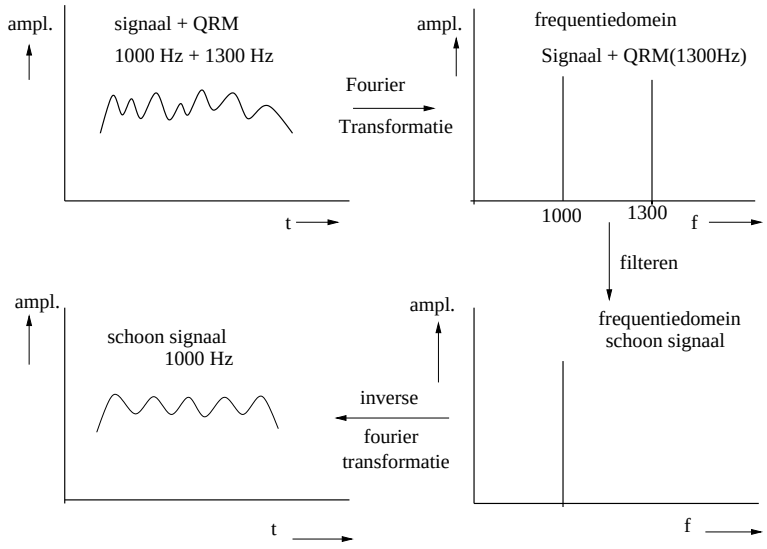

Figuur 8: Voorbeeld IIR-filter met 3 terugkoppelingen

met die meetwaarden op de verticale as en op de horizontale as de tijdstippen waarop die waarden zijn gemeten krijg je een beeld van het signaal zoals dat er op een oscilloscoop uit zou zien, het signaal tegen de tijd dus.

In sommige gevallen willen we graag op een andere manier naar een signaal kijken. Bijvooorbeeld als we willen zien of een zendersignaal ook nog ongewenste componenten bevat. Dan gebruiken we een spctrumanalyzer. Die geeft op de verticale as eveneens de amplitude van het signaal weer, maar op de horizontale as is nu de frequentie van het signaal uitgezet. Zo'n weergave noemen we een weergave van het signaal in het **frequentiedomein** en heet ook wel een **spectrum**. Computers blijken in staat om de meetwaarden van een signaal die met een ADC zijn gemeten om te zetten van het tijdsdomein naar het frequentiedomein **en omgekeerd**. Het proces waarmee de computer die omzetting van tijdsdomein naar frequentiedomein bewerkstelligd heet **fouriertransformatie**. De omgekeerde bewerking heet de **inverse fouriertransformatie**.

In digitale signaalverwerking wordt van deze techniek dankbaar gebruikgemaakt omdat een aantal bewerkingen veel makkelijker in het frequentiedomein kunnen worden uitgevoerd. We moeten daarbij vooral denken aan het filteren van signalen. Een voorbeeld. Stel we luisteren naar een CW-station dat we zó hebben afgestemd dat in de luidspreker morse met een mooie 1000 Hz toon klinkt. Plotseling komt er een ander station doorheen dat een 300

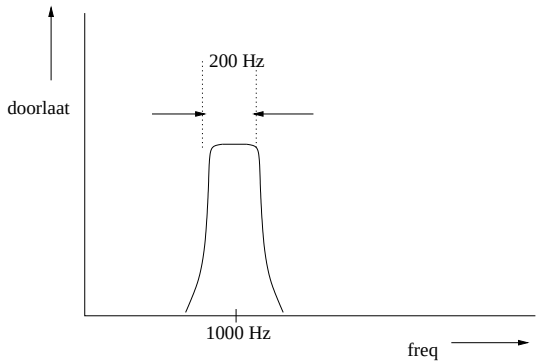
Hz hogere toon, dus 1300 Hz produceert en waar we flink last van hebben. Graag zouden we die 1300 Hz dus weg willen filteren. Met een scherp (bandbreedte 200 Hz) CW-kristalfilter kan dat. Met een DSP kan het ook, met alle eerder genoemde voordelen. Figuur 9 laat zien welke stappen daarvoor nodig zijn.




Figuur 9: Fourierfilteren van 1300 Hz storing op 1000 HzCW-signaal

Het filteren in het frequentiedomein is dus betrekkelijk simpel: je zet de amplitude van de frequenties die je niet hebben wilt gewoon op nul en transformeert de zaak daarna weer naar het tijdsdomein. Dat kan wat mooier door het spectrum (de figuur van amplitude tegen de frequentie) punt voor punt te vermenigvuldigen met een filter van de gewenste vorm als getekend in figuur 10. Door de flanken niet al te steil te laten lopen voorkom je dat het filter gaat "rinkelen" net als sommige erg steile kristalfilters voor cw.

Voor het uitvoeren van de fouriertransformatie heeft de dsp-processor behoorlijk wat tijd nodig. Om deze berekening in de pas te kunnen houden met de nieuw binnenkomende signaalmonsters moet je niet alleen een snelle processor hebben, maar moet het programma daarvoor ook zeer efficiënt werken. Cooley en Tukey hebben daarvoor in 1965 een methode ontwikkeld die zeer veel sneller werkt dan alle tot dan toe gebezigde methoden. Voorwaarde daarvoor is dat het aantal meetpunten dat ter transformatie wordt aangeboden een 2-macht is, dus bestaat uit een reeks van 256, 512, 1024, etc.




Figuur 10: DSP CW-filtervorm

meetwaarden. Deze methode staat bekend als de Fast Fourier Transform, afgekort (**FFT**. Als niet precies zo 2-macht aan datapunten beschikbaar is, kan gebruik worden gemaakt van de **DFT**, de Discrete Fourier Transformatie. Die werkt met een willekeurig aantal meetpunten,maar is wel een factor 1000 trager dan de FFT. Voor beide vormen is ook software beschikbaar voor de omgekeerde (inverse) transformatie.

Tabel 1: Meetwaarden sinus met "puist"

| tijd | meetwaarde (V) | gefilterde waarde (V) |
|---|---|---|
| 1 | 150 | x |
| 2 | 176 | 175 |
| 3 | 200 | 199 |
| 4 | 221 | 219 |
| 5 | 237 | 235 |
| 6 | 247 | 245 |
| 7 | 250 | 248 |
| 8 | 247 | 243 |
| 9 | 232 | 246 |
| 10 | 260 | 230 |
| 11 | 200 | 212 |
| 12 | 176 | 175 |
| 13 | 150 | 150 |
| 14 | 124 | 125 |
| 15 | 100 | 101 |
| 16 | 79 | 81 |
| 17 | 63 | 65 |
| 18 | 53 | 55 |
| 19 | 50 | 52 |
| 20 | 53 | 55 |
| 21 | 63 | 65 |
| 22 | 79 | 81 |
| 23 | 100 | 101 |
| 24 | 124 | 125 |
| 25 | 150 | x |