Den Teknisk-Naturvidenskabelige Basisuddannelse

Storgruppe 9736

Titel: Digital signalbehandling

Projektperiode: P2 2/2/98 - 29/5/98

Projektgruppe: 347

Deltagere:

Claus Albøge
Mads Christensen
Tonny Gregersen
Karsten Jensen
Peter Korsgaard
Robert Stepien

Vejledere:

Børge Lindberg
Susanne Flydtkjær

Oplagstal: 10

Sideantal: 128

Bilag: Diskette

Synopsis:

Dette projekt omhandler digital signalbehandling (DSP). Den generelle udvikling i elektronikindustrien, der består i en overgang fra fremstilling af dedikeret hardware til anvendelsen af generelle hardwareløsninger, der tilpasses de specifikke formål gennem software, behandles.

En række konsekvenser af denne udvikling undersøges.

Med udgangspunkt i konstruktionen af en digital mixer med en 3-bånds equalizer til PC søges det demonstreret, at udviklingen medfører en række tekniske fordele.

På baggrund af erfaringerne med denne konstruktion overvejes det, hvilke krav udviklingen stiller til metoder til udvikling af software.

Det konkluderes, at der er behov for at fokusere på udviklingen af nye metoder til softwareudvikling.

Forord

Nærværende skrivelse er en rapport udarbejdet af projektgruppe 347 som dokumentation for dennes arbejde i p2-projektperioden 1998 på den Teknisk-Naturvidenskabelige Basisuddannelse.

Vi takker Morten Lydorf og Børge Lindberg for det udmærkede projektenheds-kursus om Digital Signalbehandling.

Læsevejledning

I rapporten vil henvisninger til litteratur ske på følgende måde: [<forkortelse>, s. <side>]. Værket, som forkortelsen henviser til, kan findes i litteraturhenvisningen.

Hvis kilderne er internetsider bliver de i rapporten refereret til på følgende måde: [<forkortelse>]. Forkortelsen henviser til en URL, som kan findes i kildehenvisningen.

Engelske fagudtryk og danske oversættelser anvendes, hvor det findes passende. Figurer, tabeller og ligninger er nummereret endimensionalt fortløbende hver for sig. Der er anvendt amerikansk decimaltalsangivelse i grafer og tal fra MATLAB. Pseudokode angives på grå baggrund.

Specielt teorikapitlerne om digital signalbehandling anbefales læst i kronologisk rækkefølge, da teorien i de tidligere kaptiler forudsættes i de senere.

Rapporten er udarbejdet af:

Claus Albøge	**Mads Christensen**	**Tonny Gregersen**
Karsten Jensen	**Peter Korsgaard**	**Robert Stepien**

Indholdsfortegnelse

1. Indledning

DSP er en forkortelse for det engelske Digital Signal Processing, der på dansk kaldes digital signalbehandling. DSP anvendes i forbindelse med redigering, detektering og generering af analoge signaler, ved hjælp at digitale metoder. En af DSP'ens forcer er, at det benytter sig af generel hardware med specifik software, hvor man ellers er vant til dedikeret hardware.
Derfor er man også inden for den seneste årrække begyndt at anvende DSP på flere og flere områder; bl.a. inden for audio- og videobehandling. Det betyder, at man gradvist er begyndt at udvikle generel hardware med specifik software, frem for specifik hardware.

Det er netop i denne udvikling nærværende projekts tager sit udgangspunkt. Hidtil er al signalbehandling foregået analogt og har delvist skabt et samfund, der var indrettet efter den analoge verdens normer. Og det giver selvsagt en række problemer, at skulle overgå til en verden, hvor der gøres brug af digitale systemer.
Udover disse samfundsmæssige aspekter, har der fra projektgruppen samtidigt lydt et ønske om at lave digital lydbehandling, dels for at opnå kendskab til den tekniske del af DSP, men også for at vise, hvorledes mulighederne er for at implementere forskellige dele af et DSP baseret system.

2. Problemanalyse

2.1 Udviklingen i elektronikindustrien

2.1.1 Fra hardware til software

Traditionelt blev analoge elektriske kredsløb specialfremstillet til at opfylde specifikke mål. Skulle de anvendes til andre formål eller blev specifikationerne ændret bare en lille smule, måtte de tilpasses ved rekonstruktion af måske hele eller store dele af kredsløbene. Hardware fremstillet med et bestemt mål for øje, kaldes dedikeret hardware.
De senere år er udviklingen i elektronikindustrien gået mod anvendelsen af generelle hardware-løsninger, der tilpasses de specifikke formål gennem software, altså ved programmering. Eller som Otto Vinter, leder af softwareprocesforbedringer hos Brüel & Kjær Sound & Vibration Measurement A/S, formulerer det: "Det er snart kun selve transduceren, som opfanger lyden eller vibrationerne og omsætter dem til elektriske impulser, der er rigtig elektronik. Resten er PC, Windows og C++ programmering" [Hansson]. Han anslår, at 80 pct. af udviklingstiden går på programmering. [Hansson].

Udviklingen skyldes i høj grad, at mikroelektroniske komponenters ydeevne i forhold til prisen har været konstant stigende de seneste årtier. [ITcenter] Denne udvikling medfører ændringer for producenterne på følgende områder :

- **Produktion**

  At softwaren integreres i elektroniske løsninger medfører, at programudviklere i stigende grad bliver en del af det nye produktionsapperat. Kompleksiteten og fleksibiliteteten af udstyret til produktion af elektriske kredsløb falder drastigt, da al tilpasning foretages ved programmering. Hardwareløsningerne kan masseproduceres som enkelte generelle modulopbyggede løsninger, i stedet for flere forskellige dedikerede løsninger.

- **Arbejdsmiljø**

  Det fysiske arbejdsmiljø ændrer sig ikke. Udviklingen af elektriske kredsløb er gennem en del år foregået ved hjælp af computere.

- **Personale/uddannelse**

  Udviklerne skal ikke længere tegne diagrammer og vælge komponenter, men programmere. [Hansson]. Dette vil enten kræve ansættelse af nyuddannede dataingeniører, dataloger, eleltronikingerniører osv., eller efteruddannelse af gammelt personale.

- **Udviklingsmetoder**

  Der anvendes andre udviklingsmetoder til fremstilling af software end til fremstilling af hardware. Udviklingen af programmer er ikke så moden en disciplin som elektronik. Desuden er metoderne under forandring. Gennem de sidste ti år er analyse og design gået fra at være struktureret til at være objektorienteret. Desuden påpeger Otto Vinter, at udviklingen ikke bør skilles i en hardware-del og en software-del men samles - metoderne til en sådan udvikling eksisterer endnu ikke. [Hansson]

- **Økonomi**

  Økonomisk er der flere aspekter. Et af de vigtigste er den samlede fortjeneste, der i dag er ved produktion af DSP systemer. Problemet er for lav profit! [Sound Pro] På trods af lave materiale-omkostninger, er der tendens til at nettofortjenesten bliver lav, idet der påføres nye udgiftsposter, såsom online-brugerhjælp, opdatering af software o.lign.

2.1.2 Digital signalbehandling

Digital signalbehandling har parallelt med den generelle udvikling i elektronikindustrien overtaget en stadig større dele af den analoge signalbehandling. Digital signalbehandling kan simpelt beskrives som:

- digitalt at behandle, detektere eller generere analoge signaler.

Signalerne kan være elektromagnetisk stråling eller f.eks. mekaniske bølger opfanget eller udsendt til omverdenen.

Selve den digitale behandling foregår i signalprocessorer. Opfanges signalerne skal de analoge, tidskontinuerte signaler, forinden omdannes til digitale, tidsdiskrete, signaler. Dette gøres ved sampling - konvertering fra antalog til digital (A/D).

Af typiske traditionelle anvendelsesområder kan nævnes:

- Elektronisk filtrering og equalizering
- Spektrumanalyse
- Datatransmission
- Spredt spektrum radiokommunikation
- Procesregulering og -styring
- Test- og måleudstyr
- Lagring og behandling af HIFI- og videosignaler
- Talegenkendelse og talesyntese

[Hüche, s. 15].

Branchen er i en rivende udvikling, og der bliver stadig flere anvendelsesområder. De her nævnte stammer fra en DSP-bog fra 1986 - og er muligvis ikke fyldestgørende.

2.1.3 Signalprocessorer

Signalprocessoren udfører selve den digitale signalbehandling, der består i en algoritme. Kravet til processoren, det, der gør den velegnet som signalprocessor, er dens evne til at udføre aritmiske beregninger, multiplikationer og additioner, med stor hastighed og nøjagtighed. Hastigheden er specielt en kritisk faktor i systemer, der skal fungere i realtid.
De realiseres typisk som en integreret, eller delvist integreret, signalprocessorkreds, der indeholder multiplier, adder, accumulator, program- og lagerhukommelse plus input- og

outputbuffere. [Hüche, s. 19]. I dag programmeres signalprocessorerne i C, i modsætning til tidligere, hvor de blev programeret i assembler [Hansson].

2.1.4 Tekniske fordele og ulemper

Overgangen, fra den dedikerede hardware til generelle hardwareløsninger, tilpasset de enkelte opgaver gennem softwaren, kan tilskrives en række uomtvistelige tekniske fordele i forbindelse med digitale signalbehandlingssystemer:

- **Kontrolleret signal-støjforhold og dynamikområde**
  Disse kan bestemmes, og kontrolleres, gennem digitale systemers bitantal, hvorimod de i analoge systemer er mere ukontrollerbare.

- **Nemmere/billigere specifikationsændringer**
  Hele elektriske kredsløb, printplader, skal ikke omkonstrueres, hvilket kan være en langsommelig og omkostningskrævende proces sammenlignet med ændringer i software. Udgifterne til lønninger og materialer bliver således reduceret.

- **Nemmere/billigere fejlretning**
  Serier af produkter skal ikke nødvendigvis kasseres, men blot softwareopgraderes ved fejl. Ellers gælder samme fordele som ved specifikationsændringer. Hardwaren kan selvfølgelig stadig være behæftet med fejl.

- **Minimering af problemer i forbindelse med komponentdrift og komponenttolerance**
  Elektriske komponenters karakteristika kan være mere eller mindre temperaturafhængige og ældes. Med begrænsning af hardwareomfanget, dvs. komponentantal, mindskes disse problemer.

- **Kraftig reduktion af hardwareomfang**
  En større dele af hardwaren bliver, ved overgangen til digital signalbehandling, mikroprocessorer, og hardwareomfanget reduceres således.

- **Ingen trimning af produktet**

  De varierende karakteristikker for de forskellige komponenter, der måtte indgå i et analogt kredsløb, kan nødvendiggøre en trimning af produktet. Alt efter produktet foretages tidskrævende trimning manuelt af personale.

- **Realisering af analogt ikke realiserbare funktioner**

  Blandt andet adaptive systemer kan være problematiske at realisere analogt. Filtre af meget høj orden vil gøre analoge systemer yderst komplekse, eller helt umulige. I digital signalbehandling er filterorden udelukkende et spørgsmål om processorkraft.

- **Mere intuitive brugerflader**

  Brugerfladen kan konstrueres traditionelt med fysiske drejeskiver og knapper, men kan også realiseres gennem en skærmbaseret brugerflade, som f.eks. Windows, med mulighed for en mere intuitiv betjening. Den kan, alt efter hardwarekonfiguration, tilpasses forskellige målgrupper [Hansson].

Et par fænomener, der må betegnes som ulemper ved digital signalbehandling, er aliasering og kvantiseringsstøj. [Hüche, s. 15].

2.1.5 PC'en

Pc-teknologien er et markant eksempel på væksten af generelle hardwareløsninger, der tilpasses specielle formål ved hjælp af software, og mange virksomheder mærker konsekvenserne af pc'ens fremmarch. Det gælder f.eks. den danske måleinstrumentindustri. Hvor man i denne industri tidligere udviklede egne apparater helt fra bunden, og derfor havde en unik kompetence, specielt indenfor hardware-design, som vanskeligt lod sig efterligne, er det nu pc'er, der udgør fundamentet i moderne måleinstrumenter. Produktudvikling er derfor i høj grad blevet softwareudvikling på pc-platforme. Det betyder, at store dele af produkterne hardwaremæssigt minder mere og mere om hinanden. Det er både en fordel, idet produkterne lettere kan udvikles på samme platform, og en ulempe, da den specielle teknologiske kompetence alene er knyttet til sensorerne, der opsamler måledata [Ramskov].

Den stigende integration af pc-teknolgien i måleinstrumenter kan medføre, at kunderne venter samme stærkt faldende priser, som for andre pc-produkter. Denne forventning kan udløse et pres på de meget høje avancer på det enkelte apparat, der traditionelt er i denne industri [Ramskov].

Ikke alle måleinstrumentvirksomheder har været i stand til at omstille sig til de nye krav. Resultatet er mange steder stagnation eller fald i omsætningen. Udviklingen har mange steder medført udskiftning af ledelse og ejere, heraf nogle til udenlandske investorer, inden for de sidste få år.[Ramskov].

2.1.6 Softwareudvikling

Det ovenfor beskrevne eksempel er udtryk for nogle generelle tendenser. Erhvervslivet befinder sig som følge af disse tendenser midt i en strukturændring, idet f.eks. elektroniske apparaters funktion og dermed virksomhedernes konkurrenceevne, i et hidtil uset og stadig stigende omfang bestemmes af den software, apparaterne er forsynet med[ITcenter]

Som nævnt angiver Otto Vinter, at 80 pct. af udviklingstiden går til software. Leif Christiansen, projektchef hos GN Nettest, Elmi division, anslår tallet til at være 60-70 pct. afhængigt af hvilket udstyr vi taler om [Hansson]. Dette giver anledning til visse problemer, der er forbundet med udviklingen af programmer. Christiansen siger, at det ikke er ualmindeligt med en overskridelse af persontiden på op til 50 pct. [Hansson]. Ifølge en rapport fra forskningsministeriet medgår over halvdelen af udviklingsomkostningerne, i mange virksomheder, til software [ITcenter].

Metoderne til softwareudvikling er ikke så modne som de, der anvendes til udvikling af hardware. Det ses blandt andet af, at der ifølge rapporten fra forskningsministeriet er et behov for forbedret produktivitet i udviklingen [ITcenter]. Virksomhedernes ledelser har heller ikke indarbejdet metoderne [Hansson]. Store softwareprojekter kræver særlige kompetencer, som erhvervslivet endnu ikke besidder [ITcenter].

Tidligere har softwareudvikling hovedsageligt bestået i udvikling af administrative systemer. Den stigende andel af softwareudvikling i forbindelse med elektroniksystemer, f.eks. instrumenter og måleapparatur, har dog ikke medført problemer, der adskiller sig fra de allerede kendte, i forbindelse med udvikling af administrative systemer. Den største forskel består i, at elektroniksystemerne i højere grad skal afvikles i realtidssammenhæng, hvorimod de administrative systemer har flere databaser. [Hansson]

En del af problemerne med softwareudvikling, i forbindelse med elektroniksystmer, tilskriver Otto Vinter, at virksomheder forsøger at dele udviklingen op i to adskilte dele. Dette, mener han, bevirker, at koordineringen mellem delene ikke bliver god nok. Der skal være et fælles ansvar for, at produktet bliver færdigt til tiden. [Hansson]. Han mener endvidere, at der mangler gode softwareudviklere.

Det er specielt igennem de indledende faser af udviklingen, dvs. fasen frem mod en specifikation, at problemerne skabes, fastslår rapporten fra forskningsministeriet. Den påpeger endvidere, at "Omfanget af forskningsindsatsen på Danmarks universiteter og højere læreranstalter på softwareområdet er langtfra øget i samme udstrækning som erhvervslivets indsats på området. Samtidig har der været en tendens til, at mange af universitetsmiljøernes forskning inden for software ikke har været rettet mod de problemer, som erhvervslivet kæmper med. Kontakten mellem universiteterne og de godkendte teknologiske serviceinstitutter [..] på softwareområdet har også traditionelt været ret spinkel" [ITcenter].

3. Afgrænsning og problemformulering

En stadig større del af udviklingen i elektronikindustrien er softwareudvikling. Denne udvikling er behæftet med både muligheder og begrænsninger. Der er behov for at styrke softwareudviklingsmetoder, specielt hvad angår de tidlige faser af softwareudviklingen og der er behov for bedre softwarekvalitet, karakteriseret ved overlegen ydeevne, lang levetid, udstrakt genbrug, elegant design, brugerglæde, fejlfrie og effektive programmer samt problemfri vedligeholdelse [ITcenter].

Med udgangspunkt i vores analyse vil vi forsøge at demonstrere visse tekniske fordele som overgangen fra analog til digital signalbehandling har.

Vi vil konstruere en digital lydmixer med følgende specifikationer:

2 indgange, 1 udgang.

Lyden fra de to indgange skal behandles i to seperate kanaler med følgende effekter:

- Gain 3 bånd EQ
- Noise Gate
- Volume
- VU-meter

De to kanaler skal herefter mixes vha. en crossfader.
Sidst i signalvejen skal der være en samlet volume-regulering.
Mixeren skal fungere i realtid, og skal benytte lydkort af typen Sound Blaster 16 til input/output

Med baggrund i konstruktionen af mixeren vil vi forsøge at vise, hvor enkelt det er at ændre specifikationerne for denne ved hjælp af relativt simple ændringer i programmet.

Som programmeringssprog har vi valgt Pascal, som vi har et godt forudkendskab til.

Da der i elektronikindustrien er et behov for, at softwareudviklingen foregår metodisk, har vi tilstræbt en metodisk udvikling af programmet. Som metode har vi valgt elementer fra metoden Struktureret Programudvikling (SPU). Dette valg er foretaget på baggrund af en vurdering af vores ressourcer, da vi havde brug for en let tilgængelig metode. Vi er bekendt med at SPU ikke længere er tidssvarende, idet man i dag fortrinsvis benytter Objektorienteret Programudvikling.

Problemløsningsfasen består af to dele: en del omhandlende digital signalbehandling og en omhandlende fremstillingen af mixeren.

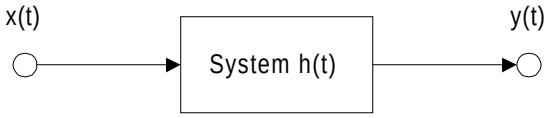
4. Digital signalbehandling

I fremstillingen af hele kapitlet om digital signalbehandling er Erik Hüches "Digital Signalbehandling" [Hüche] og James H. McClellan, Ronald W. Schafer og Mark A. Yoders: DSP First [DSP First] anvendt.

4.1 Tidskontinuert systemteori

4.1.1 System

Et system er et elektronisk kredsløb, en formel eller en funktion, der behandler signaler. Se **Figur 1.**



***Figur 1**: System, med input x(t) og output y(t). Sammenhængen mellem x(t) og y(t) kaldes h(t), impulsresponsen.*

Eksempelvis er et lavpasfilter et system, der behandler indgangssignalet x(t), ved kun at lade de lave frekvenser passere til udgangssignalet y(t). Systemet kan både realiseres som et analogt elektrisk kredsløb, passivt eller aktivt, og digitalt som en formel eller funktion. Sammenhængen mellem indgangssignalet x(t) og udgangssignalet y(t) kan beskrives med impulsresponsen h(t) i tidsdomænet, eller f.eks. med den komplekse overføringsfunktion H(ω) i frekvensdomænet.

I gennemgangen af teorien bag DSP, vil vi begrænse os til systemer med følgende egenskaber:

Linearitet*:*

Et system er lineært, hvis $ax_1(t) + bx_2(t) \rightarrow ay_1(t) + by_2(t)$ (1)

Dette svarer til, at systemet opfylder både homogenitetsprincippet

$$x(t) \rightarrow y(t) \Leftrightarrow ax(t) \rightarrow ay(t) \quad (2)$$

og superpositionsprincippet

hvis $x_1(t) \rightarrow y_1(t)$ og $x_2(t) \rightarrow y_2(t)$ gælder følgende, hvis superpositionsprincippet er opfyldt:

$$x_1(t) + x_2(t) \rightarrow y_1(t) + y_2(t) \quad (3)$$

Tidsinvarians:

Et tidsinvariant system er karakteriseret ved, at dets parametre er uafhængige af tiden. Systemet vil altså opføre sig ens til tiden t som til tiden t_1:

$$x(t) \rightarrow y(t) \Leftrightarrow x(t - t_1) \rightarrow y(t - t_1) \quad (4)$$

Kausalitet:

Et system er kausalt, hvis det ikke afgiver udgangsrespons, før det har fået tilført et indgangssignal.

4.1.2 Fourierrækker

Enhver periodisk funktion, f(t), kan skrives som en Fourierrække, som er en uendelig sum af sinus- og cosinuskomposanter med forskellige amplituder. Dette kan noteres på kort form som

$$\sum_{m=0}^{\infty} a_m \cdot \cos(m\omega t) + b_m \cdot \sin(m\omega t) \quad (5)$$

Sinus- og cosinuskomposanterne har hver en frekvens, som er et heltalsmultiplum af grundfrekvensen ω, også kaldet den m'te harmoniske af grundfrekvensen. Koefficienterne a_m og b_m kaldes reelle Fourierkoefficienter, og angiver den numeriske værdi af sinus- og

cosinuskomposanternes amplitude.

Ønskes et ukendt periodisk signal beskrevet som en Fourierrække, kan Fourierkoefficienterne a_m og b_m beregnes med følgende integraler, hvor $T = \frac{1}{f} = \frac{2\pi}{\omega}$ er det periodiske signals periodetid:

$$a_0 = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)\,dt \tag{6}$$

$$a_m = \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)\cdot\cos(m\omega t)\,dt \tag{7}$$

$$b_m = \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)\cdot\sin(m\omega t)\,dt \tag{8}$$

4.1.3 Eksponentiel Fourierrække

Indsættes den eksponentielle notationsform for sinus- og cosinussignalet i (5), fås:

$$\begin{aligned} f(t) &= \sum_{m=0}^{\infty} \frac{a_m}{2}(e^{jm\omega t} + e^{-jm\omega t}) + \frac{b_m}{2j}(e^{jm\omega t} - e^{-jm\omega t}) \\ &= \sum_{m=0}^{\infty} \frac{1}{2}(a_m - jb_m)e^{jm\omega t} + \frac{1}{2}(a_m + jb_m)e^{-jm\omega t} \\ &= \sum_{m=0}^{\infty} c_m e^{jm\omega t} + c_m^* e^{-jm\omega t} \end{aligned} \tag{9}$$

hvor $c_m = \frac{1}{2}(a_m - jb_m)$ og $c_m^* = \frac{1}{2}(a_m + jb_m)$ er de komplekse Fourierkoefficienter.

Da $c_m^* = c_{-m}$ kan ligningen forkortes til:

$$f(t) = \sum_{m=0}^{\infty} c_m e^{jm\omega t} + c_{-m}e^{j(-m)\omega t} = \sum_{m=-\infty}^{\infty} c_m e^{jm\omega t} \tag{10}$$

Denne ligning repræsenterer den eksponentielle Fourierrække. Det periodiske signal er her

beskrevet af en sum af komplekse elementarsvingninger med amplituden $|c_m|$, fasen $\angle\phi_m$ (hvor $c_m = |c_m| \angle\phi_m = |c_m| e^{j\phi_m}$) og frekvensen $\pm m\omega$.

De komplekse Fourierkoefficienter c_m kan beregnes med integralet

$$c_m = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) \cdot e^{-jm\omega t} \, dt \qquad (11)$$

hvor $m = -\infty K -1,0,1K \infty$

4.1.4 Fouriertransformation

Med Fourierrækker var det kun muligt at beskrive periodiske funktioner. Med Fouriertransformation indføres en metode til at beskrive aperiodiske signaler. Denne signaltype gennemløber kun samme forløb én gang i tidsintervallet $-\infty < t < \infty$, derfor kan signalet opfattes som et periodisk signal med uendelig periodetid T. Det periodiske signals frekvensspektrum er diskret. Afstanden mellem spektrallinierne er $\omega = 2\pi f = 2\pi/T$. Da periodetiden for et aperiodisk signal går mod uendelig, må afstanden mellem spektrallinierne gå mod 0. Altså er frekvensspektret for et aperiodisk signal ikke diskret, men kontinuert. Den diskrete frekvensvariabel mω erstattes med en kontinuert variabel ω, ligesom de komplekse Fourierkoefficienter c_m erstattes af en kontinuert funktion F(ω). Hvis vi indsætter de nævnte ændringer i (11) får vi Fouriertransformationsudtrykket (Forward Fouriertransform):

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt = \Im[f(t)] \qquad (12)$$

hvor F(ω) er en kompleks funktion, og kan afbildes som et amplitudespektrum (lige funktion) og et fasespektrum (ulige funktion).

Det inverse Fouriertransformationsudtryk skrives som

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{j\omega t}d\omega = \Im^{-1}[F(\omega)] \qquad (13)$$

4.1.5 Fouriertransformationsregler

Regel	**f(t)**	**F(ω)**
F1	$af_1(t) + bf_2(t)$	$aF_1(\omega) + bF_2(\omega)$
F2	$f(t - t_0)$	$F(\omega)e^{-j\omega t_0}$
F3	$f(t)e^{j\omega_0 t}$	$F(\omega - \omega_0)$
F4	$\int_{-\infty}^{\infty} f_1(t) \cdot f_2(t-\tau)d\tau$	$F_1(\omega) \cdot F_2(\omega)$
F5	$f_1(t) \cdot f_2(t)$	$\int_{-\infty}^{\infty} F_1(\omega) \cdot F_2(\omega - \Omega)d\Omega$

Tabel 1: Tabel over Fouriertransformationsregler. [Hüche, s. 90].

F1: Linearitetsreglen

Fouriertransformationen er en lineær proces. Superpositionsbetingelsen er opfyldt og der er tale om en homogen proces. Derfor kan en sum transformeres ledvis, og koefficienter (a og b) går uændret igennem transformationen.

F2: Tidsforskydningsreglen

En tidsforskydning af f(t) på t_0 sekunder medfører en faseforskydning.

F3: Frekvensforskydningsreglen

En multiplikation i tidsdomænet med en kompleks elementarsvingning med frekvensen ω medfører en frekvensforskydning på ω_0.

F4: Tidsdomænefoldning

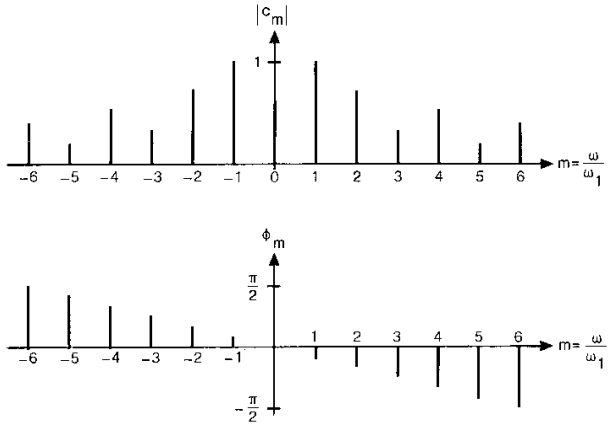
Foldning af to tidsfunktioner svarer til, at de to tidsfunktioners spektrumfunktioner multipliceres frekvens for frekvens.

F5: Frekvensdomænefoldning:

Multiplikation af to tidsfunktioner svarer til foldning af de to signalers spektrumfunktioner.

4.1.6 Komplekst frekvensspektrum

Da et periodisk signal, som nævnt ovenover, består af komplekse elementarsvingninger med frekvensen mω, hvor ω er grundfrekvensen og m er heltal mellem -∞ og ∞, vil signalets frekvensspektrum være diskret. Et sådant spektrum består af både et amplitudespektrum og et fasespektrum, da c_m er komplekse koefficienter. I amplitudespektret tegnes $|c_m|$ som funktion af frekvensen. I fasespektret tegnes ϕ_m som funktion af frekvensen. Se Figur 2



***Figur 2**: Diskret amplitude- og fasespectrum for et periodisk signal. [Hüche, s. 75].*

Det ses at amplitudespektret er en lige funktion $(f(x) = f(-x))$ mens fasespektret er ulige $(f(x) = -f(-x))$.

4.2 Laplacetransformation

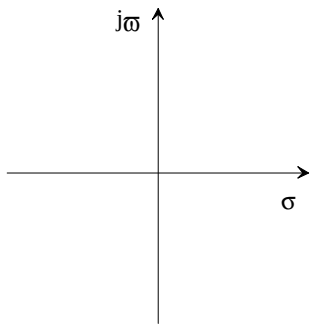
Laplacetransformation er en vigtig transformationsmetode, som bruges til at overføre en tidsfunktion, f(t), fra tidsdomænet til s-domænet (Laplacedomænet).

4.2.1 S-planen

Til Laplacetransformation bruges en kompleks frekvensvariabel, s, som er defineret på følgende måde:

$$s = \sigma + j\omega \qquad (14)$$

hvor σ angiver den reelle frekvensvariabel og $j\omega$ angiver den imaginære del. s-planen er et todimensionalt komplekst talplan, som grafisk ser ud på følgende måde, Figur **3**:



***Figur 3**: s-planen.*

4.2.2 Transformationen

Laplacetransformation kan bruges ved kausale tidsfunktioner, dvs. funktioner, hvor:

- f(t) = 0 for t < 0
- f(t) er defineret for t > 0

- f(t) kan være defineret for t = 0, men den behøver ikke at være det.

Laplacetransformation er defineret ved Laplaceintegralet, hvis integrationsgrænse går fra 0 til ∞:

$$F(s) = \int_0^{\infty} f(t) \cdot e^{-st}\, dt \qquad (15)$$

F(s) er således den Laplacetransformerede f(t), hvilke også kan udtrykkes på følgende måde:

$$F(s) = £[f(t)] \qquad (16)$$

Tidsfunktionen f(t) er ved Laplacetransformationen blevet overført fra tidsdomænet til s-domænet.
Modsat kan man ved invers Laplacetransformation fremstille tidsfunktionen f(t), hvis man kender F(s). Denne transformation udtrykkes på følgende måde:

$$f(t) = £^{-1}[F(s)] \qquad (17)$$

Der er lavet forskellige tabeller med Laplacetransformationspar, som letter omskrivningsprocessen. Det er dog ikke altid muligt at bruge disse tabeller til invers Laplacetransformation. Man er således nød til at foretage den inverse transformation ved partielbrøksopløsning (Se Appendiks B).

4.2.3 Regler for Laplacetransformationer

Generelt gælder der en række regler, som bruges ved Laplacetransformationer:

Regel	f(t)	F(s)
L1	$af_1(t) + bf_2(t)$	$aF_1(s) + bF_2(s)$
L2	$f(t-t_0)$	$F(s) \cdot e^{-st_0}$
L3	$f(t) \cdot e^{s_0 t}$	$F(s-s_0)$
L4	$\frac{d}{dt} f(t)$	$s \cdot F(s)$
L5	$\int f(t)dt$	$\frac{1}{s} \cdot F(s)$

Tabel 2: *Regler for Laplacetransformation. [Hüche, s. 97].*

- **L1** er liniaritetsreglen, som angiver at Laplacetransformationen opfylder superpositionsprincippet. En sum af tidsfunktioner kan dermed transformeres ledvis uden at eventuelle koefficienter ændres.
- **L2** er tidsforskydningsreglen, som siger, at en tidsforsinkelse af tidsfunktionen f(t) på t_0 sekunder svarer til at multiplicere F(s) med e^{-st_0}.
- **L3** er frekvensforskydningsreglen, som angiver, at multiplikation af tidsfunktionen f(t) og den komplekse elementarsvingning $e^{s_0 t}$ forårsager en forskydning af F(s)'s poler og nulpunkter i s-planen med værdien s_0.
- **L4** er differentiationsreglen. Ifølge denne regel svarer differentiation i tidsdomænet til at multiplicere den Laplacetransformerede med s.
- **L5** er integrationsreglen, som siger, at integration i tidsdomænet svarer til at dividere den Laplacetransformerede med s.

4.2.4 Overføringsfunktioner i s-domænet

Hvis man har den Laplacetransformerede til en kendt stimulusfunktion X(s), og kender systemets udgangsrespons i s-domænet Y(s), kan man bestemme systemets overføringsfunktion i s-domænet H(s) på følgende måde:

$$H(s) = \frac{Y(s)}{X(s)} \tag{18}$$

Generelt vil H(s) have følgende form:

$$H(s) = \frac{a_0 + a_1 s + a_2 s^2 + ... + a_m s^m}{b_0 + b_1 s + b_2 s^2 + ... + b_n s^n} \tag{19}$$

hvor m og n angiver højeste eksponent af s i henholdsvis tæller og nævner.

4.2.5 Poler og nulpunkter

Poler og nulpunkter er kritiske frekvenser, som bestemmes vha. systemets overføringsfunktion H(s). De bruges ved indtegning i et pol-nulpunktsdiagram i s-planen til at give en grafisk beskrivelse af det pågældende systems egenskaber.

Et givet systems poler er defineret som de værdier af s, for hvilke $H(s) = \infty$. Da $H(s) \to \infty$ for $X(s) \to 0$, er polerne identiske med rødderne i H(s)'s nævnerpolynomium X(s). De afbildes i et pol-nulpunktsdiagram med et kryds. Se Figur **4**

Nulpunkterne for et givet system er defineret som de værdier af s, for hvilke H(s) = 0. Da H(s) $\to$ 0 for Y(s) $\to$ 0 er nulpunkterne identiske med rødderne i H(s)'s tællerpolynomium. De afbildes i et pol-nulpunktsdiagram med en cirkel. Se Figur **4**

Eksempel.
Et 2. ordens system er givet ved følgende overføringsfunktion:

$$H(s) = \frac{s^2 + 4s + 4}{s^2 + 4s + 13}$$

Først bestemmes nulpunkterne: $s^2 + 4s + 4 \Rightarrow s = -2$

Der er altså et dobbelt nulpunkt i –2.

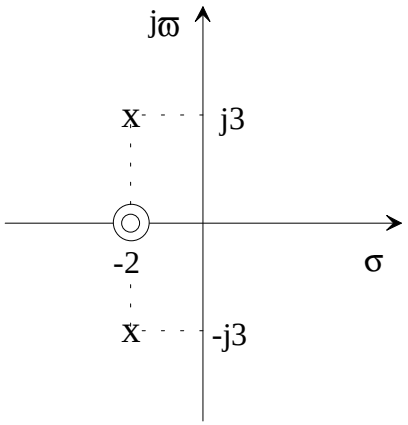
Derefter bestemmes polerne: $s^2 + 4s + 13 \Rightarrow s = -2 \pm j3$

Der er altså følgende to komplekst konjugerede poler:

$s = -2 + j3$

$s^* = -2 - j3$

Systemets pol-nulpunktsdiagram kommer dermed til at se således ud.:



***Figur 4**: Pol-nulpunktsdiagram.*

4.2.6 Impulsrespons

En vigtig indikator for et systems stabilitetsforhold er systemets impulsrespons. Impulsresponsen h(t) er identisk med et systems udgangsrespons y(t), hvis stimulusfunktionen er enhedspulsen $\delta(t)$.

Enhedsimpulsen $\delta(t)$ er karakteriseret ved følgende egenskaber:

$$\int_{-\infty}^{\infty} \delta(t)\,dt = 1 \text{ og } \delta(t) = 0 \text{ for } t \neq 0 \qquad (20)$$

Enhedsimpulsen eksisterer altså kun for t = 0 og arealet under funktionens kurve er pr. definition 1. Enhedsimpulsen bruges som stimulusfunktion ved teoretisk systemanalyse, da dens Laplacetrasformerede er meget simpel:

$$£[\,\delta(t)\,] = 1 \tag{21}$$

Ved at bruge enhedsimpulsen som stimulusfunktion, dvs. $x(t) = \delta(t)$, fås følgende

$$X(s) = £[x(t)] = £[\delta(t)] = 1 \tag{22}$$

Udgangsresponsen i s-domænet bliver dermed:

$$Y(s) = X(s) \cdot H(s) = 1 \cdot H(s) = H(s) \tag{23}$$

Det er herefter muligt at bestemme impulsresponsen h(t) ved invers Laplacetrasformation:

$$h(t) = £^{-1}[H(s)] \tag{24}$$

Afhængig af impulsresponsen kan et system have en af de tre følgende stabilitetstilstande. Se Figur 5

1. Stabilt system

Ved et stabilt system går dets impulsrespons mod nul for t gående mod ∞:

$$h(t) \to 0 \text{ for } t \to \infty \tag{25}$$

Polerne for stabile systemer ligger alle i s-planens venstre halvplan.

2. Marginalt stabilt system

Ved et marginalt stabilt system er impulsresponsen enten en fast DC-værdi eller oscillerer sinusformet med konstant amplitude.

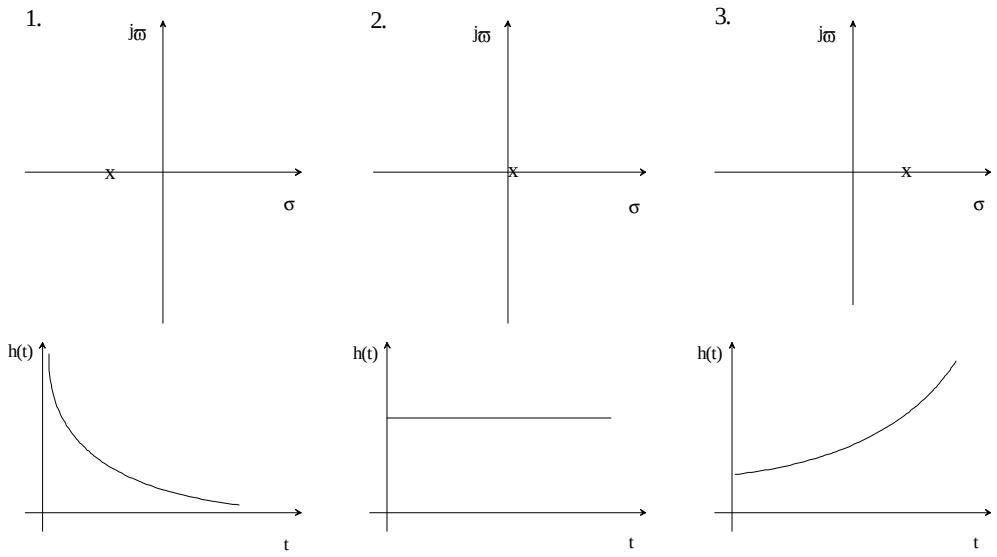
Polerne for marginalt stabile systemer ligger på s-planens $j\omega$-akse.

3. Ustabilt system

Ved et ustabilt system går impulsresponsen mod uendelig:

$$h(t) \to \infty \text{ for } t \to \infty \qquad (26)$$

Polerne for et ustabilt system ligger i s-planens højre halvplan.



Figur 5*: De tre stabilitetstilstande: 1) Stabilt system 2) Marginalt stabilt system 3) Ustabilt system. Øverst ses polplacering, og nedenunder de tilhørende impulsresponser.*

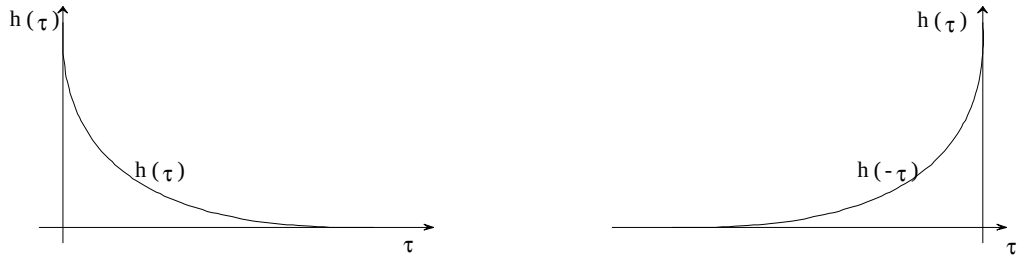
4.2.7 Foldning

Et meget anvendt værktøj til systemanalyse er foldning. For ethvert system med lineær og tidsinvariant impulsrespons h(t) kan systemets udgangsrespons y(t) beregnes ved alle stimulusfunktioner x(t) vha. foldningsintegralet:

$$y(t) = \int_{-\infty}^{\infty} x(t)h(t-\tau)\,d\tau = x(t) * h(t) \qquad (27)$$

hvor "$*$" læses som "foldet med". Variablen τ er en såkaldt "dummy variabel", dvs en regneteknisk spidsfindighed, og t er den reelle tidsvariabel. Integrationsgrænserne er i foldningsintegralet sat til at være $-\infty$ og ∞, men vil begge i praktis blive erstattet med endelige værdier.

Grunden til at ovennævnte integrale kaldes foldningsintegralet er, at impulsresponsen $h(\tau)$ ved udførsel af foldningsintegralet bliver spejlet omkring y-aksen ("foldet") med en mulig tidsforskydning. Herunder er vist en foldning i det specialtilfælde, hvor t = 0:



***Figur 6**: Foldning ved t = 0.*

Ved foldning bestemmes grænserne for foldningen ved at betragte grænseværdierne for stimulusfunktionen ($x(\tau) = 0$) og multiplikatorfunktionen ($h(t-\tau) = 0$).

Foldning er en kommutativ proces, hvilket vil sige, at de to funktioner i foldningsintegralet kan bytte plads uden at foldningsresultatet ændres:

$$y(t) = x(t) * h(t) = h(t) * x(t) \qquad (28)$$

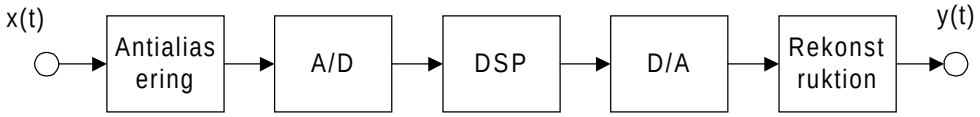
eller

$$y(t) = \int_{-\infty}^{\infty} x(\tau) \cdot h(t-\tau)\,d\tau = \int_{-\infty}^{\infty} h(\tau) \cdot x(t-\tau)\,d\tau \qquad (29)$$

Ved at udnytte denne egenskab kan man i nogle tilfælde lette udregningerne.

4.3 Samplingsteori

4.3.1 Sampling



***Figur 7**: Blokdiagram over et typisk digitalt signalbehandlingssystem til realtidsbehandling af analoge signaler.*

Ved en sampleproces ændres et tidskontinuert signal $x(t)$ til et tidsdiskret signal $x_s(t)$. Dette sker ved at måle amplituden af det kontinuerte signal til nogle bestemte sampletidspunkter. Ved jævn eller periodisk sampling er afstanden mellem sampletidspunkterne, dvs sampleintervallet, ens. Sampleintervallet benævnes T, og dets reciprokke værdi er samplefrekvensen $f_s = \frac{1}{T}$.

Sampletidspunkterne eller samplesignalet kan beskrives med en funktion af tiden, som består af tidsforskudte Dirac-impulser, der kun eksisterer i tidspunkterne nT, hvor n er heltal mellem -∞ og ∞.

$$p(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT) \qquad (30)$$

Sampleprocessen kan nu beskrives som en multiplikation af signalet p(t) med det tidskontinuerte signal x(t):

$$x_s(t) = x(t) \cdot p(t) = x(t) \cdot \sum_{n=-\infty}^{\infty} \delta(t - nT) \qquad (31)$$

Samplede værdier til tiden nT angives som: $x[n]$.

4.3.2 Nyquist-frekvensen

x(t) kan i ligningen (31) substitueres med x(nT), da p(t) er nul for $t \neq nT$:

$$x_s(t) = x(nT) \cdot \sum_{n=-\infty}^{\infty} \delta(t - nT) = \sum_{n=-\infty}^{\infty} x(nT) \cdot \delta(t - nT) \qquad (32)$$

Produktet af x(t) og p(t) svarer til en amplitudemodulation, og da man ved, at modulation har indflydelse på frekvensspektret, vil vi undersøge spektret for det samplede signal, $x_s(t)$.

Da p(t) er et periodisk signal med grundfrekvensen $\omega_s = 2\pi f_s$, kan samplesignalet skrives som en uendelig kompleks Fourierrække med Fourierkoefficienterne c_m:

$$p(t) = \sum_{m=-\infty}^{\infty} c_m \cdot e^{jm\omega_s t}, \text{ hvor } c_m = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} p(t) \cdot e^{-jm\omega_s t}\, dt \qquad (33)$$

I tidsrummet $-\frac{T}{2} < t < \frac{T}{2}$ er p(t) = 0 for $t \neq 0$. p(t) kan derfor erstattes med δ(t). Desuden kan man nøjes med at se på tidspunktet t = 0:

$$c_m = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} \delta(t) \cdot e^{-jm\omega_s t}\, dt = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} \delta(t) \cdot e^{-j0}\, dt = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} \delta(t)\, dt = \frac{1}{T} \qquad (34)$$

p(t)'s Fourierrække bliver altså:

$$p(t) = \sum_{m=-\infty}^{\infty} \frac{1}{T} \cdot e^{jm\omega_s t} \qquad (35)$$

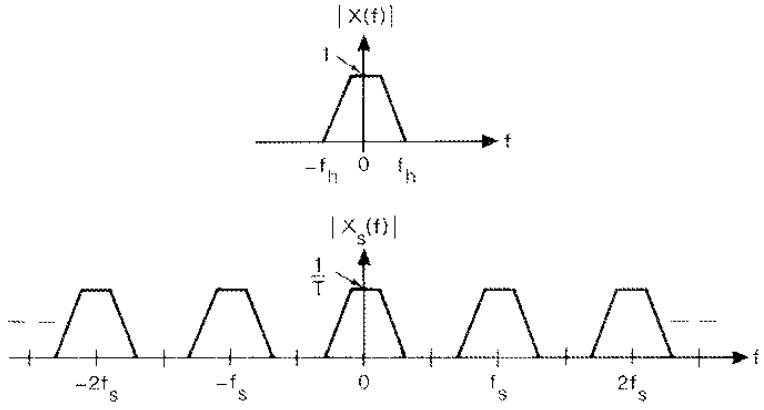
Det impulssamplede signal $x_s(t)$ kan nu skrives som:

$$x_s(t) = x(t) \cdot p(t) = x(t) \cdot \sum_{m=-\infty}^{\infty} \frac{1}{T} \cdot e^{jm\omega_s t} = \frac{1}{T} \cdot \sum_{m=-\infty}^{\infty} x(t) \cdot e^{jm\omega_s t} \qquad (36)$$

Ved at Fouriertransformere det impulssamplede signal $x_s(t)$ kan vi finde signalets spektrum $X_s(\omega)$. Ledet $x(t) \cdot e^{jm\omega_s t}$ kan ifølge frekvensforskydningsregelen F3 i Tabel 1 skrives som $X(\omega - m\omega_s)$. Ifølge linearitetsregelen F1 i Tabel 1, forbliver summationen uændret:

$$X_s(\omega) = \frac{1}{T} \cdot \sum_{m=-\infty}^{\infty} X(\omega - m\omega_s) \Leftrightarrow X_s(f) = \frac{1}{T} \cdot \sum_{m=-\infty}^{\infty} X(f - mf_s) \qquad (37)$$

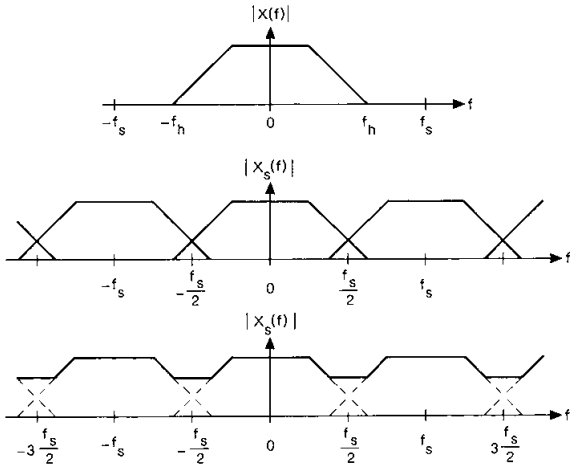
Vi kan nu se, at det impulssamplede signals spektrum består af x(t)'s oprindelige spetrum samt et uendeligt antal frekvensforskudte kopier af dette spektrum forskudt med et heltalsmultiplum af samplefrekvensen f_s.



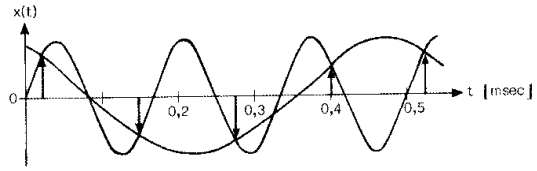
***Figur 8**: Amplitudespektrum for impulssamplet signal.*
[Hüche, s. 203].

En korrekt gendannelse af det oprindelige tidskontinuerte signal bliver kun muligt ved at lavpasfiltrere det tidskontinuerte udgangssignal ved frekvensen $\frac{f_s}{2}$. Men gengivelsen bliver alligevel ikke korrekt, hvis det tidskontinuerte signal, som samples, indeholder frekvenskomposanter, der overstiger den halve samplefrekvens $\frac{f_s}{2}$. Disse frekvenskomposanter

vil under samplingsprocessen blive forskudt med et heltalsmultiplum af samplefrekvensen, og vil derfor også blive forskudt så de kommer til at ligge under den halve samplefrekvens. Dette fænomen kaldes aliaseringsstøj.



***Figur 9**: a) Spektrum for signal, der ikke er tilstrækkeligt båndbreddebegrænset. b) Delspektre for impulssamplet version af samme signal. c) Resulterende aliaseret amplitudespektrum. [Hüche, s. 206].*



***Figur 10**: 6 kHz sinussignal impulssamplet med 8 kHz [Hüche, s. 207].*

Aliaseringsstøj undgås, hvis det tidskontinuerte signals højeste samplede frekvens f_h, er lavere end den halve samplefrekvens, også kaldt Nyquist frekvensen eller foldningsfrekvensen:

$$f_h < \frac{f_s}{2}, \text{ eller } f_h < f_0, \text{ hvor } f_0 = \frac{f_s}{2} = \frac{1}{2T} \qquad (38)$$

Før samplingen skal det tidskontinuerte signal derfor altid lavpasfiltreres ved f_0.

4.4 Tidsdiskrete systemer

Ved et tidsdiskret system forstår vi en proces, der ved hjælp af en algoritme, omformer en sekvens af heltalsværdier, til en ny sekvens af heltalsværdier, og hvor både input og output er tidsdiskrete signaler.

4.4.1 FIR-systemer

En vigtig klasse af tidsdiskrete systemer er FIR-filtre. De er defineret ved følgende relation mellem input og output

$$y[n] = \sum_{k=0}^{M} b_k \cdot x[n-k] \tag{39}$$

M betegner filterets orden, og antallet af koefficienter betegner filterets længde. Denne vil altid være M + 1.

4.4.2 Enhedsimpuls

Enhedsimpulsen er en simpel sekvens, idet den kun har en værdi forskellig fra nul, nemlig når n = 0. Den matematiske notoation for enhedsimpulsen er som følger.

$$\delta[n] = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases} \tag{40}$$

En forskudt enhedsimpuls $\delta[n-k]$ afviger fra nul når $n-k=0$. Dette begreb om den forskudte enhedsimpuls kan anvendes til at repræsentere systemer og signaler.

4.4.3 Impuls respons

Begrebet enhedsimpuls kan bruges til at definere impulsresponsen for systemer med endelig længde. Denne kan defineres som outputtet af et system, hvor inputtet er enhedsimpulsen, og

den er repræsenteret ved følgende formel

$$h[n] = \sum_{k=0}^{M} b_k \cdot \delta[n-k] \qquad (41)$$

Impulsresponsen for FIR-filteret, når inputtet er enhedsimpulsen, er en række af koefficienter i en differensligning. Hvis h[n] = 0 for n < 0 og for n > M, er længden af impulsrespons-sekvensen endelig. Dette er forklaringen på navnet FIR, der står for *Finite Impulse Response.*

4.4.4 Foldning

Vi kan nu udlede et udtryk for FIR-filterets output udtrykt ved impulsresponsen. Da filterkoefficienterne er identiske med værdierne for impulsresponsen, kan vi udskifte b_k i (39) med h[k], og får så:

$$y[n] = \sum_{k=0}^{M} h[k] \cdot x[n-k] \qquad (42)$$

Relationen mellem FIR-filterets output og input udtrykt ved inputtet og impulsresponsen, er en foldning af sekvenserne x[n] og h[n].

4.5 Z-transformationer

I analysen af tidsdiskrete systemer kan det i visse tilfælde være hensigtsmæssigt at bevæge sig fra et matematisk domæne til et andet. Vi har tidligere introduceret Laplace/s-domænet og tids/n-domænet. I dette kapitel vil vi introducere z-domænet, der ligesom s-domænet er et komplekst taldomæne. Relationen mellem s- og z-domænet vil blive behandlet i et senere kapitel, mens vi i det følgende vil beskrive z-transformationen, når vi bevæger os fra n- til z-domænet.

4.5.1 Definition af z-transformationen

Et signal med endelig længde kan repræsenteres ved følgende relation

$$x[n] = \sum_{k=0}^{M} x[k] \cdot \delta[n-k] \tag{43}$$

Den z-transformerede af dette signal er defineret ved ligningen

$$x[z] = \sum_{k=0}^{M} x[k] \cdot z^{-k} \tag{44}$$

hvor z repræsenterer et vilkårligt komplekst tal.

X(z) er et polynom med variablen z^{-1}, hvis koefficienter er lig værdien af sekvensen x[n], således at den k-te værdi er lig koefficienten til den k-te grad af z^{-1}. Dette fremgår af følgende sammenhæng.

$x[n] = \delta[n - n_0]$ er i z-domænet repræsenteret ved $X(z) = z^{-n_0}$.

4.5.2 Overføringsfunktionen

For at vise den praktiske betydning af z-transformationen, vil vi se på den generelle

differensligning

$$y[n] = \sum_{k=0}^{M} b_k \cdot x[n-k] \tag{45}$$

En alternativ repræsentation af input-output relationen er foldningssummen

$$y[n] = x[n] * h[n] \tag{46}$$

Hvor h[n] er impulsresponsen. Denne er ifølge (41) givet ved relationen

$$h[n] = \sum_{k=0}^{M} b_k \cdot \delta[n-k] \tag{47}$$

Vi lader inputtet til systemet være signalet:

$$x[n] = z^n \text{ for alle n} \tag{48}$$

hvor z er et vilkårligt komplekst tal. Til dette input svarer ifølge (45) outputtet

$$y[n] = \sum_{k=0}^{M} b_k \cdot x[n-k] = \sum_{k=0}^{M} b_k \cdot z^{n-k} = \sum_{k=0}^{M} b_k \cdot z^n \cdot z^{-k} = (\sum_{k=0}^{M} b_k \cdot z^{-k}) \cdot z^n \tag{49}$$

Udtrykket inden i parentesen er et polynomium, hvis form afhænger af koefficienterne b_k, og som betegnes overføringsfunktionen. Af definitionen af z-transformationen ses det at være den z-transformerede af impulsresponsen.

Vi har således vist, at for inputtet $x[n] = z^n$, for alle n er outputtet

$$y[n] = H(z) \cdot z^n \tag{50}$$

Vi skal senere se, at dette er et generelt udsagn, der kan anvendes på alle lineære tidsinvariante systemer.

Vi har altså defineret overføringsfunktionen

$$H(z) = \sum_{k=0}^{M} b_k \cdot z^{-k} \tag{51}$$

og kan nu finde denne for en vilkårlig differensligning, idet

$$H(z) = y[n] \cdot z^{-n} \tag{52}$$

4.5.3 Superposition

Z-transformationen har nogle egenskaber som det kan være nyttigt at kende. Den første følger af det forhold, at z-transformationen er en lineær transformation, hvilket fremgår af definitionen af z-transformationen. Det betyder, at den ligeledes opfylder betingelsen for superposition. Hvis

$$x[n] = a \cdot x_1[n] + b \cdot x_2[n] \tag{53}$$

da er den z-transformerede

$$X(z) = a \cdot X_1(z) + b \cdot X_2(z) \tag{54}$$

4.5.4 Invers z-transformation

En anden vigtig egenskab ved z-transformationen er, at et delay på n samples svarer til en multiplikation af den z-transformerede med z^{-n}. Denne egenskab er betydningsfuld, da den angiver, hvorledes vi bevæger os fra z-domænet til n-domænet. Dette kaldes invers z-transformation (Se Appendiks E).

4.5.5 Z-transformation for systemer med uendelig impulsrespons

Indtil nu har vi defineret z-transformationen for signaler med endelig længde. Det er muligt at udvide definitionen til signaler med uendelig længde, ved at flytte de øvre og nedre grænser. En mere generel definition ser således ud

$$X(z) = \sum_{n=-\infty}^{\infty} x[n] \cdot z^{-n} \tag{55}$$

En uendelig sum af komplekse tal kan resultere i et uendeligt resultat, hvilket nødvendiggør en undersøgelse af den z-transformerede funktions konvergensområde.

Vi betragter en uendelig sekvens, $h[n] = a^n \cdot u[n]$. Ved at anvende definitionen på z-transformationen på denne får vi

$$H(z) = \sum_{n=0}^{\infty} a^n \cdot z^{-n} = \sum_{n=0}^{\infty} (a \cdot z^{-1})^n \tag{56}$$

hvilket er summen af en geometrisk række, hvor forholdet mellem to efter hinanden følgende udtryk er az^{-1}. Vi kan deraf slutte os til, at hvis $\left|az^{-1}\right| < 1$, så er summen endelig, og kan udtrykkes på en lukket form

$$H(z) = \sum_{n=0}^{\infty} a^n \cdot z^{-n} = \frac{1}{1 - a \cdot z^{-1}} \tag{57}$$

De værdier af z i det komplekse plan, der tilfredsstiller betingelsen for, at H(z) kan udtrykkes på en sådan form, kaldes for konvergensområdet, og i dette område er det altså muligt at finde overføringsfunktionen for et system med uendelig impulsrespons.

4.5.6 Foldning og z-transformationen

Tidligere fremførte vi, at en forsinkelse i tidsdomænet på n samples svarer til en multiplikation med z^{-n} i z-domænet. Denne egenskab, sammen med egenskaben superposition, giver os mulighed for at bevise, at foldning i n-domænet udføres som en multiplikation i z-domænet ved alle lineære tidsinvariante systemer.

Foldning i det diskrete domæne af to sekvenser med endelig længde, er givet ved formel (42). Denne formel bruger vi til at finde den z-transformerede af y[n], hvilket fører til

$$Y(z) = \sum_{k=0}^{M} h[k] \cdot (z^{-k} \cdot X(z)) = (\sum_{k=0}^{M} h[k] \cdot z^{-k}) \cdot X(z) = H(z) \cdot X(z) \qquad (58)$$

Foldning af to endelige sekvenser i tidsdomænet er altså ækvivalent med polynomisk multiplikation i z-domænet i alle lineære tidsinvariante systemer. Dette gælder også i systemer med uendelige impulsresponser (IIR)

4.5.7 Overføringsfunktionen for systemer med uendelig impulsrespons

Overføringsfunktionen for et system med endelig impulsrespons er altid et polynomium. I et system med uendelig impulsrespons (IIR), hvor differensligningen har feedback, viser det sig, at overføringsfunktionen er et forhold mellem to polynomier, en rationel funktion. I dette afsnit vil vi bestemme en sådan overføringsfunktion for en førsteordens differensligning.

Den generelle form for en førsteordens differensligning med feedback er

$$y[n] = a_1 \cdot y[n-1] + b_0 \cdot x[n] + b_1 \cdot x[n-1] \qquad (59)$$

Da denne ligning er gældende for alle værdier af n, kan vi foretage en z-transformation på begge sider af lighedstegnet, hvoraf vi får at

$$Y(z) = a_1 \cdot z^{-1} \cdot Y(z) + b_0 \cdot X(z) + b_1 \cdot z^{-1} \cdot X(z) \qquad (60)$$

Ved at trække udtrykket $(a_1 \cdot z^{-1} \cdot Y(z))$ fra på begge sider af lighedstegnet får vi følgende:

$$Y(z) - a_1 \cdot z^{-1} \cdot Y(z) = b_0 \cdot X(z) + b_1 \cdot z^{-1} \cdot X(z) \quad (61)$$

Dette omskrives til:

$$(1 - a_1 \cdot z^{-1}) \cdot Y(z) = (b_0 + b_1 \cdot z^{-1}) \cdot X(z) \quad (62)$$

Da det er et lineært tidsinvariant system, gælder det, at $Y(z) = H(z) \cdot X(z)$, hvor H(z) er overføringsfunktionen for systemet. Ved at løse ligningen for H(z) får vi

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 \cdot z^{-1}}{1 - a_1 \cdot z^{-1}} \quad (63)$$

Vi har således vist, at overføringsfunktionen for et system med uendelig impulsrespons, er et forhold mellem to polynomier. Dette resultat gælder også for systemer af højere orden.

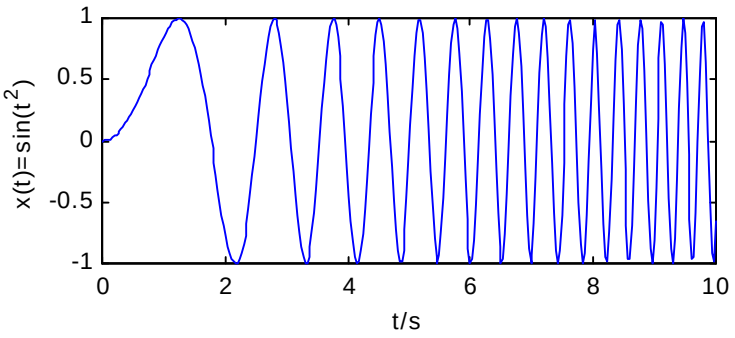
4.6 Frekvensresponsanaslyse

4.6.1 Introduktion

Tilføres et stabilt lineært system et sinusformet stimulus med konstant amplitude måles frekvensresponsen som udgangssekvensen, der ligeledes er en sinusformet sekvens. Der kan være sket en ændring i amplitude og fase – Det er netop disse ændringer, der undersøges i en frekvensresponsanalyse.

Praktisk kan frekvensresponsen af f.eks. en HI-FI højtaler i princippet bestemmes ved af forbinde en sinusgenerator med en højttaler og måle signalet med en mikrofon tilsluttet et voltmeter. Ved at variere frekvensen på generatorsignalet og holde amplituden konstant kan frekvensresponsen plottes som spændingen som funktion af frekvensen.

Kontinuert kan frekvensresponsen undersøges ved at benytte et sinussweep, eks. $x(t) = \sin(t^2)$, som stimulus. Et sådant ser ud som følger:



***Figur 11**: Sinussweep til bestemmelse af frekvensresponsen [MATLAB].*

Sweepet skal foregå så langsomt, at signalet når at falde til ro ved den pågældende frekvens, det vil sige, at alle indsvingsfænomener i systemet er døet ud [Hüche, s. 278]. Dette kaldes "steady state" repsonsanalyse.

Ved tidsdiskrete systemer kan det være vanskeligt at foretage en egentlig måling af frekvensresponsen. Her bestemmes frekvensresponsen matematisk for forskellige diskrete frekvenser i et tilsvarende sweep.

Frekvensresponsanalyse er en generel analysemetode, der ikke kun er blevet anvendt i forbindelse med digital signalbehandling men også i forbindelse med analyse af analoge filtre.

4.6.2 Frekvensresponsfunktionen

I s-planen befinder sinusformede signaler sig på jω-aksen, dvs. $\sigma = 0$. I z-planen overføres s-planens jω-akse i form af enhedscirklen:

$$\begin{aligned} z &= e^{sT} = e^{(\sigma+j\omega)T} = e^{j\omega T} = \cos(\omega T) + j\sin(\omega T) \\ z &= (Re(z), Im(z)) = (\cos(\omega T), \sin(\omega T)) = (1)_{\omega T} \end{aligned} \qquad (64)$$

Hvor σ er den reelle frekvensandel, j den imaginære enhed, ω vinkelhastigheden og T perioden, altså $T = \frac{1}{f_s}$.

De fremkomne frekvenser (f), $0<\omega<\infty$, givet ved sammenhængen $\omega = 2\pi \cdot f$, befinder sig herpå, og en frekvensanalyse behøver således kun omfatte værdierne af z på enhedcirklen, altså: $z = e^{j\omega T}$.

Samplefrekvensen f_S svarer da til argumentet 2π.

Den tidsdiskrete frekvensresponsfunktion er således identisk med overføringsfunktionen H(z). Denne kan så udtrykkes som følgende komplekse funktion:

$$H(z) = H\left(e^{j\omega T}\right) = \left.\frac{Y\left(e^{j\omega T}\right)}{X\left(e^{j\omega T}\right)}\right|_{z=e^{j\omega T}} \qquad (65)$$

Funktionen skrives også som: $H(j\omega) = H(z)$ og på polær form $H(j\omega) = |H(\omega)| \angle \phi(\omega)$

Systemets amplitudefunktion, $|H(\omega)|$, angives normalt i dB, beregnet ved:

$$|H(\omega)| = 20\log\frac{|Y(j\omega)|}{|X(j\omega)|}[dB] \qquad (66))$$

Her er $|X(j\omega)|$ indgangssekvensens amplitude og $|Y(j\omega)|$ udgangssekvensens. $\phi(\omega)$ er fasefunktionen, der angiver faseforskydningen mellem ind- og udgangssekvensen.

4.6.3 Fasefunktionen

Et systems amplitudefunktion, $|H(\omega)|$, er kun den halve sandhed. Overføringsfunktionen er en kompleks funktion, og amplitudefunktionen er kun modulus af denne. Argumentet, fasefunktionen $\phi(\omega)$, er den manglende del.

Fasefunktionen beregnes som summen af nulpunktvektorernes vinkler ψ_i minus summen af polvektorernes vinkler θ_i, jævnfør reglerne for division og multiplikation af komplekse tal. Begge vinkler er i forhold til den positive realakse. Altså ser fasefunktionen således ud:

$$\phi(\omega) = \psi_1 + \psi_2 + .. + \psi_i - (\theta_1 + \theta_2 + ... + \theta_i) \qquad (67)$$

4.6.4 Grafisk frekvensresponsanalyse

På faktoriseret form kan overføringsfunktionen H(z) af n'te orden skrives som:

$$H(z) = a_0\frac{(z-z_1)(z-z_2)...(z-z_n)}{(z-p_1)(z-p_2)...(z-p_n)} \qquad (68)$$

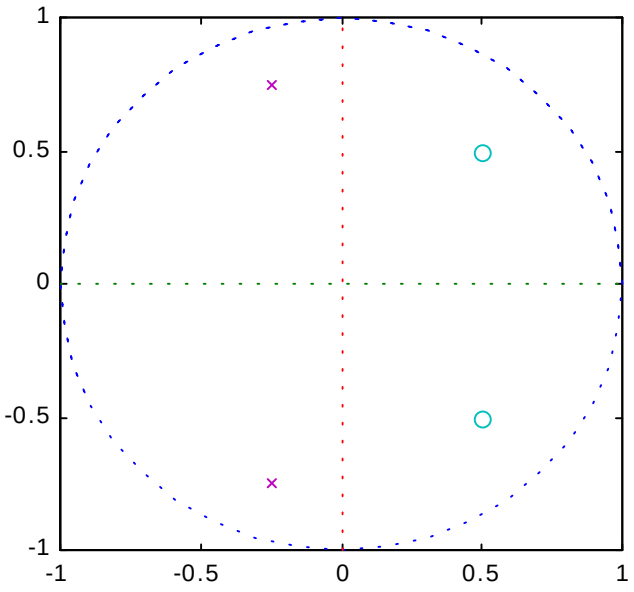
Hvor z_i er funktionens nulpunkter, p_i dens poler og a_0 forstærkningsfaktoren.

Da de enkelte punkter (z_i, p_i og z) er komplekse størrelser angivet ved $z = (\mathrm{Re}(z), \mathrm{Im}(z))$ $= (\cos(\omega T), \sin(\omega T))$ i z-planen, kan de de enkelte størrelser anskues som stedvektorer, og hver tæller- og nævnerfaktor svarer da til en vektordifference mellem polen eller nulpunktet og de evaluerede punkter på enhedscirklen.

Systemets amplitudefunktion kan da beregnes ved:

$$|H(\omega)| = 20 \log a_0 \frac{|z - z_1| \cdot |z - z_2| \cdot \ldots \cdot |z - z_n|}{|z - p_1| \cdot |z - p2| \cdot \ldots \cdot |z - p_n|}, \; z = e^{j\omega T} . \qquad (69)$$

Altså ud fra afstanden mellem poler/nulpunkter og de evaluerede punkter. Det vil sige, at amplitudefunktionen kan udledes direkte af et pol-nulpunktsdiagram, der således er et praktisk analyseværktøj.



***Figur 12**: Frekvensresponsen kan bestemmes ud fra afstandene mellem nulpunkter/poler og frekvenser på enhedscirklenet i et pol-nulpunktsdiagram [MATLAB].*

4.6.5 Fortolkning af pol-nulpunktsdiagrammet

Ved at betragte amplitudefunktionen, $|H(\omega)| = 20\log a_0 \frac{|z-z_1|\cdot|z-z_2|\cdot\ldots\cdot|z-z_n|}{|z-p_1|\cdot|z-p2|\cdot\ldots\cdot|z-p_n|}$, kan man gøre en række vigtige iagtagelser angående betydningen af poler og nulpunkters placering. En pol nær enhedscirklen vil gøre en af nævnerfaktorerne, $|z-p_i|$, i funktionen lille, og dermed medføre et tilsvarende forstærkningspeak for frekvensen på enhedscirklen radialt ud for polen. Ligger polen på enhedscirklen, går $|H(\omega)|$ mod ∞, da der divideres med 0. Tilsvarende vil et nulpunkt nær enhedscirklen medføre en dæmpning, og ligger det på enhedscirklen vil dæmpningen blive -∞ for den pågældende frekvens, da tælleren multipliceres med 0.

Generelt gælder for polers placering:

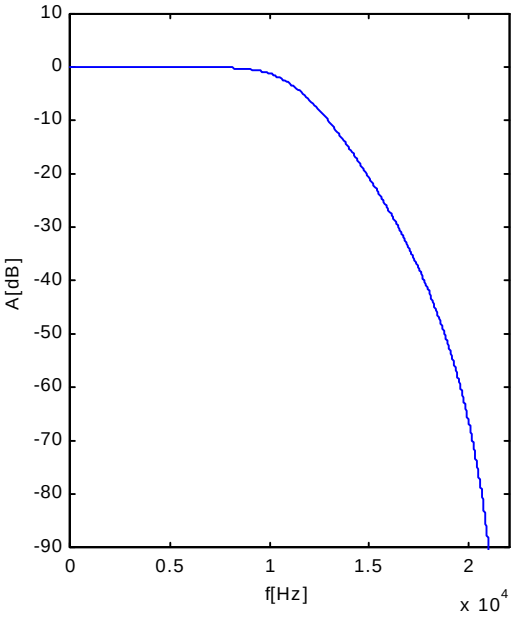
- Ligger en eller flere poler udenfor enhedscirklen, er der tale om et ustabilt system.
- Ligger en eller flere poler på enhedscirklen, er systemet marginalt stabilt.
- Ligger alle poler indefor enhedscirklen, er systemet stabilt.

Hastigheden, hvormed impulsresponsens amplitude ændrer sig, er bestemet af modulus af polens placering i den komplekse z-plan, og oscillationsfrekvensen er bestemt ved argumentet heraf.

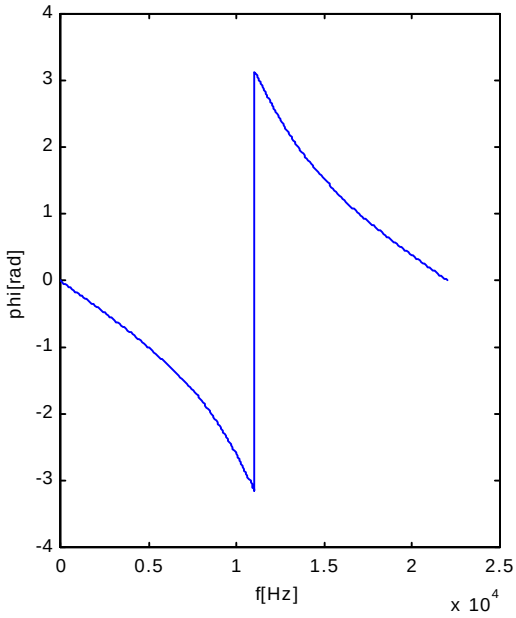
En pol eller et nulpunkt i z-planens origo har kun indflydelse på systemets fasefunktion, idet alle vektorlængderne for enhver frekvens er 1, alt imens vinklerne ψ og θ er proportionale med frekvensen. Desuden vil polerne og nulpunkterne på enhedscirklen forårsage diskontinuitet i fasen for nulpunkts- og polfrekvenserne.

Poler og nulpunkters placering har intet med den absolutte frekvensværdi i z-planen at gøre. Disse er relative til samplefrekvensen.

Følgende grafer er henholdsvis fase- og amplitudekarakteristikken for et 4. ordens Butterworth-lavpas-filter:



***Figur 13**: Amplitudekarakteristik for et 4. ordens Butterworth-lavpasfilter [MATLAB].*



***Figur 14**: Fasekarakteristik for et 4. ordens Butterworth-lavpasfilter [MATLAB].*

Bemærk at Butterworth-filteret ikke har lineær fase.
Dette er typisk for IIR-filtre.

Disse grafer kan således betragtes som resultatet af en grafisk frekvensresponsanalyse, bestemt som beskrevet ud fra pol-nulpunktsdiagram.

I praksis er de, som filteret, fremstillet i MATLAB, ved hjælp af frekvensanalyse funktioner i Signal Processing Toolbox'en, vi har benyttet.

4.7 FIR-filtre

4.7.1 Fordele

FIR-filtre kan fremstilles med absolut lineær fasekarakteristik, hvilket gør dem egnede til anvendelse på områder, hvor der arbejdes med transiente signaler. De er, når de er ikke-rekursive, karakteriseret ved kun at have poler i origo og er derfor altid stabile. Endelig er FIR-filterets koefficientfølsomhed og internt frembragte afrundingsstøj lav.

4.7.2 Ulemper

Da FIR-filterets poler alle er i origo, har de ingen indflydelse på amplituderesponsen. Der kræves derfor et stort antal nulpunkter for at realisere en given amplituderespons, hvilket bevirker et relativt højt ordenstal, og dermed et stort antal forsinkelseselementer i systemet. Afskæringen af FIR-filterets impulsrespons, til et endeligt antal samples, har endvidere en uheldig virkning på amplituderesponsen, idet denne får påbygget pas- og stopbåndsripple. Sidstnævnte ulempe kan dog reduceres til et acceptabelt lavt niveau, hvis FIR-filterets impulsrespons modificeres med en såkaldt vinduesfunktion. Denne vil blive beskrevet senere.

4.7.3 Differensligning

FIR-filteret er beskrevet ved følgende differensligning

$$y[n] = \sum_{k=0}^{M} b_k \cdot x[n-k] \qquad (70)$$

Af udtrykket fremgår det, at FIR-filterets udgangssekvens består af den nuværende og M forsinkede samples, der alle vægtes med en koefficient b_k. Som beskrevet i kapitel 4.4.1 gælder det, at hvis x[n] er en enhedssample, $\delta[n]$, så er koefficientværdierne, b_k, lig impulsresponsen.

For at FIR-filteret skal have lineær fasekarakteristik, er det en betingelse at impulsresponsen er symmetrisk omkring midtpunktet [Hüche, s427]. Den skal altså tilfredsstille betingelsen $b_k = b_m$

$_{-k}$, for k = 0, 1, 2, 3,.....M.

4.7.4 Frekvensresponsanalyse

Ved konstruktion af FIR-filtre findes der ikke nogen formel metode, hvorved filterets nøjagtige ordenstal kan bestemmes på forhånd udfra den specificerede amplitudekarakteristik. Det kan derfor være hensigtsmæssigt at foretage en frekvensresponsanalyse af filteret, og herefter korrigere ordenstallet, hvis analysen viser, at filteret ikke overholder de opstillede krav til amplituderesponsen.

Da FIR-filterets impulsrespons er symmetrisk omkring midtpunktet, kan vi for at lette analyse arbejdet foretage en opsplitning af impulsresponssekvensen, idet vi af praktiske grunde begrænser os til at analysere systemer med et lige ordenstal

$$h[n] = \sum_{k=0}^{\frac{M}{2}-1} b_k \cdot \delta[n-k] + b_{\frac{M}{2}} \cdot \delta[n-\tfrac{M}{2}] + \sum_{k=\frac{M}{2}+1}^{M} b_k \cdot \delta[n-k] \qquad (71)$$

Da $b_k = b_{M-k}$, og på grund af symmetrien, kan h[n] også skrives som

$$h[n] = b_{\frac{M}{2}} \cdot \delta[n-\tfrac{M}{2}] + \sum_{k=0}^{\frac{M}{2}-1} b_k \cdot \left[\delta[n-k] + \delta[n-(M-k)]\right] \qquad (72)$$

FIR-filterets overføringsfunktion kan nu bestemmes som den z-transformerede af h[n]

$$H(z) = b_{\frac{M}{2}} \cdot z^{-\frac{M}{2}} + \sum_{k=0}^{\frac{M}{2}-1} b_k \cdot \left(z^{-k} + z^{-(M-k)}\right) = b_{\frac{M}{2}} \cdot z^{-\frac{M}{2}} + \sum_{k=0}^{\frac{M}{2}-1} b_k \cdot z^{-\frac{M}{2}} \cdot \left(z^{\frac{M}{2}-k} + z^{-(\frac{M}{2}-k)}\right) \qquad (73))$$

Frekvensresponsen kan nu ifølge (64) beregnes ved at erstatte z med $e^{j\omega T}$. Da det er et samplet system er frekvensresponsfunktionen periodisk med samplefrekvensen. Alle informationer om

amplitude- og faseforløb, er derfor tilstede i frekvensområdet $0 < f < \frac{f_s}{2}$, hvorfor vi kan nøjes med at analysere dette område. Vi frekvensnormerer i forhold til den halve samplefrekvens, og indfører en variabel, γ, der udtrykker den normerede frekvens, og er bestemt som $\gamma = f/\frac{f_s}{2}$. Vi får så følgende udtryk for frekvensresponsfunktionen

$$H(\gamma) = b_{\frac{M}{2}} e^{-j\frac{M}{2}\gamma\pi} + \sum_{k=0}^{\frac{M}{2}-1} e^{-j\frac{M}{2}\gamma\pi}\left(e^{j(\frac{M}{2}-K)\gamma\pi} + e^{-j(\frac{M}{2}-k)\gamma\pi} \right) \tag{74}$$

Da parentesen i summen er en cosinusfunktion med amplituden 2, kan $H(\gamma)$ også skrives som

$$H(\gamma) = e^{-j\frac{M}{2}\gamma\pi}\left[b_{\frac{M}{2}} + \sum_{k=0}^{\frac{M}{2}} 2b_k \cdot \cos\left((\tfrac{M}{2} - k)\gamma\pi\right) \right] \tag{75}$$

Amplitudefunktionen, der er den numeriske del af (75), beregnes i db som

$$\left|H(\gamma)\right| = 20 \cdot \log\left[b_{\frac{M}{2}} + \sum_{k=0}^{\frac{M}{2}} 2b_k \cdot \cos\left((\tfrac{M}{2} - k)\gamma\pi\right) \right] \tag{76}$$

Filterets lineære fasefunktion, der kan bestemmes udfra $e^{-j\frac{M}{2}\gamma\pi}$ som

$$\phi(\gamma) = -\tfrac{M}{2}\gamma\pi\,, \tag{77}$$

aftager lineært med γ.

Betragter vi fasen som en funktion af frekvensen

$$\phi(f) = -\frac{M}{2} \cdot \frac{2f}{f_s} \cdot \pi = -M\pi Tf\,, \tag{78}$$

kan vi udfra denne beregne FIR-filterets gruppeløbstid, der er fasefunktionens negative differentialkvotient

$$T_g = -\frac{d\phi(f)}{d\omega} = -\frac{1}{2\pi} \cdot \frac{d\phi(f)}{df} = -\frac{1}{2\pi} \cdot (-M\pi T) = \frac{M}{2} T \qquad (79)$$

Da frekvensen ikke indgår i udtrykket for gruppeløbstiden, kan vi konkludere, at der er tale om konstant gruppeløbstid. Det fremgår endvidere at tidsforsinkelsen er betragtelig for et FIR-filter af høj orden.

4.7.5 Konstruktion af et FIR-filter

I afsnit 4.1.2 blev det beskrevet, hvordan en periodisk funktion kan opløses i en række frekvenskomposanter. Ved at udføre den modsatte proces er det muligt, ved hjælp af den såkaldte Fourierkoefficientmetode, at beregne koefficienterne til FIR-filterets overføringsfunktion.

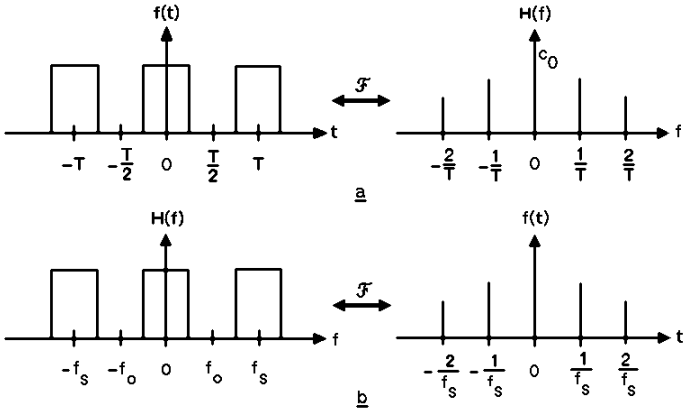
Ved beregningen af overføringsfunktionens koefficienter skal FIR-filterets amplitudefunktion $|H(f)|$, der er periodisk med samplefrekvensen, opløses i Fourierkomposanter i tidsdomænet. Disse koefficienter er et udtryk for FIR-filterets impulsresponssamples og dermed for overføringsfunktionens koefficienter.

Vi husker fra (11), at de komplekse Fourierkoefficienter var givet ved

$$c_m = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) \cdot e^{-jm\omega t}\, dt \qquad (80)$$

Ved at foretage en variabelombytning og samtidig erstatte perioden, T, med samplefrekvensen, f_s, kan vi opstille en tilsvarende ligning for Fourierkoefficienterne i tidsdomænet:

$$c_m = \frac{1}{f_s}\int_{-\frac{f_s}{2}}^{\frac{f_s}{2}} |H(f)| \cdot e^{-jmT2\pi f}\, df \qquad (81)$$



***Figur 15**: Fourieropløsningen af periodiske funktioner. a) Tidsfunktionen opløst i frekvens komposanter. b) Amplituderesponsfunktionen opløst i impulsresponskomposanter. [Hüche, s: 437]*

Dette udtryk kan ved hjælp af Eulers sætning skrives om til

$$c_m = \frac{1}{f_s}\int_{-\frac{f_s}{2}}^{\frac{f_s}{2}} |H(f)| \cdot (\cos(2\pi mTf) - j\sin(2\pi mTf))\, df \qquad (82)$$

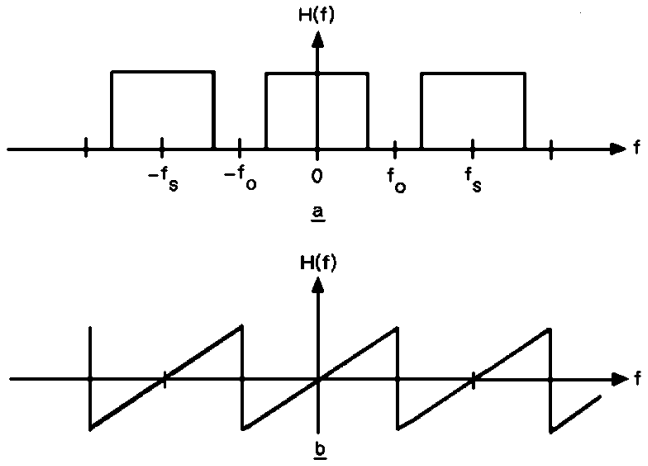
Amplitudefunktionen, kan efter ombytningen med f(t) skrives som den endelige Fourierrække

$$H(f) = \sum_{m=-\frac{M}{2}}^{\frac{M}{2}} c_m e^{j2\pi mTf} \qquad (83)$$

Da $|c_m| = |c_{-m}|$ er det tilsrækkeligt at beregne integralet fra $0 < f < \frac{f_s}{2}$ og herefter multiplicere

integralet med 2. Da det endvidere er en betingelse for at opnå lineær fasekarakteristik, at amplitudefunktionen ikke indeholder sinus- og cosinuskomposanter samtidigt, kan vi begrænse os til at betragte $|H(f)|$ som en lige amplitudefunktion. Amplitudefunktionen er lige, når Fourierrækken opløses i en cosinusrække, og ulige, når den opløses i en sinusrække. Se Figur **16**. Denne kan da omskrives til

$$c_m = \frac{2}{f_s}\int_0^{\frac{f_s}{2}} |H(f)| \cdot \cos(2\pi m T f)\, df \qquad (84)$$



***Figur 16**: Amplitudefunktionstyper. a) Lige amplitudefunktion. b) Ulige amplitudefunktion. [Hüche,s 439]*

Det svarer til, at Fourierkoefficienterne, og dermed overføringsfunktionens komposanter er relle.

Som nævnt er c_m et udtryk for overføringsfunktionens koefficienter. Den nøjagtige sammenhæng findes ved at opstille overføringsfunktionen, med de beregnede Fourierkoefficienter indsat, og herefter omskrive denne til en funktion af z.

$$z = e^{j\omega t} = e^{j2\pi ft} \qquad (85)$$

H(z) kan derfor skrives som

$$H(z) = H(f)\Big|_{e^{j2\pi ft}=z} = \sum_{m=-\frac{M}{2}}^{\frac{M}{2}} c_m z^m \tag{86}$$

For at modificere således, at der opnåes et kausalt filter, indføres en forsinkelse på $\frac{M}{2}$. Dette gøres ved at multiplicere udtrykket med $z^{-M/2}$: Derved fås

$$H(z) = \sum_{m=-\frac{M}{2}}^{\frac{M}{2}} c_m z^{m-\frac{M}{2}} \tag{87}$$

H(z) repræsenterer nu et kausalt system uanset værdien af m. For at nå frem til et simplere udtryk indføres variablen k, der defineres som

$$k = \tfrac{M}{2} - m \Leftrightarrow m = \tfrac{M}{2} - k \tag{88}$$

$$\text{Da} \quad \tfrac{M}{2} - k = -\tfrac{M}{2} \Rightarrow k = M \tag{89}$$

$$\text{og} \quad \tfrac{M}{2} - k = M \Rightarrow k = 0 \tag{90}$$

kan vi ved at indføre denne variabel i (87), og dernæst bytte grænserne om, skrive H(z) som

$$H(z) = \sum_{k=0}^{M} c_{\frac{M}{2}-k} \cdot z^{-k} \tag{91}$$

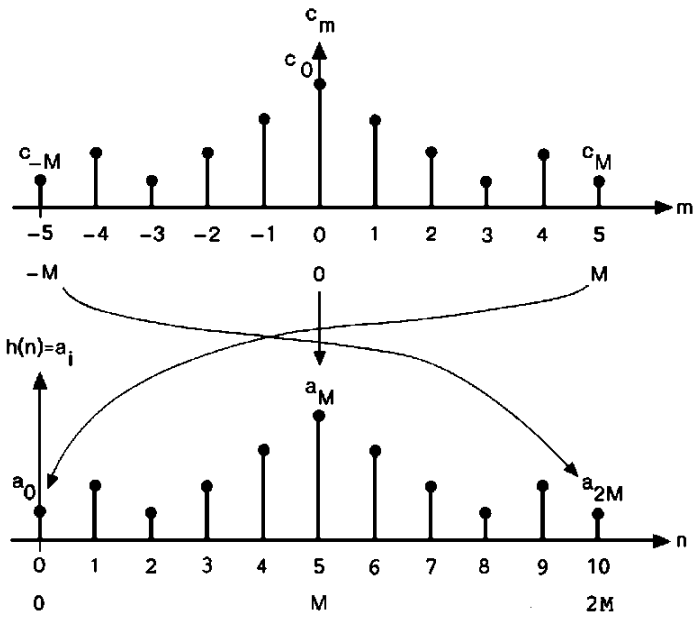
Vi indfører nu den generelle overføringsfunktions koefficienter, og definerer dem som

$$b_k = c_{\frac{M}{2}-k} \tag{92}$$

Indføres denne i (91) får vi et udtryk for den kausale FIR overføringsfunktion:

$$H(z) = \sum_{k=0}^{M} b_k \cdot z^{-k} \qquad (93)$$

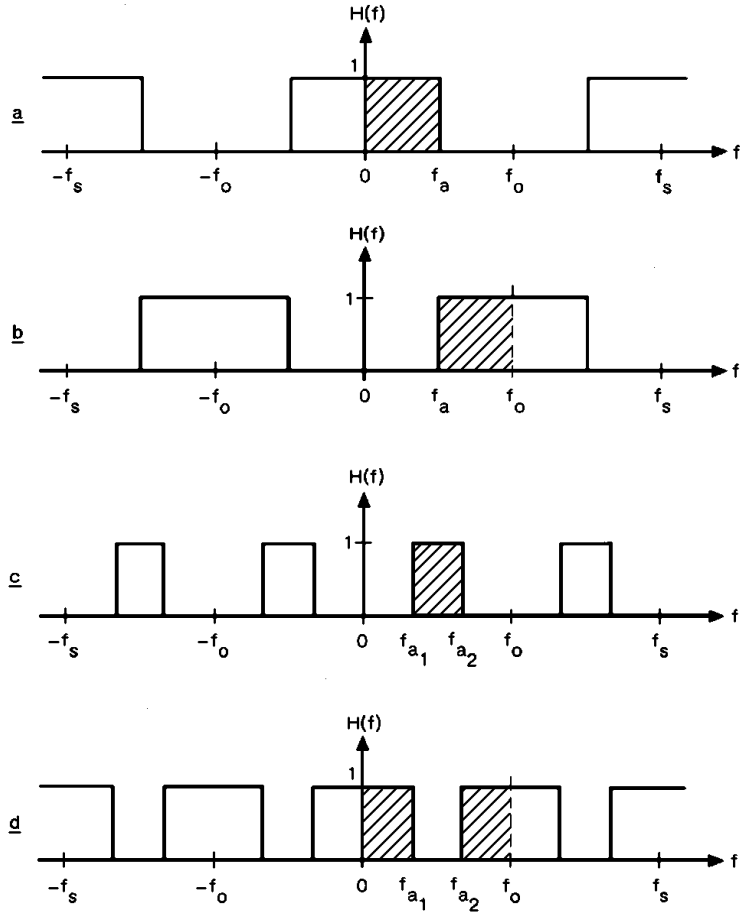
Sammenhængen mellem det realiserbare FIR-filters koefficienter og de beregnede ikke-kausale Fourierkoefficienter er angivet i Figur **17**. Koefficienterne er repræsenteret ved de tilsvarende impulsresponssamples.



***Figur 17**: Sammenhæng mellem Fourierkoefficienterne c_m og FIR overføringsfunktionens a_i koefficienter. [Hüche, s 443]*

4.7.6 Beregning af koefficenter

Beregningen af FIR-koefficienterne kan altså udføres, når man har en defineret og integrerbar amplitudefunktion. Når det drejer sig om standard filterfunktioner som lavpas-, højpas-, båndpas- og båndstopfunktioner, anvendes de ideelle amplitudefunktioner, der er karakteriseret ved, at forstærkningen er 1 i pasbåndet og 0 i stopbåndsområdet.



***Figur 18**: Ideelle FIR-filtre amplitudekarakteristik. a) Lavpas. b) Højpas. c) Båndpas. d) Båndstop.[Hüche, s 445]*

Vi får så følgende udtryk for standard filterfunktionerne, hvor f_a er afskæringsfrekvensen.

Lavpas:

$$c_m = \frac{2}{f_s}\int_0^{f_a} \cos(2\pi mTf)\,df \qquad (94)$$

Højpas:

$$c_m = \frac{2}{f_s}\int_{f_a}^{\frac{f_s}{2}-a} \cos(2\pi mTf)\,df \tag{95}$$

Båndpas:

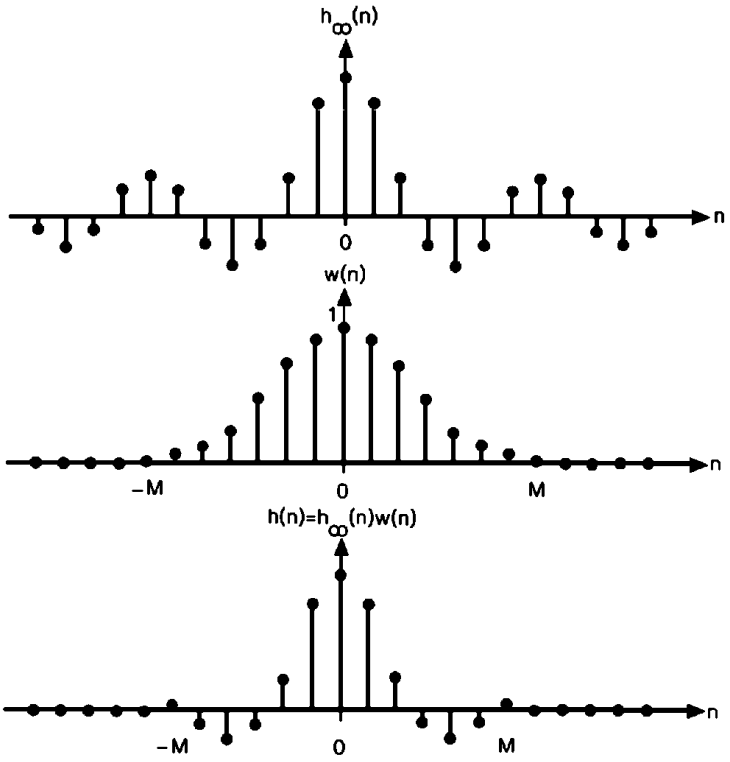
$$c_m = \frac{2}{f_s}\int_{f_a1}^{f_a2} \cos(2\pi mTf)\,df \tag{96}$$

Båndstop:

$$c_m = \frac{2}{f_s}\int_{0}^{f_a1} \cos(2\pi mTf)\,df + \frac{2}{f_s}\int_{f_a2}^{\frac{f_s}{2}} \cos(2\pi mTf)\,df \tag{97}$$

4.7.7 Vinduesfunktioner

Ved afskæringen af den uendelige impulsrespons viser det sig, som tidligere nævnt, at der fremkommer en uacceptabelt stor pas- og stopbåndsripple. Denne kan dog reduceres til et acceptabelt lavt niveau, hvis impulsresponsen vægtes med en såkaldt vinduesfunktion. Vinduesfunktionen afskærer impulsresponsen på en sådan måde, at h[n] vokser og aftager som en jævn funktion af tiden. Se Figur **19**.



***Figur 19**: Afskæring af en impulsrespons ved hjælp af en vinduesfunktion. [Hüche, s 458]*

Dette medfører, at den ideelle amplitudekarateristik ikke kan realiseres, da overgangsområdet mellem pas- og stopbånd får en vis udstrækning i tid. Det realiserede FIR-filter bliver derved et kompromis, hvor der både må tages hensyn til behovet for at reducere de uønskede ripples, og behovet for at filteret skal være så stejlt som muligt. Et eksempel på en sådan vinduesfunktion er Hanningvinduet, hvor de enkelte impulsresponssamples vægtes med en række koefficienter der beregnes således

$$w_m = w_{-m} = 0.5 + 0.5 \cdot \cos\left(\frac{2\pi m}{M}\right) \qquad (98)$$

4.7.8 MATLAB

I praksis foregår digital filterdesign som regel ved hjælp af softwareværktøjer, der tillader en sofistikeret kontrol af filterets karakteristik. I MATLAB findes der flere af sådanne

designprogrammer. Et eksempel på et sådant program er FIR1. Dette program tillader os at specificere et ønsket frekvensområde for eksempelvis et pasbånd. Computeren beregner da en god tilnærmelse til det ideelle filter, f.eks. ved hjælp af hanningvinduesfunktionen.

5. Digital Mixer

5.1 Specifikationer

For at vise fordele/ulemper ved DSP, vil vi i det følgende konstruere en digital lydmixer. Som DSP-processor har vi valgt at bruge en almindelig PC, da den i teorien fungerer som en dedikeret DSP-kreds. Vi har foretaget dette valg, da vi ikke har et gennemgående kendskab til elektronik og dermed ikke viden nok til at kunne konstruere en digital mixer med en DSP-kreds.

Vi har opstillet følgende specifikationer for mixeren, som vi ønsker at fremstille:

2 kanals mixer

Da mixeren kun tjener til demonstration af DSP, vælger vi at lave en to-kanals mixer, idet det er de samme principper, der gør sig gældende ved to kanaler som ved 16 kanaler.
Rent praktisk bliver mixeren i stand til i realtidssammenhæng at mixe to 16-bit mono signaler fra et soundblaster16 (eller kompatibelt) lydkort.
Det mixede signal sendes til lydkortet igen, hvor det afspilles som et 8 bit mono signal. Grunden til, at udgangssignalet kun er i 8 bit, er at lydkortet ikke tillader højre opløsning, når det skal indspille og afspille på samme tid (full duplex). Dog kunne vi vælge indgangssignalet til at være 8 bit og udgangssignalet til at være 16 bit.

Vi har valgt at udstyre mixeren med de mest almindelige funktioner fra en analog mixer. Digitale mixere indeholder meget ofte en række ekstra effekter. Derfor har vi valgt at tilføje en effekt, for at demonstrere, hvor simpelt det er at tilføje effekter, samt hvor lidt det kræver af hardwaren. Hver kanal i mixeren kommer til at indeholde følgende features:

Gain ± 15dB:

Gain, der står for forstærkningen af indgangssignalet, bruges til at tilpasse indgangssignalerne således, at man opnår samme niveau i de 2 kanaler. Ved at vælge en gain på ± 15dB er det muligt at opnå samme niveau selv ved stor forskel mellem indgangssignalerne.

3-bånds equaliser (tonekontrol) ±15dB:

En equalizer er en af de mest almindelige funktioner i en mixer. Equalizeren bruges til at "farve lydbilledet" ved at give mulighed for at dæmpe/hæve forskellige frekvensbånd i lyden. Vi har valgt at lave en 3 bånds equalizer med følgende delefrekvenser:

- Bas: 0-f_s/100
- Mellemtone: f_s/200-f_s/20
- Diskant: f_s/20-f_s/2

Til equaliseringen skal benyttes filtre med en orden, der er så stor som muligt, men som samtidigt er realiserbare på en almindelig PC (166 MHz Pentium)

Noisegate:

Gaten er den effekt, vi har valgt at implementere i mixeren som en ekstraeffekt. Grunden til dette valg er, at gaten er forholdsvis simpel at realisere digitalt, og at den er en af de effekter, som tit inkluderes i digitale mixere.

Selve ideen i en gate er ret simpel. Når indgangssignalet er under et vist forudbestemt niveau, "threshold", sættes udgangen til nul.

Dog er der et lille problem med denne enkle metode, idet man risikere at skære nogle lave signaler væk, der ikke skulle fjernes. Derfor indføres en "hold time". Det er et fastsat tidsrum, i hvilket indgangssignalet skal forblive under thresholdværdien, før udgangen bliver sat til nul. Når indgangssignalet igen kommer over thresholdværdien åbnes udgangen, og indgangssignalet passerer uhindret gaten..

Desværre er der endnu et par problemer med gaten, der skal afklares, inden den bliver brugbar. For som det ser ud nu vil udgangssignalet falde abrupt til nul, når indgangssignalet har været lavere en "threshold" niveauet i "hold time" perioden. Det resultere i at udklingende lyde vil blive pludseligt afbrudt. For at kompensere for dette, indføres en fader, der i stedet for at lade udgangen gå abrupt mod nul, bruger en såkaldt "release time" til at fade udgangssignalet ned i.

Det samme gøres også når indgangssignalet igen overstiger "thresholdniveauet" og udgangen åbnes. På den måde skrues der op for udgangen, så man undgår en pludselig åbning. Den tid udgangen er om at skrue op kaldes "attack", og er sammenlignet med "release time" meget lille.

Den sidste feauture, der med fordel kan indgå i gaten er en "hysterisis", med en "hysterisis" definerer man en forskel mellem den "thresholdværdi" hvor gaten lukker og åbner udgangen. "Thresholdværdien" for at gaten åbner skal altså være større end værdien, for at den lukker. På den måde undgår man at gaten kommer til at stå og åbne og lukke, hvis størrelsen af indgangssignalet ligger lige omkring "thresholdværdien".

Volume:

Denne effekt bruges til at justere lydstyrken af signalet for hver af kanalerne. Vi har valgt, at lade vores volume lave en forstærkning fra $-\infty$ til 10dB, hvilket er typisk grænse i mixere.

Cross-fader:

Cross-faderen er en volumekontrol, der virker mellem to kanaler, dvs. at man kan mixe imellem to kanaler, med én fader, i stedet for med to, hvor der henholdsvis skrues op for den ene, og ned for den anden. Dette er meget praktisk, hvis brugerfladen ikke tillader ændring af flere indstillinger samtidig.

Main-volume:

Med denne indstilling kan den samlede lydstyrke for det mixede signal reguleres. Vi har valgt en typisk forstærkning med grænserne $-\infty$ til 10dB.

Clip-indicator:

Efter nedmixningen af de to kanaler til én kanal skal signalet sendes ud til lydkortet som en 8-bit datastrøm. Da den samlede forstærkning i mixeren maximalt kan blive (15+15+10+10)dB=50dB, kan man nemt komme til at overskride denne grænse. Hvis grænsen overskrides, klippes signalet. For at gøre brugeren opmærksom på dette, indføres clip-indikatoren, som er en "lampe" i brugerfladen.

5.2 Konstruktion af filtrene

Filter-koefficienterne til de 3 bånd i equalizeren kan beregnes ud fra den gennemgåede teori, men dette er en besværlig og ikke mindst langsommelig proces, derfor har vi benyttet Signal Processing toolbox'en (ver. 2.0b) til MATLAB. MATLAB har desuden en række udmærkede analysefaciliteter.

Filtrene fremstilles hver for sig og implementeres parallelt. Pasbåndet kan så forstærkes eller dæmpes ved multiplikation med en faktor.

Ved konstruktion af filtrene skal der tages stilling til følgende:

- Filterteknik
- Orden
- Inddeling (Afskæringsfrekvenser)
- Filterfunktion/Vinduesfunktion

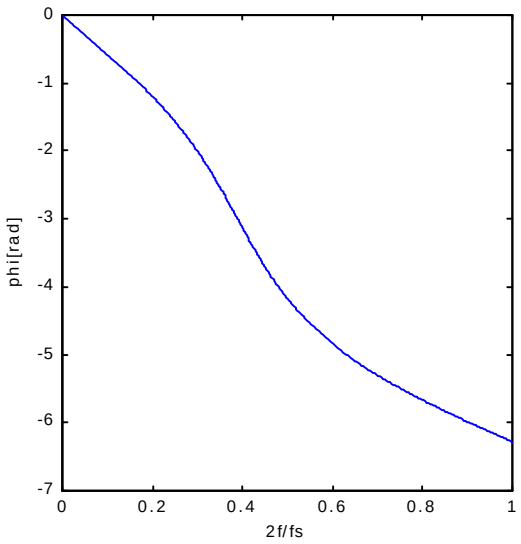
Disse er også valgene, der skal foretages i forbindelse med konstruktion af filtre i MATLAB.

5.2.1 Filterteknik

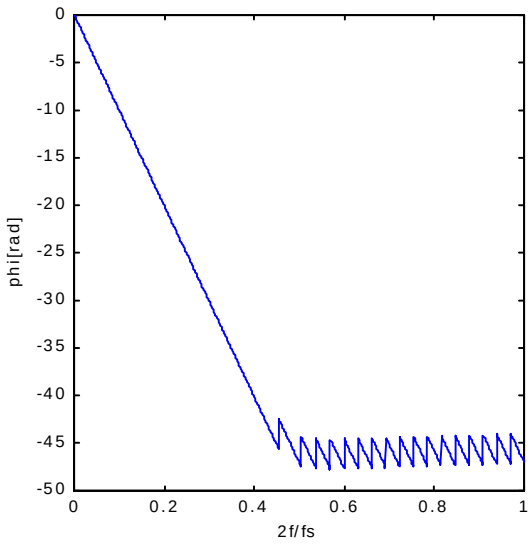
Hovedargumentet, for at anvende FIR-filterteknik fremfor IIR-, er, at FIR-filtre har lineær fasekarakteristik i pasbåndet, dvs. konstant gruppeløbstid. Den transiente natur af Mikserens inputs gør FIR-filtre velegnede til equalizeringen [Hüche, s. 422]. Gruppeløbstiden kan aflæses som hældningen på kurven med modsat fortegn.

Figur 20 og Figur 21 viser forskellen i fasekarakteristikken mellem IIR og FIR:

FIR-filtre er desuden langt mindre koefficientfølsomme - Dette skulle dog ikke betyde noget, da både beregningerne i MATLAB og i vores program foretages i 64 bit Floating Point (flydende komma), dvs. double precision.



Figur 20: *Fasekarakteristik af 4. ordens IIR-filter (Butterworth) med afkæring ved 0,4 af den halve samplefrekvens.*



Figur 21: *Fasekarakteristik af 64. ordens FIR-filter (Hamming-vindue) med afskæring ved 0,4 af den halve samplefrekrevens.*

IIR-filtre kan til gengæld give meget stejlere flanker, dvs. stor selektivitet, ved lavere orden.

5.2.2 Orden

Ordenen af filtrene, vi kan realisere, afhænger af den processorkraft, vi har til rådighed. Følgende, grove, udtryk kan stilles op for antallet af Floating Point instruktioner, der skal udføres per sekund, dvs. FLOPS (FLoating Point Operations Per Second):

$I = (N+1) \cdot f_s \cdot C \cdot 2$, hvor N er filterordenen, f_s samplefrekvensen og C antallet af kanaler.

Normalt ville tre filtre af N'te orden implementeres efter følgende princip:

$$Y[n] = (X[n] * H_1) + (X[n] * H_2[n]) + (X[n] * H_3[n]) \qquad (99)$$

Det vil sige som tre uafhængige foldninger. Af følgende ses, at foldningerne kan udføres i en foldning:

$$\begin{aligned} Y[n] &= \sum_{k=0}^{N} X[n-k]\cdot H_1[k] + \sum_{k=0}^{N} X[n-k]\cdot H_2[k] + \sum_{k=0}^{N} X[n-k]\cdot H_3[k] \\ &= \sum_{k=0}^{N} X[n-k]\cdot \left(H_1[k] + H_2[k] + H_3[k]\right) \end{aligned} \qquad (100)$$

Altså: $Y[n] = X[n] * q[n]$, hvor $q[n] = H_1[n] + H_2[n] + H_3[n]$ (101)

Princippet er altså det, at de tre bånd i equalizeren kan implementeres som ét filter. Dette filter skal så beregnes, hver gang forstærkningen af et af båndene ændres.

Forsøg, vi har udført, har vist, at en Pentium 166 MHz, som vi fremstiller mixeren til, kan afvikle programmet ordentligt med en filterorden på 100.

En 3-bånds EQ i en 2-kanals mixer bestående af 100-ordens filtre vil kræve en processor, der kan udføre: $(100+1)\cdot 44100\cdot 2\cdot 2 = 17{,}8$ millioner FPU-instruktioner per sekund (MFLOPS).

5.2.3 Inddeling af båndene

Den ideelle equalizer, består af en række af lige brede bånd, målt i oktaver, dvs. på en logaritmisk skala, der hver reguleres amplitudemæssigt ved multiplikation med en forstærkningsfaktor. Da det her drejer sig om tre bånd, i området fra 20 Hz til 20 kHz, vil den ideelle inddeling af filtrene være dekadisk, hvor hvert bånd er 5 oktaver bredt. Opdelt i frekvensintervaller giver det:

20 - 200 Hz
200 - 2000 Hz
2000 - 20000 Hz

Vi har realiseret 3-bånds equalizerens i form af et lav-, et bånd og et højpasfilter.

Da den benyttede samplefrekvens er 44100 Hz, vil det være en fordel at ændre

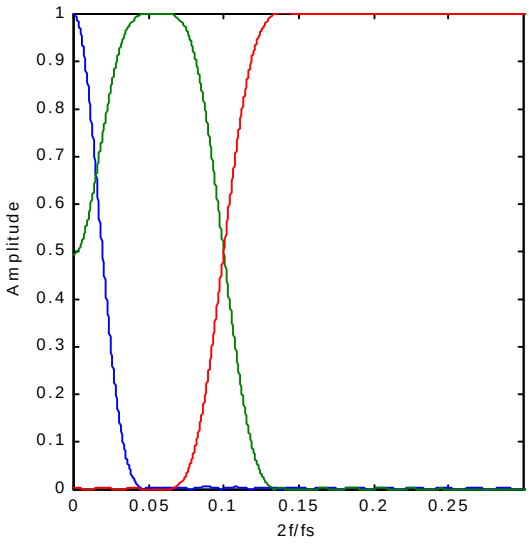
frekvensintervallerne, således at disse kan beskrives ved "lige" brøker af samplefrekvensen

$$\frac{f_s}{2000} - \frac{f_s}{200} \approx 22 - 221\,\text{Hz}$$

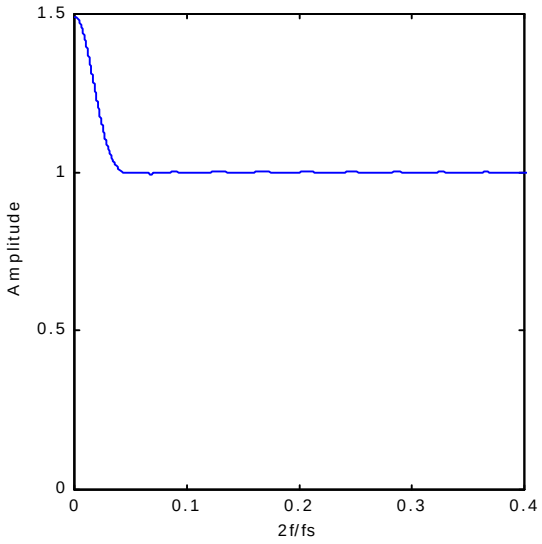
$$\frac{f_s}{200} - \frac{f_s}{20} \approx 221 - 2205\,\text{Hz}$$

$$\frac{f_s}{20} - \frac{f_s}{2} \approx 2205 - 22050\,\text{Hz}$$

Denne inddeling giver dog problemer i forbindelse beregningen af filterkoefficienter i MATLAB, som Figur 22 og Figur 23 viser (Her FIR-filtre konstrueret ud fra Hamming-vinduet):



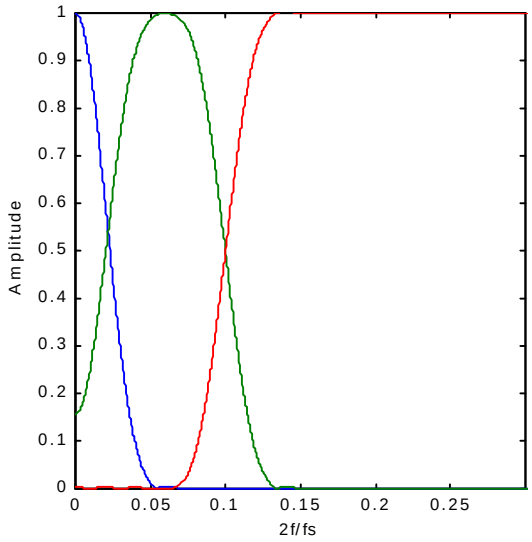
Figur 22: *Amplitudekarakteristik af de 3 bånd hver for sig. Bemærk skærigen ved 0,5 af pasbåndamplituden mellem båndpas-filteret og højpas-. Mellem lav- og båndpasfiltreret sker skæringen ved 0,65.*



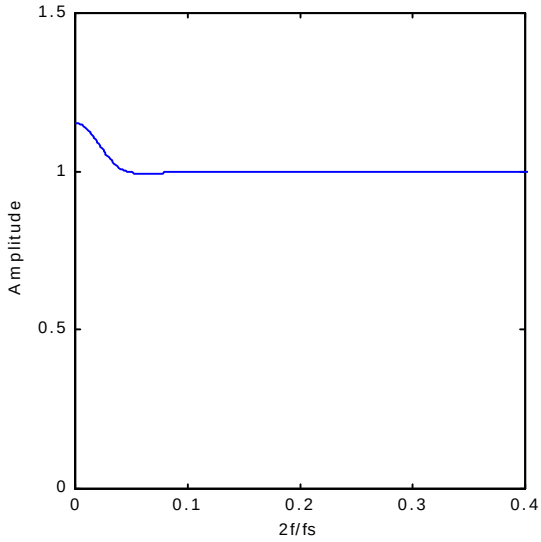
Figur 23: *Amplitudenkarakteristikken (ikke i dB) af de 3 bånd (summeret). Ideelt skulle ampltiduden være 1 i hele pasbåndet.*

Dette skyldes de lave afskæringsfrekvenser i forhold til den halve samplefrekvens. Dette kan løses ved at nedsample signalet, hvilket vil gøre f.eks. en afskæring ved 221 Hz til en større brøkdel af den halve samplefrekvens. Dette vil dog medføre komplikationer i programmet, og

vi har følgelig valgt at flytte afskæringsfrekvensen for skæringen mellem lav- og båndpasfilteret fra 0,01 til 0,02.



Figur 24: *Amplitudekarakteristik af de 3 bånd hver for sig, med skæring ved 0,02 og 0,1 af den halve samplefrekvens. Sammenlign med* ***Figur 22****.*

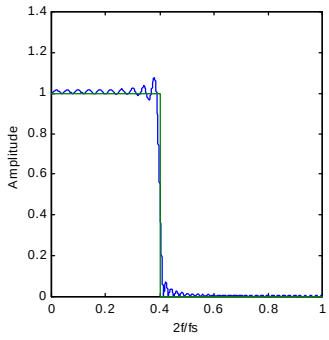


Figur 25: *Amplitudkarakteristik af de 3 bånd (summeret), med skæring ved 0,02 og 0,1 af den halve samplefrekvens. Sammenlign med* ***Figur 23***

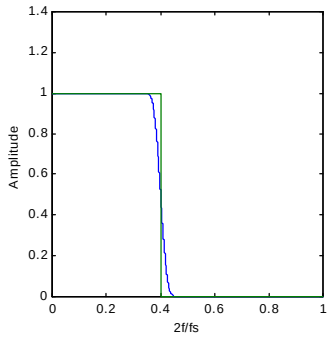
Amplitudekarakteristikken af filtrene er her mere i retning af det ideelle, uden at lavpasfiltret, dvs. bassen, kommer utilfredsstillende langt op i mellemtone-området.

5.2.4 Vinduesfunktion

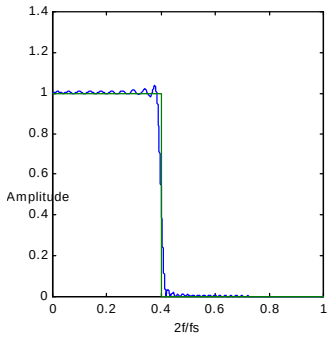
Følgende grafer viser amplitudekarakteristikker af lavpasfiltre genereret ud fra de vinduesfunktioner MATLAB tilbyder (Boxcar er udeladt) set i forhold til det ideelle filter. Filterne er relative til den halve samplefrekvens, og har afskæring ved 0,4.



Figur 26: *Chebwin*

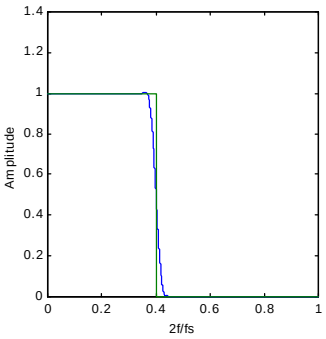


Figur 27: *Blackman*

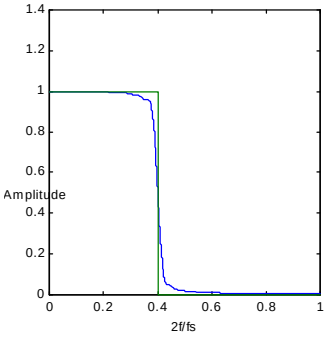


Figur 28: *Kaiser-*

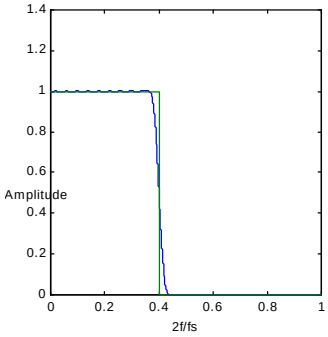
Kaiser- og Chebychev-vindusfunktionerne giver relativt megen ripple i pas- og stopbåndet. Blackman-vinduesfunktionen har den ulempe, at amplituden skærer meget blødt ved afskæringsfrekvensen. Dette er en ulempe i forbindelse med filtre med lav pasbåndsbredde.



***Figur 29**: Hanning*



***Figur 30:** Bartlett*



***Figur 31:** Hamming*

Lavpasfiltret, der er genereret ud fra Bartlettvinduesfunktionen, skærer tidligt sammenlignet med de andre (se arealet mellem kurverne) og giver en relativt dårlig dæmpning i stopbåndet [Hüche, s. 475]. Hanning-vinduesfunktionen forårsager en lille pukkel ved afskæringsfrekvensen og stopbåndsfrekvensen, hvorimod Hamming giver en smule ripple i pasbåndet - Denne er dog uforholdsmæssig stor på grafen - den er langt mindre end den ene pukkel ved Hanning.

Generelt gælder for disse vinduesfunktioner (bortset fra Bartlett), at den minimale dæmpning i stopbåndet er så stor (under 45 dB), at det ikke vil have nogen betydning i programmet, da output er 8 bit.

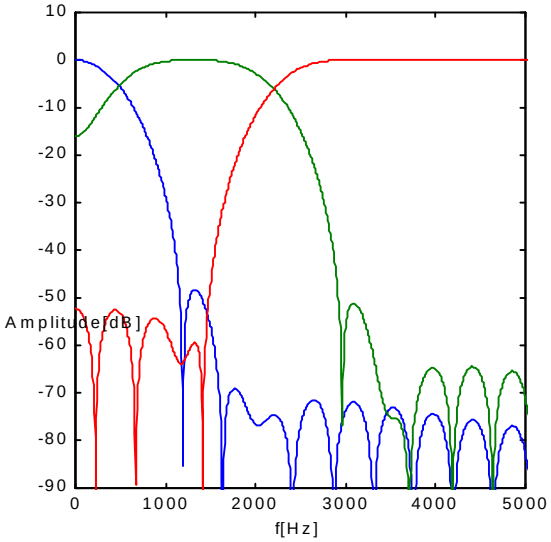
Alle har i deres egenskab af FIR-filtre lineær fasekarakteristik i pasbåndet. Filtrene er også alle 0,5 af pasbåndsamplituden ved afskæringsfrekvensen.

5.2.5 De tre bånd

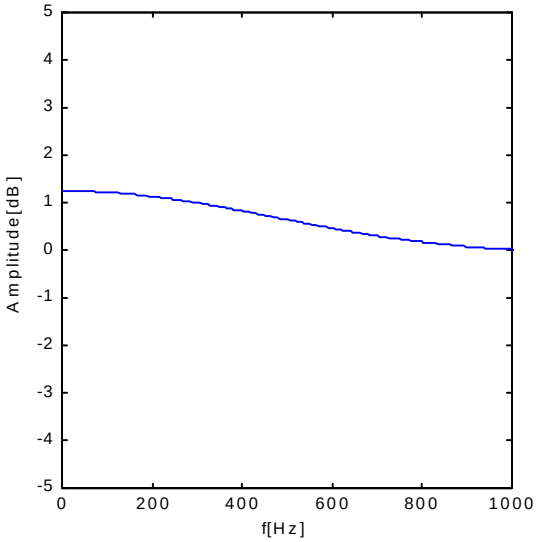
Alle disse overvejelser har foranlediget, at vi realiserer 3-bånds equalizeren i form af 3 FIR-

filtre: et lav-, bånd- og et højpasfilter. Disse har følgende karakteristika: Hamming-vinduesfunktionen anvendes til konstruktionen. Afskæringsfrekvenserne mellem de tre filtre, vi har besluttet af anvende, er 0,02 og 0,1 i forhold til den halvesamplefrekvens, dvs. ca. 440 Hz og 2205 Hz.

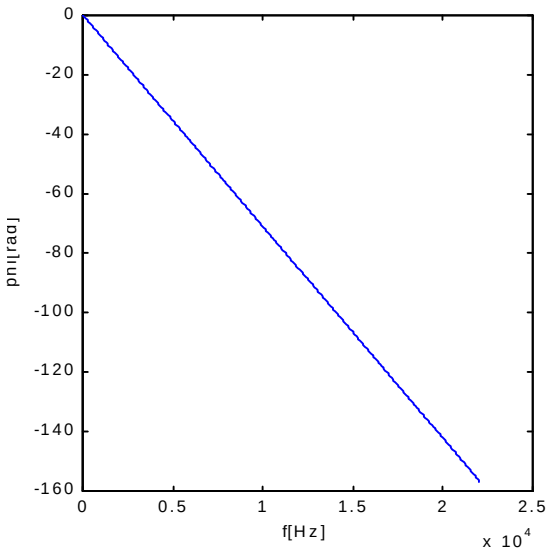
Følgende grafer viser den endelige 3-bånds equalizer amplitude- og fasekarakteristikker:



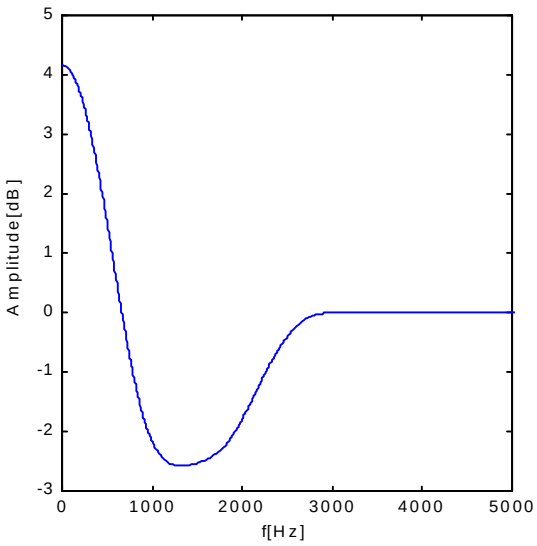
Figur 32: *Amplitudekarakteristik af de 3-bånd hver for sig.*



Figur 33: *Amplitudekarakteristik af de 3-bånd (summeret) ved det lave (kritiske) frekvensområde.*



Figur 34: *Fasekarakteristik af de 3-bånd (summeret).*



Figur 35: *Amplitudekarakteristik for det samlede filter, hvor bassen er forstærket 3,5 dB, og mellemtoneområdet er dæmpet -2,5 dB.*

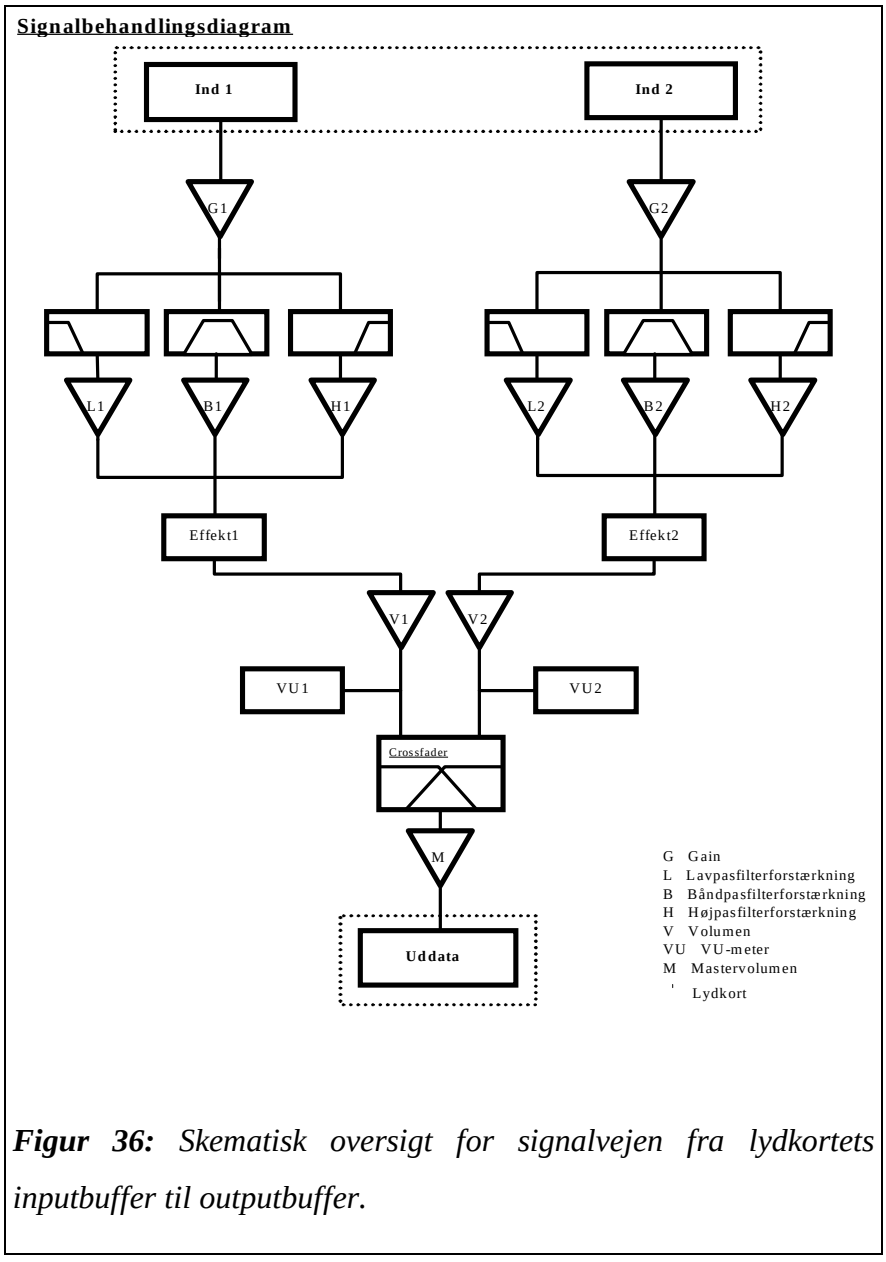
5.3 Programdokumentation

5.3.1 Analyse

For at kunne afvikle programmet MIXER347 kræves et Sound Blaster 16 eller 100% kompatibelt lydkort. Dette lydkort er valgt, fordi det giver mulighed for full duplex operation, dvs. man er i stand til både at afspille og optage lyd på samme tid. Desuden er Sound Blaster 16 lydkortet meget udbredt og står for en standard, de fleste andre lydkort på markedet er kompatible med.

Dele af programmet er ikke dokumenteret, da de ikke virker til at fremme essensen af vores projekt. Disse dele omfatter brugerfladen og hardwaredelen, hvor brugerflade dækker over: GUI-programmeringen (Graphical User Interface), og hardware over: Sound Blaster, DMA- (Direct Memory Access) og Interrupts-programmeringen.

I det følgende bliver de enkelte trin i signalvejen mellem lydkortets input og output analyseret. De forskellige trin udføres separat på hver af de to kanaler, indtil kanalerne mixes sammen.



Figur 36: *Skematisk oversigt for signalvejen fra lydkortets inputbuffer til outputbuffer.*

Gain

Med gain forstås forstærkning af indgangssignalet inden signalbehandlingen. Dette gøres ved at gange signalets samples med en forstærkningsfaktor. I kravspecifikationen blev gain bestemt til en forstærkning på ± 15 dB. Amplitude i decibel beregnes ved formelen:

$$A[\mathrm{dB}] = 20 \log B$$

hvor B er forstærkningen, givet ved forholdet mellem udgangs- og indgangssignalet, dvs.

$B = \frac{ud}{ind}$, og A er dette forhold i dB.

Omregnet til en forstærkningsfaktor giver det:

$$\pm 15\text{dB} = 10^{\left(\frac{\pm 15}{20}\right)} = \begin{cases} 5{,}62 \\ 0{,}178 \end{cases}$$

Disse værdier angiver forstærkningsfaktorens grænser i programmet.

Input: Forstærkningsfaktor fra brugerfladen, lydbuffer

Output: Lydbuffer

EQ

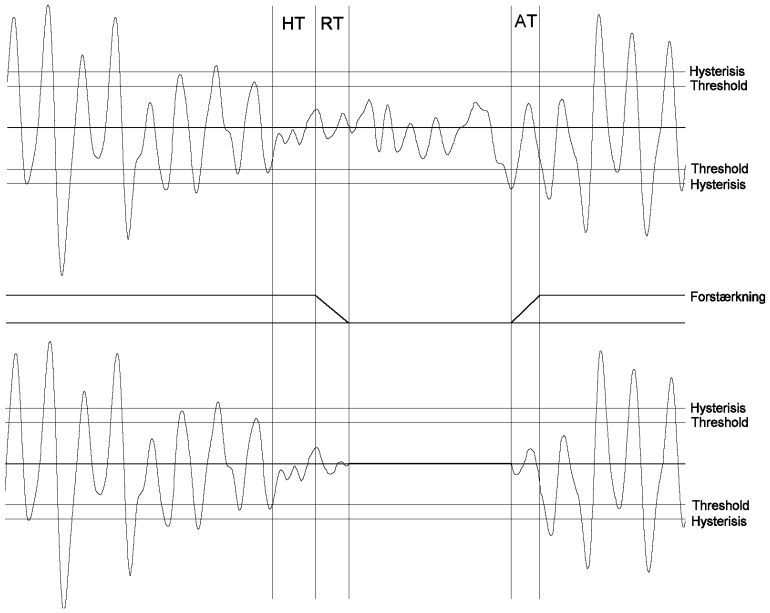
EQ'en udfører equaliseringen på signalet. Equalizeringen realiseres ved 3 FIR filtre (Se afsnit 5.2). De tre filtre består af et lav-, bånd- og højpasfilter. Filtrenes koefficienter (impulsrespons) hentes fra en fil, lavet i matlab. Filtreringen udføres ved at folde impulsresponsen med signalet. De tre filtre multipliceres hver for sig med en forstærkningsfaktor på $\pm 15\text{dB}$, hvilket omregnet svarer til forstærkningsgrænserne fra 0,178 til 5,62. Forstærkningsfaktoren bestemmes via brugerfladen. Herefter skal outputtet fra de 3 filtre mixes.

Input: 2x3 amplitudekoefficienter fra brugerflade, filterkoefficienter fra fil, lydbuffer.

Output: Ny lydbuffer.

Noisegate

I en gate "lukkes" signalet, når dets amplitude kommer under en bestemt grænse ("threshold"). Når signalet kommer op over en bestemt grænse ("threshold"+"hysterisis"), "åbnes" der igen for signalet. Den tid, der går inden gaten begynder at lukke, kaldes "hold time". Tiden det herefter tager at lukke gaten, bestemmes af "release-time", mens den tid det tager at åbne igen kaldes "attack-time". Se **Figur 37**



Figur 37: *Gatens virkemåde. Øverst signal før gaten. Midterst forstærkningen i gaten. Nederst signal efter gaten. HT="hold-time", RT="release-time", AT="attack-time".*

Gatens "threshold" indstilles via brugerfladen og skal kunne dække hele amplitudeområdet. Den maksimale amplitude før gain er, pga. 16-bits input fra lydkortet, 32768, og efter 15 dB gain $5.62 \cdot 32768 = 184268$, hvilket svarer til den maksimale "thresholdværdi". "Hysterisis" skal ligeledes kunne indstilles via brugerfladen af brugeren, og vi har valgt at lade den gå fra 0 til 32767.

Gatens "hold time" skal kunne varieres således, at tiden der går inden gaten begynder at fade ud, højest er 2000 ms. Dette omregnes til antal samples, hvilket ved en 44.100 Hz samplingsfrekvens svarer til 88.200 samples. Den korteste tid bestemmes således, at gaten fungerer uforstyrret inden for det hørbare frekvensområde fra 20 Hz til 20.000 Hz. Da et 20 Hz signal har den længste periode $T = \frac{1}{20\,\mathrm{Hz}} = 50\,\mathrm{ms}$, skal den korteste tid, der går, inden gaten begynder at fade ud, være større end 50 ms, hvilket svarer til 2.205 samples ved en samplingsfrekvens på 44.100 Hz.

"Release time" omregnes i brugerfladen til en "release-hastighed", som angiver, hvor hurtigt gatens forstærkningsfaktor falder pr. sample. Gatens forstærkningsfaktor kan antage værdier

mellem nul og én. “Release-time” defineres til at gå fra 0 til 2000 ms. og dermed går “release-hatigheden” fra ∞ til $\frac{1}{2000}$ “Attacktime” skal ligesom “release time” værdierne defineres til at gå fra 0 til 2000 ms. og omregnet til “attack-hastighed” fra ∞ til $\frac{1}{2000}$. Den “uendelige” hastighed realiseres ved at sætte hastigheden til 1 i brugerfladen.

Input: Værdier for “threshold”, “hysterisis”, “holdtime”, “releasespeed” og “attackspeed” fra brugerfladen, lydbuffer.
Output: Lydbuffer

Volumekontrol

Volumekontrolen benyttes til at forstærke lydsignalet, efter det har været igennem de forskellige effekter. Volumeforstærkning sker ved, at man ganger en, fra brugerfladen given, værdi på lydsignalet. Forstærkningen skal ifølge kravsspecifikationen gå fra $-\infty \text{dB}$ til 10dB.
Dette svaret til forstærkningsgrænser fra 0 til 3,16 dB

Input: Lydbuffer, lydstyrke fra brugerfladen
Output: Lydbuffer

VU-meter

Et VU-meter angiver grafisk lydsignalets amplitude. Amplituden bestemmes som den maksimale værdi i lydbufferen. Herefter skal VU-meterets indikator opdateres med den fundne værdi.

Input: Lydbuffer.
Output: Opdatering af VU-indikator i brugerfladen.

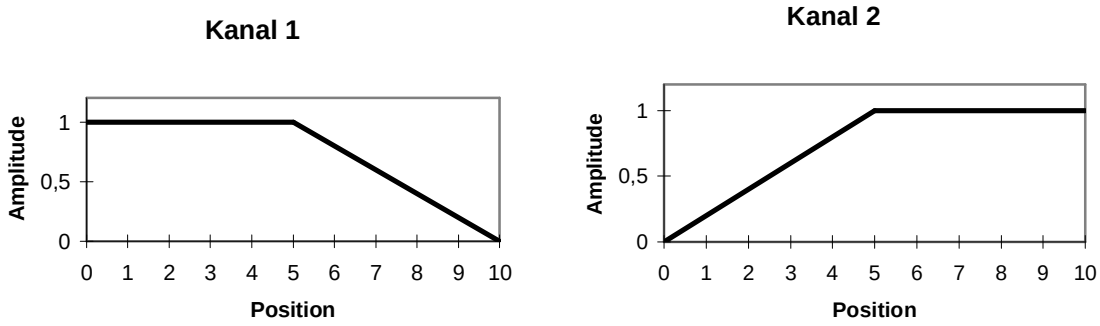
Cross-fader

Cross-faderen er en fader, der virker mellem to kanaler. På den måde kan man mixe kanalerne med én fader. Lidt specielt gælder det for cross-faderen, at den går fra at have gange nul forstærkning i den ene yderposition, til at have gange én forstærkning i både midterposition og den anden yderposition. Se illustration af princippet i **Figur 38**. Det er selvfølgelig omvendt for

den anden kanal. Crossfaderen kan fra brugerfladen få værdier mellem 0 og 2.

Input: Lydbuffere, crossfaderposition fra brugerflade

Output: Lydbuffere



***Figur 38**: Amplitudekarakteristikker for begge kanaler i cross-fader'en*

Master-volume

Den fælles volume skal virke på det samlede signal med en forstærkning på $-\infty \text{dB}$ til 10dB.

Dette svaret til forstærkningsgrænser fra 0 til 3,16 dB

Input: Lydbuffer, lydstyrke fra brugerfladen

Output: Lydbuffer

Mixning

I en mixer mixes signalerne fra flere kanaler til et samlet signal. Dette gøres i teorien ved at addere signalerne efter superpositionsprincippet. Hvis signalerne interfererer konstruktivt vil det ved to kanaler i værste tilfælde resultere i en amplitudefordobling af den mixede udgangsamplitude i forhold til indgangsamplituden. Derfor skal signalets styrke halveres, inden det sendes ud til lydkortet. Da Sound Blaster 16 lydkortet som før nævnt ikke samtidigt kan optage og afspille i 16 bit kvalitet, er afspilningen begrænset til 8 bit. Dette betyder, at de 8 mindste bit skal fjernes, hvilket svarer til at dividere signalets styrke med $2^8 = 256$. Mixningen foretages altså ved at addere de to signaler, og dividere signalstyrken med $2 \cdot 2^8 = 512$.

Skulle det alligevel ske, at signalstyrken overstiger lydkortets 8-bits grænse, er man nødt til at

"klippe" signalet. Ved 8-bit med fortegn er amplitudegrænserne i lydkortet -128 og 127. Ved "klipning" skal alle samples, der har en værdi større end 127, få værdien 127 og alle samples med en værdi mindre end -128, få værdien -128. Ved en sådan klipning vil lydsignalet få tilført uønsket støj. For at gøre brugeren opmærksom på klipningen, indføres en klip-indikator i brugerfladen.

Input: 2 lydbuffere
Output: Ny lydbuffer, opdatering af klip-indikatoren i brugerfladen

5.3.2 Design og implementering

Programmet skrives i TMT-Pascal. Dette valg har vi foretaget, da denne Pascal-compiler ikke har nogle af de hukommelsesbegrænsninger som Turbo Pascal har. TMT-Pascal genererer protected-mode programmer, som frit kan benytte hele PC'erens hukommelse. En sharewareversion af compileren kan hentes over internet på adressen: http://www.tmt.com

For at demonstrere fordelene ved at opbygge en mixer i software er programmet opbygget således, at signalbehandlingsdelen er uafhængig af brugerfladen og omvendt. Dette realiseres ved at definere de faktorer brugerfladen skal kunne ændre som globale variable. Således skal brugerfladen kun give brugeren mulighed for at ændre disse variable, og signalbehandlingsdelen kan køre uafhængigt af brugerfladen. Desuden er input/output delen ligeledes uafhængig af resten af programmet. Dette er dels gjort for, at man kan teste programmet ved at lave input/output med wave filer, og dels for at gøre det lettere senere at understøtte andre typer af lydkort.

De globale variable er alle defineret i unit'en VARIABLE. Det er gjort for at lette overskueligheden og højne fleksibiliteten. Desuden gør dette, at man ikke risikerer fænomenet 'circular unit reference', hvilket opstår, hvis en unit A bruger en unit B, der samtidig bruger unit A. Dette kan ellers ofte blive et problem, når man sætter de forskellige programdele sammen. I Tabel 3 ses variablene, som bruges til signalbehandlingen i UNIT'en EQ. Disse bruges senere i programdokumentationen og fremhæves med kursiv skrift.

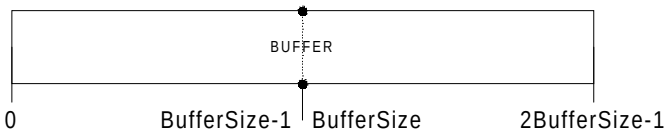
Den grafiske brugerflade, unit GUI, er defineret til at have en initialiseringsrutine og en deinitialiseringrutine, som foruden de globale variable og grafiske opdateringsprocedurer, er den eneste grænseflade til resten af programmet. På samme måde har lydkortsdelen, unit DSP og MIXER, initialiserings- og deinitialiseringsrutiner. Disse initialisering- og deinitialiseringsrutiner er samlet i unit'en STARTUP, som desuden nulstiller alle nødvendige variable og henter filterkoefficienterne fra en fil. Denne unit skal så bare kaldes fra hovedprogrammet, hvilket minimerer hovedprogrammets størrelse og kompleksitet.

Før lyden kan behandles i programmet, bliver lydkortet programmeret til kontinuert at levere en datastrøm af samples. I praksis realiseres dette ved at programmere lydkortet og den såkaldte DMA-controller (Direct Memory Access). DMA-controlleren overfører data fra lydkortet til en buffer i PC'ens hukommelse, uden at computerens CPU er involveret. Vi kan

Konstanter	Værdi	Beskrivelse
BufferSize	1024	Størrelsen på bufferen, skal altid være af 2^n
KoefficientFilename	'mix347.dat'	Filnavnet på filterkoefficientfilen
FirOrder	100	Antallet af koefficienter i impulsresponsen
Variable	**Type**	**Beskrivelse**
InputBuffer:	array[0..(2*BufferSize-1),1..2] of integer	Todelt buffer. 2x16bit input fra lydkortet.
OutputBuffer:	array[0..(2*BufferSize-1)] of byte	Todelt buffer. 8bit output til lydkortet.
InputPosition,OldInputposition, OutputPosition,OldOutputPosition:	longint	Hjælpevariable til den todelte, cirkulære bufferstruktur.
X1,X2:	array[0..(2*BufferSize-1)] of double	Todelt cirkulær lydbuffer
Y1,Y2:	array[0..(BufferSize-1)] of double	Cirkulær lydbuffer
A1lowpass,A1bandpass,A1highpass, A2lowpass,A2bandpass,A2highpass	double	Forstærkningskoefficienter til de to equalizere.
Gain1, Gain2	double	Forstærkningsfaktorer
Volume1,Volume2	double	Forstærkningsfaktorer
CrossVolume	double	Crossvolumens position
MasterVolume	double	Forstærkningsfaktor
StopProgram	boolean	Hovedprogrammet afbrydes, hvis sand
Hlowpass,Hbandpass,Hhighpass,H1,H2	array[0..FirOrder] of double	Impulsresponskoefficienter
ReCalcImpulse1,ReCalcImpulse2:	boolean	Genregn samlet impulsrespons, hvis sand
Gate1Threshold	longint	amplitude-grænse for fade-out
Gate1Hysterisis	longint	amplitude-grænse for fade-in
Gate1HoldTime	longint	Antal samples, der skal ventes inden fade-out
Gate1HoldCount	longint	Tæller, bruges sammen med hold-time
Gate1ReleaseSpeed	double	Gate1Amp ændring pr. sample under fade out
Gate1AttackSpeed	double	Gate1Amp ændring pr. sample under fade-in
Gate1Amp	double	Forstærkningsfaktor
Gate2Threshold	longint	Amplitude-grænse for fade-out
Gate2Hysterisis	longint	Amplitude-grænse for fade-in
Gate2HoldTime	longint	Antal samples, der skal ventes inden fade-out
Gate2HoldCount	longint	Tæller, bruges sammen med hold-time
Gate2ReleaseSpeed	double	Gate2Amp ændring pr. sample under fade out
Gate2AttackSpeed	double	Gate2Amp ændring pr. sample under fade-in
Gate2Amp	double	Forstærkningsfaktor

Tabel 3: De vigtigste globale variable i programmet. Disse bruges senere i programdokumentationen og fremhæves med kursiv skrift.

fra programmet hente data fra denne buffer, men skal undgå at læse fra bufferen samtidig med at DMA-controlleren skriver til bufferen. Lydbufferen opdeles derfor i to halvdele, se **Figur 39**; når DMA flytter data til den ene halvdel af bufferen, læser vi i programmet fra bufferens anden halvdel. For at vide i hvilken halvdel af bufferen, vi kan læse, installerer vi en såkaldt interrupthandler. En interrupthandler er en rutine, der bliver kaldt når en hardwaredel i computeren beder om at blive serviceret (interrupt request). Interrupthandleren bliver kaldt hver gang lydkortet (og DMA-controlleren) har fyldt halvdelen af bufferen med data.
I interrupthandleren ændrer vi variablen *InputPosition*, så den enten er lig 0 eller den halve bufferstørrelse. Denne variabel bruges i programmet til at adressere den n'te sample i den rigtige halvdel af bufferen, med *inputbuffer[inputposition+n]*, hvor n går fra 0 til ½ *buffersize-1*.



Figur 39: *Opbygningen af en todelt cirkulær buffer.*

På samme måde sendes de behandlede data fra computeren til lydkortet ved hjælp af DMA-controlleren og en interrupthandler.

Det skal bemærkes, at brugen af input/output-buffere giver en tidsforsinkelse af signalet på *BufferSize* samples. Ved en buffersize på 1024 svarer det til en tidsforsinkelse på 1024 samples/44100Hz = 23ms.

Før man fra programmet kan læse eller skrive i lyd-bufferne, må man tage højde for computerens hastighed. Hvis behandlingen af data mellem lyd-bufferne tager længere tid for CPU'en, end det tager lydkortet at fylde den ene halvdel af en lyd-buffer, er computeren for langsom til at udføre lydbehandlingen. Hvis computeren er hurtig nok, dvs. hvis CPU'en er færdig med lydberegningerne før DMA-controlleren og lydkortet, er man i programmet nødt til at vente på DMA-controlleren, før man læser næste halvdel af bufferen.

Alt dette gør, at man fra selve hovedprogrammet ikke skal bekymre sig om at hente og sende data til lydkortet, men bare skal vente indtil DMA-controlleren har overført dataene.

Signalbehandlingen er i programmet samlet i UNIT'en EQ og signalvejen er her følgende:

1. Procedure EQ.Input_16_to_FP
2. Procedure EQ.Do_FirFilter
3. Procedure EQ.Do_Gate
4. Procedure EQ.Calculate_VUpos
5. Procedure EQ.SetVolume
6. Procedure EQ.Output_Mix_FP_to_8

EQ.Input_16_to_FP

I denne procedure venter vi først på at DMA-controlleren har overført en blok af samples. Dette gøres ved at indføre en global variabel *OldInputPosition*, som ved opstart sættes lig *InputPosition*. Når *InputPosition* bliver forskellig fra *OldInputPosition*, kan programmet fortsætte. Indholdet af *InputBuffer*'en er 16-bits stereo-samples. Denne buffers indhold bliver nu konverteret til Floating Point (F.P.) og lagt i to nye buffere: X_1 og X_2. Grunden til, at vi konverterer til F.P., er, at det giver den bedste lydkvalitet, da F.P. er den datatype i PC'en, der har det største dynamikområde og den største præcision. Der findes flere typer Floating Point variable, vi har valgt at benytte "double precision" i hele signalvejen.

```
gentag indtil inputposition<>oldinputposition
oldinputposition=inputposition

for n=0 til (buffersize-1) gør
  {X1[n+oldinputposition]=FloatingPoint(inputbuffer[n+oldinputposition,kanal1])
  X2[n+oldinputposition]=FloatingPoint(inputbuffer[n+oldinputposition,kanal2])}
```

Det ses, at vi bevarer den cirkulære, todelte bufferstruktur i X_1 og X_2, som vi havde i *InputBuffer*'en fra lydkortet. Den cirkulære struktur skyldes, at FIR-filteret skal bruge tidligere samples til foldningen, og det lader sig fint gøre, da man altid kan gå $2 \cdot BufferSize-1$ samples tilbage.

For at realisere dette vælges *BufferSize* til at være 2^n, hvor n er et naturligt tal, og udføre en 'boolsk and' på offset'et med 2·*BufferSize*-1 således at offset kan gå fra 0 til 2·*BufferSize*-1.

EQ.Do_FirFilter:

3-bånds equalisering kan udføres ved at filtrere signalet med 3 separate filtre, vægte hvert af filtrene med en forstærkningskoefficient og slutteligt addere disse 3 signaler. En filtrering foretages ved at folde filtrets impulsrespons med signalets samples. Ifølge formel 42 fra afsnittet om tidsdiskrete systemer, foretages en foldning på følgende måde:

$$y[n] = \sum_{k=0}^{FirOrder} h[k] \cdot x[n-k]$$

Det ses tydeligt, at en filtreringen kræver *FirOrder*+1 multiplikationer og additioner per sample, hvor *FirOrder* angiver firfilterets orden, altså er antallet af koefficienter *FirOrder*+1. Ganges dette med antallet af samples per sekund og antallet af filtre i mixeren, giver det $2 \cdot 44100 \cdot 6 \cdot n = 529200 \cdot n$ operationer per sekund, hvor n er antallet af FIR-filterkoefficienter. Det betyder at filtreringen er den mest krævende proces i programmet. Det er derfor denne rutine, der bør bruges mest energi på at optimere grundigt i programmet.

Normalt ville EQ' en udføres således:

$Y[n] = (X[n]*H_1[n]\cdot A_1) + (X[n]*H_2[n]\cdot A_2) + (X[n]*H_3[n]\cdot A_3)$

hvor A_1, A_2 og A_3 er forstærkningsfaktorerne.

Det er imidlertid muligt at optimere udregningen på følgende måde:

$Y[n] = X[n]*q[n]$, hvor $q[n] = H_1[n]\cdot A_1 + H_2[n]\cdot A_2 + H_3[n]\cdot A_3$

Dette kan skrive på følgende måde:

$$
\begin{aligned}
Y[n] &= A_1 \sum_{k=0}^{FirOrder} X[n-k]\cdot H_1[k] + A_2 \sum_{k=0}^{FirOrder} X[n-k]\cdot H_2[k] + A_3 \sum_{k=0}^{FirOrder} X[n-k]\cdot H_3[k] \\
&= \sum_{k=0}^{FirOrder} X[n-k]\cdot A_1 \cdot H_1[k] + \sum_{k=0}^{FirOrder} X[n-k]\cdot A_2 \cdot H_2[k] + \sum_{k=0}^{FirOrder} X[n-k]\cdot A_3 \cdot H_3[k] \\
&= \sum_{k=0}^{FirOrder} X[n-k]\cdot \left(A_1 \cdot H_1[k] + A_2 \cdot H_2[k] + A_3 \cdot H_3[k]\right)
\end{aligned}
$$

Ledene der er samlet i parentesen i den sidste linie kan forudberegnes for alle værdier af k. Man får altså én samlet impulsrespons $H[k] = A_1 \cdot H_1[k] + A_2 \cdot H_2[k] + A_3 \cdot H_3[k]$. Hvis man ser bort fra forudberegningen, som ikke er særlig processorkrævende i forhold til foldningen, kommer antallet af beregninger ned på $2 \cdot 44100 \cdot 2 \cdot n = 176400 \cdot n$ operationer pr. sekund, hvor n er antallet af filterkoefficienter. Dette skyldes, at der kun skal udføres én foldning pr. kanal i stedet for 3 foldninger pr. kanal.

Man kan i denne del implementere gainfunktionen, hvorved der også spares på processorkraften. Forstærkningsfaktoren *Gain₁* / *Gain₂* ganges blot på impulsresponsens koefficienter.

Pseudokoden kommer dermed til at se således ud:

```
for n=0 til (FirOrder) gør

{H₁[n]=Gain₁(A₁lowpass·Hlowpass[n]+A₁bandpass·Hbandpass[n]+A₁highpass·Hhighpass[n])}
for n=0 til (FirOrder-1) gør

{H₂[n]=Gain₂(A₂lowpass·Hlowpass[n]+A₂bandpass·Hbandpass[n]+A₂highpass·Hhighpass[n])}
```

Forudberegningen sker før filtreringen, men den er dog kun nødvendig, hvis en af forstærkningsfaktorene er blevet ændret fra brugerfladen. Derfor beregnes *H₁* eller *H₂* kun, hvis variablene *ReCalcImpulse₁* eller *ReCalcImpulse₂* er sande.

Herefter udføres filtreringen i højre og venstre kanal, svarende til én foldning per kanal.

```
for n=0 til (BufferSize-1) gør
  {Y₁[n]=X₁[OldInputPosition+n]*H₁[FirOrder]
```

```
    for k=1 til (FirOrder) gør
    {Y1[n]=Y1[n]+X1[(OldInputPosition+n-k) and 2*BufferSize-1]*H1[FirOrder-k]}}
for n=0 til (BufferSize-1) gør
  {Y2n]=X2[OldInputPosition+n]*H2[FirOrder]
    for k=1 til (FirOrder) gør
    {Y2[n]=Y2[n]+X2[(OldInputPosition+n-k) and 2*BufferSize-1]*H2[FirOrder-k]}}
```

EQ.Do_Gate:

Gaten opbygges således, at dens forstærkningsfaktor *Gate₁Amp* enten bliver skruet op eller ned. Dog skal denne faktor ikke kunne overstige 1 og skal ikke kunne komme under 0. På den måde kan gaten kun være i fire forskellige tilstande:

1) Gaten er ved at lukke. Forstærkningsfaktoren falder med hastigheden *Gate₁ReleaseSpeed*.
2) Gaten er lukket. Forstærkningsfaktoren er lig 0.
3) Gaten er ved at åbne. Forstærkningsfaktoren stiger med hastigheden *Gate₁AttackSpeed*.
4) Gaten er åben. Forstærkningsfaktoren er lig 1.

Tilstanden 2 er et særtilfælde af tilstanden 1, hvor forstærkningsfaktoren ikke kan blive mindre. Tilstand nr. 4 er er særtilfælde af tilstand 3, hvor forstærkningsfaktoren ikke kan stige mere. Om gaten er i tilstand 1 eller 3 afhænger af forholdet mellem variablene *Gate₁HoldTime* og *Gate₁HoldCount*.

Gaten begynder at lukke, når lydsignalets amplitude har været mindre end *Gate₁Threshold* i *Gate₁HoldTime* samples. Dette undersøges, ved at lade en tæller, *Gate₁HoldCount* begynde at tælle op, når signalet er under "thresholdværdien". Kommer amplituden i denne fase på noget tidspunkt op over "thresholdværdien", nulstilles *Gate₁HoldCount*, og der tælles forfra.

Gaten begynder at åbne, hvis lydsignalets amplitude overstiger værdien af *Gate₁Threshold+Gate₁Hysterisis*. Derved nulstilles *Gate₁HoldCount*.

```
for n=0 til BufferSize-1 gør
  {Hvis Gate1HoldCount<Gate1HoldTime så gør
    {Gate1Amp=Gate1Amp+Gate1AttackSpeed
      Hvis Gate1Amp>1 så er Gate1Amp=1
```

```
      Hvis den absolutte værdi af Y1[n]<Gate1Threshold så
{Gate1HoldCount=Gate1HoldCount+1}
      ellers {Gate1HoldCount=0}
    }
  ellers
    {Gate1Amp=Gate1Amp-Gate1ReleaseSpeed
      Hvis Gate1Amp<0 så er Gate1Amp=0
      Gate1HoldCount=maxværdien for holdtime i brugerfladen
     }
Hvis den absolutte værdi af Y1[n]>(Gate1Threshold+Gate1Hysterisis) så
  {Gate1HoldCount=0}
Y1[n]=Y1[n]*Gate1Amp
```

En detalje er at *Gate1HoldCount* sættes lig med maksimumværdien for *Gate1HoldCount* i brugerfladen. Dette gøres for at gaten ikke skal begynde at lukke op, hvis man skruer op for "HoldTime" fra brugerfladen.
Ovenstående pseudokode og variable er gældende for gaten i første kanal. Gaten i kanal to fungerer på samme måde.

EQ.CalculateVUPos:

Her findes den sample i lydbufferen, som har den største amplitude. Herefter kaldes en procedure til at opdatere den grafiske repræsentation af VU-meteret i brugerfladen. Da denne procedure kræver et tal mellem 0 og 255 som input, skal amplituden tilpasses. Maksamplituden uden forstærkning fra volumereguleringen er 32768. Altså skal amplituden tilpasses ved at skalere med faktoren $256/32768 = 1/128$. Overstiger amplituden alligevel maksimumværdien skal amplitudeværdien "klippes".

VU-Meteret skal måle signalstyrken efter kanalvolume men før crossfader og mastervolume i signalvejen, men da VU-meteret er indsat før kanalvolume i programmet (optimering), skal amplituden også tilpasses ved at gange med volumevariablen.

```
Pos1=0
for n=0 til BufferSize-1 gør
  {Hvis den absolutte værdi af Y1[n]>Pos1 så Pos1=den absolutte værdi af Y1[n]}

Pos1=Pos1*(256/32768)*Volume1
hvis  Pos1>255, så Pos1=255
```

Og tilsvarende for kanal 2.

EQ.SetVolume:

Denne procedure er meget simpel. Signalet forstærkes med en forstærkningsfaktor for de to kanaler. Dette gøres ved at gange de enkelte samples i signalet med variablen $Volume_1$ i kanal 1 og med variablen $Volume_2$ i kanal 2.

Samtidig med dette indføres mastervolumen og cross-faderen for at spare på processorressourcerne ved at multiplicere hver sample med en samlet forstærkningsfaktor. *CrossVolume* regnes om til to forstærkningsfaktorer, en for hver kanal. *MasterVolume* forstærker begge kanaler med samme forstærkning, dvs. forstærkningen gælder for det samlede, mixede signal. Forstærkningsfaktorerne ganges blot på $Volume_1$ og $Volume_2$. Herefter ganges faktorene på de to lydbuffere.

```
Hvis CrossVolume<1 så gør
   {Temp1=Volume1*CrossVolume*MasterVolume;
    Temp2=Volume2*MasterVolume}
  ellers
    {Temp1=Volume1*MasterVolume
     Temp2:=Volume2*(2-CrossVolume)*MasterVolume}

for n=0 til BufferSize-1 gør
   {Y1[n]:=Y1[n]*Temp1}

for n=0 til BufferSize-1 gør
   {Y2[n]:=Y2[n]*Temp2}
```

EQ.Output_Mix_FP_to_8

Før signalerne fra de to kanaler sendes til lydkortet skal signalerne konverteres fra F.P. til 8 bit. Dette gøres ved at addere de to kanaler efter superpositionsprincippet. Det samlede signal skal herefter nedskaleres til 8 bit. Da input for de to kanaler er 16 bit signed, vil deres maks-amplitude være 32768, og det samlede signal vil derfor have en maksimal amplitude på $2 \cdot 32768 = 65536$. For at skalere ned til 8 bit signed, hvor den maksimale amplitude er 128; skal hver sample skaleres med faktoren 128/65536 = 1/512. Da de to kanalers signal godt kan

få en større amplitude end det ovenfor angivne, pga. filterforstærkning og volumenkontrol, klippes samples'ne til 8 bit.

Dvs. hvis outputtet fra $(kanal1 + kanal2) \cdot 128 / 65536$ er større en 127, så sættes output til 127 og den boolske variabel *Clipped* sættes til true. Det samme er selvfølgelig gældende for samples med negativ fortegn.

```
Clipped=falsk
Gentag indtil OldOutputPosition<>OutPutPosition
Sæt OldOutputPosition=OutputPosition
for n=0 til BufferSize-1 udfør
 {temp=afrund til heltal ((Y1[n]+Y2[n])·(128/32768·2))
  Hvis temp<-128 så gør
    {temp=-128
     Clipped=True}
  Hvis temp>127 så gør
    {temp=127
     Clipped=True}
  OutputBuffer[OldOutputPosition+n]=temp}
OpdatérBrugerflade(Clipped)
```

5.3.3 Test

De enkelte dele af programmet er efter implementeringen blevet testet hver for sig, og eventuelle fejl er blevet rettet. Endelig blev programmet testet efter integrationen af de enkelte programdele; hvor programfejl ligeledes blev rettet.

Ved den endelige test efter integrationen viste det sig, at programmet ikke var i stand til udføre equaliseringen med et tilfredsstillende antal filterkoefficienter. For at kunne gøre dette måtte den mest processorkrævende del af programmet optimeres. Da foldningen i EQ.Do_FIRFilter er den mest tidskrævende del af programmet, var det naturligt at optimere denne procedure. Da vi allerede havde optimeret algoritmen, blev vi nød til at optimere proceduren på en anden måde. Dette gøres ved at omskrive foldningsproceduren fra pascal til assemblerkode, hvor man programmerer direkte til CPU (Central Processing Unit) og FPU (Floating Point Unit).

5.3.4 Brugervejledning

Programmet MIXER347 ligger på den medfølgende diskette i mappen MIXER347. Programmet er lavet i to versioner, med en tekstbaseret brugerflade og en grafikbaseret

brugerflade.

Systemkrav

For at kunne anvende MIXER347 skal følgende systemkrav være opfyldt:

- CPU: min. 166 MHz Pentium
- Lydkort: Sound Blaster 16 eller 100% kompatibelt
- Grafikkort: min. VGA
- Operativsystem: MS-DOS, version 6.0 eller nyere.

Installation

Programpakken består af følgende filer:

```
mix347t.exe  ;Mixerprogrammet med tekstbaseret brugerflade
mix347g.exe ;Mixerprogrammet med grafikbaseret brugerflade
mix347.dat    ;Datafil med koefficienter til equalizeren
mix347.pcx   ;Grafikfil til den grafikbaserede brugerflade
```

Filerne kopieres fra mappen MIXER347 til en ønsket destination på computeren. Filerne skal ligge i samme mappe, for at programmet kan køre. For at starte den tekstbaserede udgave af programmet køres filen mix347t.exe. Den grafikbaserede udgave startes ved at køre filen mix347g.exe.

Programbeskrivelse

Mixeren bruger lydkortet som lydinput. Lyden behandles i to seperate kanaler. Der er mulighed for at vælge mellem flere af lydkortets lydkilder til hver af mikserens kanaler: Mikrofoningang, venstre/højre aux-kanal, venstre/højre CD-kanal

Herefter gennemløber signalet følgende behandling:

- Gain ±15 dB
- 3 bånds EQ, ±15 dB, Bas: 0-441Hz, Mellemtone: 441-2205Hz, Diskant: 2205Hz-22050Hz
- Gate
- Volume -∞ til 10 dB
- Crossfader

Sidst i signanalvejen er der en samlet volume, mastervolumen, som har en forstærkning på -∞ til 10 dB.

Ændringer af indstillinger på mixeren foretages på forskellige måder i de to versioner af programmet.

I versionen med den tekstbaserede brugerflade foretages ændringer af indstillinger ved at trykke de i brugerfladen angivne bogstaver på keybordet.

I versionen med den grafikbaserede brugerflade foretages ændringerne vha. musen. Man holder musepilen over det grafiske symbol for en af indstillingerne, trykker venstre musetast ned og trækker musen frem eller tilbage. Nogle af de grafiske symboler er kontakter. Disse aktiveres/deaktiveres ved at klikke på disse med venstre museknap.

5.4 Fordele ved digital signalbehandling i programmet

Der er mange fordele ved at lave en digital mixer ved brug af DSP, frem for at lave en analog mixer.
I modsætning til en analog mixer, er det let at ændre eller udskifte dele af en digital mixer. Dette skyldes at signalbehandlingen bliver udført af et stykke software i stedet for en hardwareenhed. Man vil således typisk have en standard hardwareenhed, i stedet for en specifik hardwareenhed, som kun er i stand til at udføre én bestemt funktion. Ved ændringer vil man derfor blot skulle ændre softwaren, hvorimod at man bliver nødt til at udskifte hele hardwaredelen ved en mixer konstrueret af en specifik hardwareenhed.

For at anskueliggøre ovenstående vises herefter hvordan man udfører en

- ændring af filterspecifikationer
- ændring af kanalantal
- tilføjning af nye effekter
- ændring af brugerflade

5.4.1 Ændring af filterspecifikationer

Da mikserprogrammet indlæser filterkoefficienterne fra en ekstern fil, er det relativt simpelt at ændre filterspecifikationerne.

Ændring af filterspecifikationer kan opdeles i 4 dele

- ændring af ordenstal
- ændring af afskæringsfrekvenser
- ændring af vinduesfunktion
- ændring af antal bånd i equalizer

For at ændre ordentallet, afskæringsfrekvenser eller vinduesfunktion, skal der bare beregnes nye koefficienter i Matlab. Dette gøres med scriptet LavEQ.m (ligger i mappen MATLAB på den vedlagte diskette). Hvis ordenstallet ændres skal konstanten *FirOrder* i UNIT'en variable ændres og programmet rekompileres.

For at ændre antallet af bånd i equalizeren, skal proceduren EQ.Calculate_FIRfilter desuden ændres til også at benytte overføringsfunktionerne for de nye bånd i beregningen af den samlede overføringsfunktion. Desuden skal der tilføjes en variabel pr. ekstra bånd, der fortæller om forstærkningen af båndet, og brugerfladen skal kunne ændre denne variabel. Da beregningen af den samlede impulsrespons kun udføres ved en ændring af forstærkningskoefficienterne, kan forøgelsen af båndantal laves så programmet kun kræver en forholdsvis lille forøgning af processorens regnekraft.

5.4.2 Ændring af kanalantal

Principielt er der ingen forskel på, om man mixer to kanaler eller flere kanaler. De enkelte samples lægges sammen efter superpositionsprincippet. Det samlede signal dæmpes, så der ikke opstår klipning/forvrængning ved at dividere de enkelte samples med kanalantallet.

Der er dog et problem i forbindelse med denne metode. For hver gang kanalantallet fordobles, vil bitopløsningen per kanal blive en bit mindre, og signal/støjforholdet vil dermed falde med ca. 6dB [Hüche, s561]. Begrundelsen herfor ligger i, at udgangens samlede signal/støjforhold holdes konstant (8 bit udgang), mens de enkelte kanalers signal skal halveres, for hver gang kanalantallet fordobles (for at undgå klipning). Dette er et stort problem ved 8 bits output.

Eksempelvis vil en 16 kanals mixer have en bitopløsning på 4 bit, hvilket er uacceptabelt. Ved 16 bits output er problemet dog ikke så stort.

Softwaremæssigt er det altså meget nemt at udvide mixeren med flere kanaler. Hardwaren, derimod, skal gennemgå en række ændringer. Antallet af inputs, og dermed A/D-konvertere skal ændres. Desuden vil de ekstra kanaler også kræve ekstra processorkraft.

5.4.3 Tilføjning af nye effekter

Ved at indsætte en gatefunktion og et vu-meter i vores mixer har vi demonstreret, hvor nemt det er at tilføje nye effekter i mixersoftwaren. Skulle effekterne realiseres analogt, ville udviklingen af det analoge kredsløb tage længere tid end softwareudviklingen, idet der skal tages højde for faktorer som f.eks. analoge komponenters tolerancer. Desuden ville mixerens produktionsomkostninger stige, idet der skulle bruges flere komponenter.

Det er simpelt at tilføje flere effekter i mixeren. Det kræver følgende ændringer i softwaren:

1) En ny procedure i signalvejen, som indeholder nødvendige algoritmer og variable
2) Nye globale variable, eventuelt nye lydbuffere
3) Tilpasning af brugerfladen til de nye funktioner og variable

En tilføjning af eksempelvis en delay effekt kan udføres på følgende måde:

Et delay mixer inputsignalet med en tidsforskudt udgave af inputsignalet. Algoritmen for denne funktion kan skrives med følgende pseudokode:

```
ud(n)=a*ind(n) + b*ud(n-d), hvor a og b er forstærkningsfaktorer, og d er
tidsforskydningen (delay).
```

For at realisere tidsforskydningen må en cirkulær buffer benyttes. Den cirkulære buffer skal have en længde svarende til antallet af samples, der kan fyldes i bufferen i løbet af delaytiden. Dissse beregninger samles i en ny procedure.
Bufferen samt koefficienterne a, b og d skal tilføjes til de globale variable.

Brugerfladen modificeres, så brugeren kan ændre variablene a, b og d.

5.4.4 Ændring af brugerflade

Brugerfladen i en digital mixer kan laves på mange måder. Man kan lave den, ligesom i vores eksempel, som en grafisk brugerflade, f.eks. på et LCD-display eller en monitor. Brugerfladen kan også laves på traditionel vis, med almindelige dreje-/skydeknapper. Eller man kan lave en kombination, som det ofte ses på mange digitale mixere. Fælles for brugerflader til digitale apparater er, at softwaren kan være uafhængig af brugerfladen.

En ændring af brugerfladen vil kunne foretages forholdsvist let. Dette kan give helt nye måder at anvende brugerfladen på. Brugerfladen kan, udover at give brugeren mulighed for at indstille forskellige parametre, f.eks. få en hukommelse, så den kan huske flere indstillinger. Er brugerfladen grafisk, kan mange nye faciliteter implementeres. En mulighed er at give brugeren mulighed for at predefinere en række indstillinger, som brugerfladen derefter kan manipulere med, f.eks. ved at lave en "glidende" overgang fra indstilling til indstilling. Dette er især meget brugbart i en mixer, da der tit kan være brug for at ændre flere parametre på én gang. Ved anvendelse af en mixer i musikstudiesammenæng, er der tit brug for at lave bestemte ændringer til bestemte (meget præcise) tidspunkter i musikken. Giver man brugeren mulighed for at "programmere" disse ændringer som funktion af tiden i den grafiske brugerflade, får mixeren helt nye muligheder. Det bliver således nemmere at bruge mixeren mere præcist med mulighed for nemt at rette fejl.

6. Perspektivering

Følgende opsummerer visse erfaringer vi har gjort, der har relation til problemstillinger vedrørende den generelle udvikling i elektronikindustrien:

- Det er væsentligt med et stærkt metodegrundlag i de tidligste faser af softwareudviklingen, idet det er her, at grundlaget for et vellykket udviklingsprojekt formes. Fejl i de tidlige faser medfører et overforbrug af tid.
- Det er nødvendigt med gode teknikker til opdeling i moduler og grænseflader.
- De fleste dedikerede systemer opnår deres funktionalitet gennem en nøje afbalanceret kombination af hardware og software. Det er derfor vigtigt, at man som softwareudvikler er omhyggelig med den software, der har en direkte grænseflade til systemets hardware.
- Det er væsentligt, at der på universiteterne undervises i softwareudviklingsmetoder, og denne undervisning bør være relateret til de metoder, der anvendes i erhvervslivet.

Vi har i vores problemanalyse nævnt, at en rapport, udarbejdet af en arbejdsgruppe under forskningsministeriet, med deltagelse af repræsentanter fra både industri- og uddannelsessektoren, behandler de problemer, der er knyttet til det stigende behov for kvalitetssoftware. Rapporten foreslår oprettelsen af et virtuelt Center for Softwareteknologi, CST, som bidrag til en løsning på problemerne. Centeret har til formål at fremme dansk softwareteknologi. Dette skal opnåes ved, at centeret, i et tæt samarbejde med erhvervslivet, gennemfører forskning, kompetenceopbygning, metodeudvikling samt teknologioverførsel på specifikke områder. Centerets primære aktivitet skal være konkrete anvendelsesorienterede samarbejdsprojekter med eksterne partnere, f.eks. virksomheder og godkendte teknologiske serviceinstitutter, som Dansk Teknologisk Institut.

- Centeret skal sikre sammenhæng i den virksomhedsrelevante forskning og samle den anvendelsesorienterede forskning på softwareområdet.

- Centeret skal arbejde for at højne kvaliteten og produktiviteten af softwareudviklingen i Danmark.
- Centeret skal gennemføre forsknings- og udviklingsprojekter, som formidles bl.a. ved at involvere studenter i projekterne. Derfor skal centeret forankres i et antal universitetsmiljøer.
- Centeret skal udgøre en ramme for informationsudveksling, hvor der flyder erfaring, viden og projektideer frem og tilbage mellem virksomheder og universiteter. Dette forudsætter en gensidig kontraktlig forpligtelse til aktiv medvirken.

Centeret skal fokusere sit arbejde på projekter, som dansk erhvervsliv finder interessante, herunder f.eks. apparatsoftware, netværkssoftware og undervisningssoftware. Interessen kan udmønte sig på flere måder, herunder en interesse i kommercialisering.

De udviklede metoder og teknikker skal ikke blot præsenteres i rapporter, men deres anvendelighed skal demonstreres gennem anvendelse i konkrete projekter, hvor virksomheder deltager aktivt og forpligtende[ITcenter].

Vi finder at forslaget om at oprette et sådant center er et spændende initiativ, og er som studerende interesseret i at blive involveret i projekter, der tager afsæt i problemer, der er interessante for dansk erhvervsliv.

7. Konklusion

Vi har med baggrund i konstruktionen af en lydmixer vist, hvor enkelt det er at ændre specifikationer for mixeren ved hjælp af simple softwareændringer. Vi har dermed vist, at digital signalbehandling giver visse fordele. Samtidig må vi konstatere, at vi i forsøget på at foretage ændringer er stødt på problemer, som vi ikke har formået at løse med vor nuværende viden. Lydkortet vi bruger kan ikke optage og afspille 16 bit på samme tid, hvorfor vores output kun er 8 bit, hvilket forringer signal-/støjforhold i en grad som er utilfredsstillende. Dette problem kunne løses med et ekstra lydkort. Vi ville gerne demonstrere, hvor enkelt det er at tilføje ekstra kanaler til mixeren, men er stødt på det problem, at en forøgelse af antallet af kanaler, foruden ovennænte problemer med lydkortet, bevirker en forringelse af bitopløsningen i de enkelte kanaler, og dermed en ringere lydkvalitet.

I forbindelse med udviklingen af programmet har vi tilstræbt en metodisk tilgang. Dette har været en stor udfordring for os, og vi har ikke formået at være konsekvente i dette fortsæt. Vi har erfaret, hvor vanskeligt det er at tage højde for alle problemer fra starten af et udviklingsprojekt.

Vi mener, at det er vigtigt at fokusere på softwareudvikling som en ingeniørmæssig disciplin, der er baseret på veludviklede metoder, og at tilegnelsen og udviklingen af sådanne metoder bør indgå i forskning og undervisning på universitetet. Et center, hvor der udveksles information og erfaring mellem virksomheder og universiteter, må være til begge parters fordel, og bør kunne medvirke til at højne softwarekvaliteten i Danmark.

8. Appendiks

8.1 A) Den eksponentielle notationsform for komplekse tal

Komplekse Tal

Et komplekst tal kan noteres på rektangulær form som

$$z = x + jy \tag{a1}$$

og på polær form som:

$$z = |z| \angle\phi \Leftrightarrow z = |z|(\cos\phi + j\sin\phi) \tag{a2}$$

Ved indsættelse af Eulers sætning:

$$\cos\phi + j\sin\phi = e^{j\phi} \tag{a3}$$

i den polære notationsform, fås den eksponentielle notationsform for komplekse tal:

$$z = |z| e^{j\phi} \tag{a4}$$

Enhedsvektoren

I den komplekse talplan er enhedsvektoren en vektor med vinklen ϕ og længden 1. På eksponentiel form noteres vektoren som $e^{j\phi}$. En multiplikation med enhedsvektoren svarer til en vinkeltilvækst på ϕ radianer.

Sinusnotation

Et sinusformet signal, med amplituden A og vinkelfrekvensen $\omega = 2\pi f$ [radianer/s], hvor f er repetitionsfrekvensen målt i Hz, kan nemmest skrives som:

$$u(t) = A \cdot \sin(\omega t) \tag{a5}$$

Dette kan også noteres på eksponentiel form, ved at lade en enhedsvektor rotere i positiv omløbsretning med jævn vinkelhastighed ω i det komplekse talplan. Det sinusformede signal vil nu kunne aflæses på imaginæraksen. På reelaksen vil cosinussignalet kunne aflæses. Dette kan skrives som:

$$u(t) = \cos(\omega t) + j\sin(\omega t) \Leftrightarrow u(t) = e^{j\omega t} \tag{a6}$$

Denne signaltype kaldes en kompleks elementarsvingning. Den komplekse elementarsvingning med negativ vinkelfrekvens -ω skrives som:

$$u(t) = e^{-j\omega t} \Leftrightarrow u(t) = \cos(\omega t) - j\sin(\omega t) \tag{a7}$$

Ved en kombination af disse to signaltyper kan man finde en eksponentiel notationsform, hvor sinus- og cosinussignalet er noteret særskilt. Sinusandelen findes på følgende måde:

$$\begin{aligned} j\sin(\omega t) &= \tfrac{1}{2}(\cos(\omega t) + j\sin(\omega t)) + \tfrac{1}{2}(-\cos(\omega t) + j\sin(\omega t)) \\ &= \tfrac{1}{2}(\cos(\omega t) + j\sin(\omega t)) - \tfrac{1}{2}(\cos(\omega t) - j\sin(\omega t)) \\ &= \tfrac{1}{2}e^{j\omega t} - \tfrac{1}{2}e^{-j\omega t} \Leftrightarrow \sin(\omega t) = \tfrac{1}{2j}(e^{j\omega t} - e^{-j\omega t}) \end{aligned} \tag{a8}$$

Cosinusandelen findes på følgende måde:

$$\begin{aligned} \cos(\omega t) &= \tfrac{1}{2}(\cos(\omega t) + j\sin(\omega t)) + \tfrac{1}{2}(\cos(\omega t) - j\sin(\omega t)) \\ &= \tfrac{1}{2}e^{j\omega t} + \tfrac{1}{2}e^{-j\omega t} = \tfrac{1}{2}(e^{j\omega t} + e^{-j\omega t}) \end{aligned} \tag{a9}$$

Et sinusformet signal kan noteres på forskellige måder: Som en almindelig cosinus- eller sinusfunktion, som en eksponentiel elementarsvingning og som en sum eller differens mellem to elementarsvingninger med henholdsvis positiv og negativ frekvens.

8.2 B) Invers Laplace transformation ved partielbrøksopløsning

I de tilfælde, hvor det ikke er muligt at foretage invers Laplacetransformation vha. tabeller med Laplacetransformationspar, kan man i stedet foretage den inverse Laplacetransformation ved partielbrøksopløsning. Denne metode bruges til at omskrive en funktion F(s), hvis nævnerpolynomium har mindst 2. grad; til en sum af partielbrøker:

$$F(s) = \frac{a_m s^m + a_{m-1} s^{m-1} + \ldots + a_1 s + a_0}{s^n + b_{n-1} s^{n-1} + \ldots + b_1 s + b_0} = \frac{k_1}{s - s_1} + \frac{k_2}{s - s_2} + \ldots + \frac{k_n}{s - s_n} \qquad \text{(a10)}$$

Den praktiske udførelse af invers Laplacetransformation ved partielbrøksopløsning foretages i tre trin:

1. F(s) omskrives med nævneren på faktoriseret form:

$$F(s) = \frac{T(s)}{N(s)} = \frac{T(s)}{(s - s_1) \cdot (s - s_2) \cdot \ldots (s - s_n)} \qquad \text{(a11)}$$

   hvor $s_1, s_2, \ldots s_n$ er nævnerpolynomiets rødder (poler).

2. Partielbrøkens tællerkoefficienter bestemmes vha. følgende formel:

$$k_x = (s - s_x) \cdot F(s)\Big|_{s=s_x} \qquad \text{(a12)}$$

3. Efter partielbrøksopdelingen foretages invers Laplacetrasformation til tidsdomænet ved hvert af partielbrøkens led vha. en transformationstabel, hvorefter det ifølge regel L1 i Tabel 2, er muligt at danne f(t) som summen af de invers Laplacetransformerede partialbrøker.

8.3 C) Filterkonstruktion - IIR-filtre

IIR-filtre

Konstruktionen af et digitalt IIR-filter udføres ved at opstille et analogt prototypefilter med en overføringsfunktion, der besidder de ønskede karakteristika. Derefter foretages der en z-transformation på overføringsfunktionen H(s), der beskriver systemets poler og nulpunkter, hvorved den digitale ækvivalent H(z) findes.

Det ideelle filter ville have en frekvenskarakteristik med en forstærkning på 1 (0 dB) i pasbåndet, og 0 (-∞ dB). Filteret ville desuden have en fuldstændig abrupt overgang mellem baspånd og stopbånd. Dette er dog ikke praktisk muligt, idet det ville kræve et filter af uendeligt orden (uendeligt mange poler).

Derfor er det nødvendigt at opstille en kravspecifikation for den ønskede frekvenskarakteristik, og ud fra denne bestemme det bedste filter.

Ved et praktisk realiserbart filter vil denne kravspecifikation mindst udspecificere:

1. Maksimal tilladelig pasbåndsrippel.
2. Minimal tilladelig stopbåndsdæmpning. I modsætning til et ideelt filter vil et praktisk realiserbart filter ikke have en dæmpning på -∞ dB i stopbåndet.
3. Flankestejlhed. Et ikke ideelt filter vil ikke have en brat overgang mellem pasbåndet og stopbåndet. Overgangen vil i stedet forløbe over et givet frekvensområde.
4. Ønsket steprespons.

Indenfor filterkonstruktion benyttes følgende størrelser:

f_a = afskæringsfrekvens i hertz, ωa = samme, i radianer.

f_s = stopbåndsfrekvens i hertz., ωs = samme, i radianer.

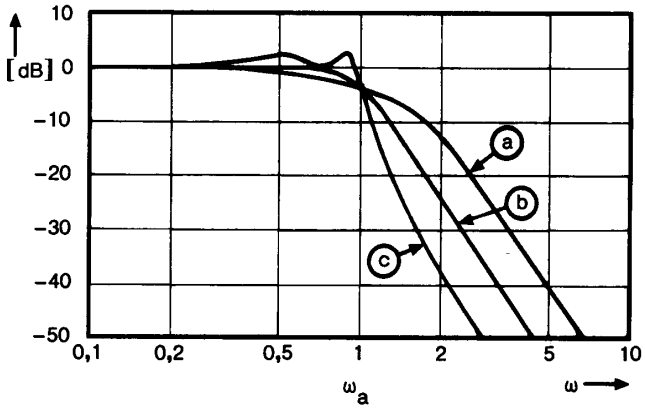
A_s = Stopbåndsdæmpning (ved f_s) i dB.

I det følgende vil 3 filterfunktioner, Butterworth- , Chebyshev- og Besselfunktionen, blive præsenteret.

Disse filterfunktioner, der er navngivet efter deres ophavsmænd, er optimeret efter forskellige kriterier, og har derfor forskellige egenskaber. Det er derfor nødvendigt, ud fra kravspecifikationen, at bestemme hvilken filterfunktion, der vil være den optimale i det pågældende tilfælde.

Forskellen mellem de 3 ovennævnte filtre illustreres mest hensigtsmæssigt ved at vise henholdsvis amplitude-, gruppeløbskarakteristik, steprespons og polplacering.

Amplitudekarakteristik

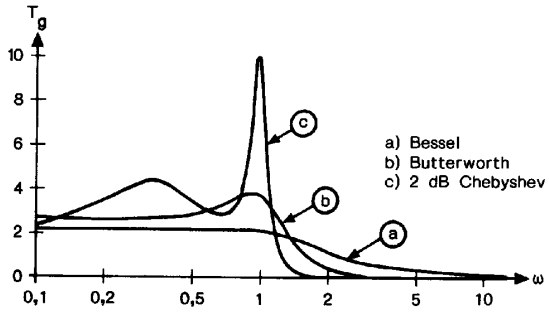


***Figur 40**: Amplitudekarakteristik for 4. ordens lavpasfiltre. a) Bessel. b) Butterworth. c) 2 dB Chebyshev. [Hüche, s 155]*

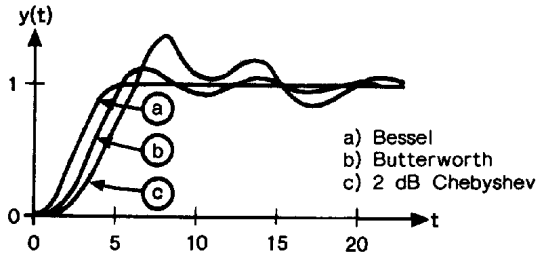
Af amplitudekarakteristikken ses Butterworthfunktionens styrke. Butterworth's funktion er optimeret med hensyn til konstant forstærkning i pasbåndet, dvs. mindst mulig pasbåndsrippel. Det ses desuden at Butterworthfiltre har en relativ stor flankestejlhed. (6dB/oktav pr. pol)
Chebyshevfunktionen udviser derimod stor pasbåndsrippel, men har til gengæld en større flankestejlhed. Dvs. at man kan realisere et filter af en given stejlhed ved et mindre ordenstal (polantal) end ved Butterworth- eller Besselfunktionen.

Af amplitudekarakteristikken ses det, at Besselfunktionen har en væsentlig dårligere stopbåndsdæmpning.

Gruppeløbskarakteristik og steprespons



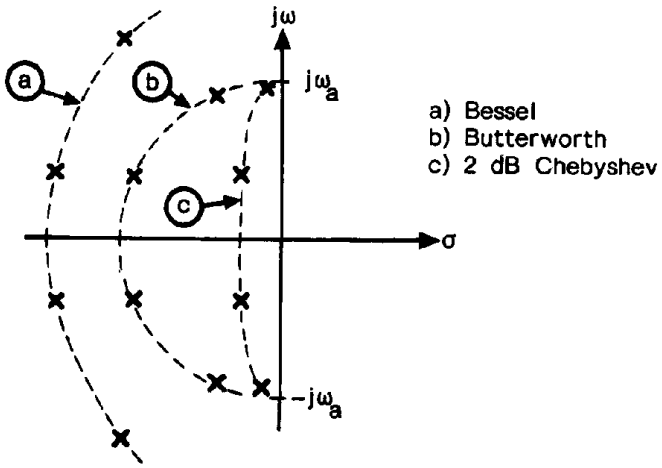
***Figur 42**: Gruppeløbskarakteristikker for 4. ordens lavpasfiltre. [Hüche, s 156]*



***Figur 41**: Steprespons for 4. ordens lavpasfiltre. [Hüche, s 156]*

Af gruppeløbskarakteristikken kan man se Besselfiltrets styrke, idet gruppeløbs-karakteristikken næsten er helt flad for denne filtertype. Dette gør, at Besselfiltret er ideelt til at gengive pulsformede signaler, hvilket kan ses på stepresponsplottet. Ligeledes viser graferne, at både Butterworth-, og især Chebyshevfilterfunktionen udviser ringning, hvilket vil sige, at de er mindre egnede til pulsformede signaler.

Polplacering

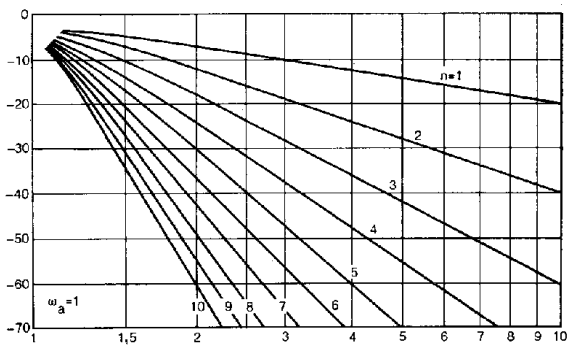


***Figur 43**: Polplacering for 4. ordens lavpasfilter.*
[Hüche, s 156]

Ud fra polplaceringen ses det, at jo nærmere polerne er på jω-aksen, jo kraftigere dæmper filtret. Desværre medfører en polplacering tæt på jω-aksen også en kraftig pasbåndsrippel og dårligere steprespons.

Ud fra denne analyse kan man konstatere, at Butterworthfunktionen bør benyttes, når kravet er konstant forstærkning i pasbåndet og forholdsvis stor dæmpning pr. pol.
Chebyshevfunktionen bør derimod anvendes, når kravet til konstant forstærkning i pasbåndet ikke er så højt, idet man dermed kan opnå et filter med en given dæmpning ved et lavere polantal.
Slutteligt bør Besselfilterfunktionen anvendes, når systemet skal behandle pulsformede signaler. Her skal dog bemærkes, at Besselfiltrets flade gruppeløbskarakteristik sker på bekostning af stopbåndsdæmpningen.

Efter filterfunktionen er bestemt, skal ordentallet (antal poler) bestemmes. Dette kan gøres ad matematisk eller grafisk vej.
Når ordenstallet skal bestemmes grafisk, gøres det ved hjælp af amplitudegrafer, der er et plot af frekvenskarakteristikken for en filterfunktion af 1. – 10. orden.



***Figur 44**: Butterworth amplitude-karakteristikker [Hüche, s 161]*

En sådan graf viser normalt kun filterfunktionen som et lavpasfilter, idet man benytter et lavpas prototypefilter til bestemmelse af polantallet for alle fire grundtyper af filtre. Dette gøres ved hjælp af en transformation af de bestemte karakteristika, der behandles herunder.

For at bestemme ordenstallet ved hjælp af amplitudegraferne, aflæser man på grafen for den pågældende filterfunktion, hvilket ordental der er nødvendigt for at realisere et filter med den ønskede stopbåndsdæmpning, A_s, ved stopbåndsfrekvensen, f_s. Dog skal der bemærkes at X-aksen er normeret i forhold til afskæringsfrekvensen, f_a.

$$\bar{f} = \frac{f}{f_a}$$

Derfor skal stopbåndsfrekvensen først frekvensnormeres, dvs. divideres med afskæringsfrekvensen, f_a.

Efter man har bestemt ordenstallet aflæses overføringsfunktionen H(s) i en tilhørende konstruktionstabel.

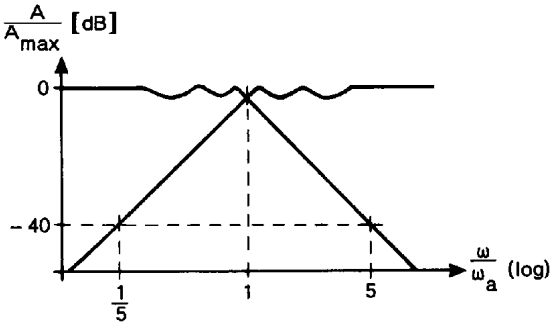
Filtertransformationer

I det foregående blev der vist, hvorledes overføringsfunktionen til et lavpasfilter findes. I dette afsnit beskrives det dels, hvorledes denne overføringsfunktion kan transformeres om til en af de 3 andre filtertyper, højpas, båndpas og båndstop, og dels hvorledes samme transformation benyttes til at transformere de besluttede karakteristika fra disse filtertyper til lavpas prototypefilteret, således at man kan benytte amplitudegraferne til at bestemme poltallet.

I det efterfølgende skelnes der mellem overføringsfunktionen for de forskellige typer af filtre ved en ekstra betegnelse, dvs. overføringsfunktionen for et lavpasfilter er H_{lp}. Ligeledes er betegnelserne for de 3 andre typer H_{hp}, H_{bp} og H_{bs}.

Selve filtertransformationen foregår ved at erstatte variablen s med en ny variabel $\hat{s}$, som er en funktion af s.

Højpastransformation



Figur 45*: Højpas filterkarakteristik opnået ved spejlning af en lavpas filter-karakteristik. [Hüche, s 175]*

Hvis et givet lavpasfilter spejles omkring afskæringsfrekvensen ω_a, ses det, at frekvenser med samme dæmpning på de to karakteristikker har reciprok sammenhæng, dvs. dæmpningen for den normerede frekvens 5 er den samme som dæmpningen for frekvensen 1/5.
Ud fra dette ses det, at funktionen som beskriver $\hat{s}$, er lig $1/s$.

$$H_{hp}(s) = H_{lp}(\hat{s})\Big|_{\hat{s}=\frac{1}{s}} \qquad \text{(a13)}$$

Det vil sige, at for at bestemme lavpasfilterets stopbåndsfrekvens f_s, benyttes transformationen

$$\hat{f}_s = \frac{1}{f_s} \qquad \text{(a14)}$$

Denne transformation bliver herefter vist for et 1.- og 2. ordens filter.

1. Orden:

$$H_{hp}(s) = \frac{\hat{A}_0}{s + \hat{B}_0}\Bigg|_{\hat{s}=\frac{1}{s}=\frac{\hat{A}_0}{\frac{1}{s}+\hat{B}_0}} \qquad \text{(a15)}$$

Herefter ganges igennem med $\frac{s}{\hat{B}_0}$

$$H_{hp}(s) = \frac{\frac{\hat{A}_0}{\hat{B}_0} \cdot s}{\frac{1}{\hat{B}_0} + s} \qquad \text{(a16)}$$

For at simplificere ovenstående yderligere inføres følgende størrelser

$$A_1 = \frac{\hat{A}_0}{\hat{B}_0}; \quad B_0 = \frac{1}{\hat{B}_0} \qquad \text{(a17)}$$

Indsat i (a16) giver det

$$H_{hp}(s) = \frac{A_1 s}{s + B_0} \tag{a18}$$

Hvilket er højpasfunktionen af 1. orden.

2. Orden:

$$H_{hp}(s) = \left. \frac{\left(\frac{A_0}{B_0}\right)}{s^2 + B_1 s + B_0} \right|_{s=\frac{1}{s}} = \frac{\left(\frac{A_0}{B_0}\right)}{\frac{1}{s^2} + \frac{B_1}{s} + B_0} \tag{a19}$$

Herefter ganges igennem med $\frac{s^2}{B_0}$

$$H_{hp}(s) = \frac{\frac{A_0}{B_0} \cdot s^2}{\frac{1}{B_0} + \frac{B_1}{B_0} \cdot s + s^2} \tag{a20}$$

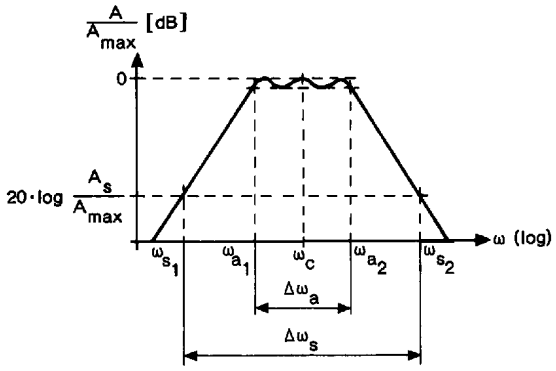
For at simplificere ovenstående indføres følgende størrelser

$$A_2 = \frac{A_0}{B_0}; \quad B_1 = \frac{B_1}{B_0}; \quad B_0 = \frac{1}{B_0} \tag{a21}$$

Indsat i (a20) giver det Højpasfunktionen af 2. orden:

$$H_{hp}(s) = \frac{A_2 s^2}{s^2 + B_1 s + B_0} \tag{a22}$$

Båndpastransformationen



***Figur 46**: Frekvensnormeret båndpasfilter med typisk amplitudekarakteristik. [Hüche, s 179]*

Analogt til højpasfilterfunktionen ses det, at båndpasfilteret er symmetrisk omkring en given frekvens. Denne frekvens er dog ikke afskæringsfrekvensen, ω_a, men centerfrekvensen, ω_c, og derfor forefindes der også her en reciprok sammenhæng mellem frekvenser med samme dæmpning.

$$\omega_{a_1} = \frac{1}{\omega_{a_2}} \text{ og } \omega_{s_1} = \frac{1}{\omega_{s_2}} \tag{a23}$$

For et båndpasfilter gælder det, at der frekvensnormeres i forhold til centerfrekvensen ω_c. Det vil sige, at de normerede båndpas- og stopbåndbredder er:

$$W_a = \frac{\omega_{a_2} - \omega_{a_1}}{\omega_c} = \frac{\Delta\omega_a}{\omega_c}$$
$$W_s = \frac{\omega_{s_2} - \omega_{s_1}}{\omega_c} = \frac{\Delta\omega_s}{\omega_c} \tag{a24}$$

Til bestemmelse af båndpasfilterets polantal benyttes formfaktoren, F, i stedet for

stopbåndsfrekvensen f_s. Formfaktoren er defineret som forholdet mellem stopbånds- og båndpasbredden.

$$F = \frac{W_s}{W_a} = \frac{\Delta\omega_s}{\Delta\omega_a} \tag{a25}$$

Transformationen udføres herefter på samme måde som ved højpastransformationen, idet sammenhængen $\hat{s} = \frac{1}{W_a}\left(s + \frac{1}{s}\right)$ benyttes.

1. Orden:

$$H_{bp}(s) = \left.\frac{\hat{A}_0}{\hat{s} + \hat{B}_0}\right|_{\hat{s}=\frac{1}{W_a}\left(s+\frac{1}{s}\right)} = \frac{\hat{A}_0}{\frac{1}{W_a}\left(s+\frac{1}{s}\right)+\hat{B}_0} \tag{a26}$$

Herefter ganges igennem med $W_a s$

$$H_{bp}(s) = \frac{W_a \hat{A}_0 s}{s^2 + 1 + W_a \hat{B}_0 s} \tag{a27}$$

For at simplificere ovenstående indføres følgende størrelser

$$A_1 = W_a \hat{A}_0;\ B_1 = W_a \hat{B}_o;\ B_0 = 1; \tag{a28}$$

Indsat i formel (a27) giver det

$$H_{bp}(s) = \frac{A_1 s}{s^2 + B_1 s + B_0} \tag{a29}$$

Her bemærkes, at 1. ordensfunktionen transformeres om til en 2. ordens funktion.

2. Orden:

$$H_{bp}(s) = \left.\frac{\hat{A}_0}{\hat{s}^2 + \hat{B}_1\hat{s} + \hat{B}_0}\right|_{\hat{s}=\frac{1}{W_a}\left(s+\frac{1}{s}\right)} = \frac{\hat{A}_0}{\left(\frac{1}{W_a}\left(s+\frac{1}{s}\right)\right)^2 + \hat{B}_1\left(\frac{1}{W_a}\left(s+\frac{1}{s}\right)\right) + \hat{B}_0} \tag{a30}$$

Herefter ganges igennem med $W_a{}^2 s^2$

$$\begin{aligned} H_{bp}(s) &= \frac{\hat{A}_0}{\left(\frac{s}{W_a} + \frac{1}{W_a s}\right)^2 + \frac{\hat{B}_1}{W_a}\left(s + \frac{1}{s}\right) + \hat{B}_0} \cdot W_a{}^2 s^2 \\ &= \frac{\hat{A}_0}{\left(\frac{s^2}{W_a{}^2} + \frac{1}{W_a{}^2 s^2} + \frac{2}{W_a{}^2}\right) + \left(\frac{\hat{B}_1 s}{W_a} + \frac{\hat{B}_1}{W_a s}\right) + \hat{B}_0} \cdot W_a{}^2 s^2 \\ &= \frac{W_a{}^2 \hat{A}_0 s^2}{s^4 + 1 + 2s^2 + \hat{B}_1 W_a s^3 + \hat{B}_1 W_a s + \hat{B}_0 W_a{}^2 s^2} \\ &= \frac{W_a{}^2 \hat{A}_0 s^2}{s^4 + \hat{B}_1 W_a s^3 + \left(2 + \hat{B}_0 W_a{}^2\right) s^2 + \hat{B}_1 W_a s + 1} \end{aligned} \tag{a31}$$

For at simplificere ovenstående indføres følgende størrelser

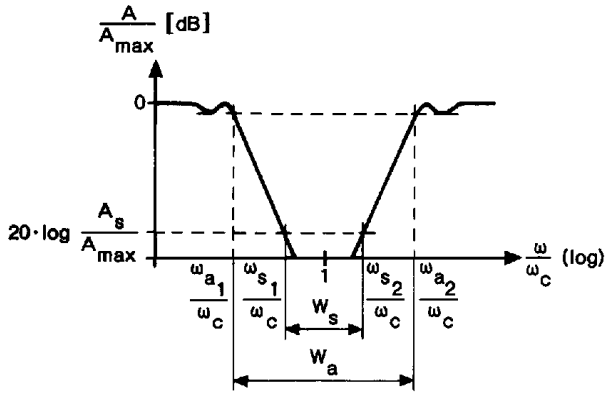
$$A_2 = W_a{}^2 \hat{A}_0; \quad B_3 = B_1 = W_a \hat{B}_1; \quad B_2 = 2 + W_a{}^2 \hat{B}_0; \quad B_0 = 1$$

Indsat i formel (a31) giver det:

$$H_{bp}(s) = \frac{A_2 s^2}{s^4 + B_3 s^3 + B_2 s^2 + B_1 s + B_0} \tag{a32}$$

Her bemærkes at transformationen af 2. ordensfunktionen har givet en 4.ordens funktion.

Båndstopfiltertransformationen



***Figur 47**: Frekvensnormeret båndstopfilter amplitudekarakteristik. [Hüche, s 188]*

Transformationen for et båndstopfilter er meget lig transformationen for et båndpas filter. Ligesom ved båndpastransformationen normeres i forhold til centerfrekvensen ω_c, men formfaktoren F er den inverse.

$$F = \frac{W_a}{W_s} = \frac{\Delta\omega_a}{\Delta\omega_s} \qquad \text{(a33)}$$

Derefter foregår transformationen på samme måde som ved båndpas; dog benyttes sammenhængen i stedet.

$$\hat{s} = \frac{W_a}{s + \frac{1}{s}}$$

1. Orden

$$H_{bs}(s) = \left.\frac{\hat{A}_0}{\hat{s} + \hat{B}_0}\right|_{\hat{s}=\frac{W_a}{s+\frac{1}{s}}=\frac{\hat{A}_0}{\frac{W_a}{s+\frac{1}{s}}+\hat{B}_0}} \tag{a34}$$

Herefter ganges igennem med $\frac{s^2+1}{\hat{B}_0}$ (a35)

$$H_{bs}(s) = \frac{\frac{\hat{A}_0 s^2 + \hat{A}_0}{\hat{B}_0}}{\frac{W_a s\left(s+\frac{1}{s}\right)}{\left(s+\frac{1}{s}\right)\hat{B}_0} + \left(s^2+1\right)} = \frac{\frac{\hat{A}_0}{\hat{B}_0}s^2 + \frac{\hat{A}_0}{\hat{B}_0}}{\frac{W_a}{\hat{B}_0}s + s^2 + 1} \tag{a36}$$

For at simplificere ovenstående indføres følgende størrelser:

$$A_2 = A_0 = \frac{\hat{A}_0}{\hat{B}_0}; \quad B_1 = \frac{W_a}{\hat{B}_0}; \quad B_0 = 1 \tag{a37}$$

Indsat i formel (a36) giver det båndstoptransformationen af 1. orden

$$H_{bs}(s) = \frac{A_2 s^2 + A_0}{s^2 + B_1 s + B_0} \tag{a38}$$

Ligesom ved båndpastransformationen ses det, at poltallet fordobles ved transformationen.

2. Orden

$$H_{bs}(s) = \left.\frac{\hat{A}_0}{\hat{s}^2 + \hat{B}_1\hat{s} + \hat{B}_0}\right|_{\hat{s}=\frac{W_a}{s+\frac{1}{s}}=\frac{\hat{A}_0}{\frac{W_a^2}{\left(s+\frac{1}{s}\right)^2}+\frac{\hat{B}_1 W_a}{s+\frac{1}{s}}+\hat{B}_0}} \tag{a39}$$

For at simplificere ovenstående indføres følgende størrelser:

$$A_0 = \frac{A_0}{B_0}; \quad B_3 = B_1 = \frac{W_a B_1}{B_0}; \quad B_2 = 2 + \frac{W_a^2}{B_0}; \quad B_0 = 1$$

Med disse størrelser kan (a39) skrives som:

$$H_{bs}(s) = \frac{A_0(s^4 + 2s^2 + 1)}{s^4 + B_3 s^3 + B_2 s^2 + B_1 s + B_0} \qquad \text{(a40)}$$

Af hensyn til den praktiske realisering opsplittes (a40) i to 2. ordens funktioner:

$$H_{bs}(s) = \frac{A_{01}(s^2 + 1)}{s^2 + B_{11} s + B_{01}} \cdot \frac{A_{02}(s^2 + 1)}{s^2 + B_{12} s + B_{02}} \qquad \text{(a41)}$$

Hvor $A_{01} = A_{02} = \sqrt{A_0} = \sqrt{\frac{A_0}{B_0}}$ (a42)

8.4 D) Bilineær z-transformation

Metoder til z-transformation

Ud fra et analogt filters specifikationer og overføringsfunktion, H(s), er det muligt at fremstille det resulterende digitale filters overføringsfunktion, H(z), med tilhørende koefficienter. Det gøres ved at transformere H(s) over i z-domænet ved hjælp af den såkaldte z-transformation. Z-transformation kan foretages efter flere metoder, bl.a.

- Matched z-transformation
- Impuls invariant z-transformation
- Bilineær z-transformation

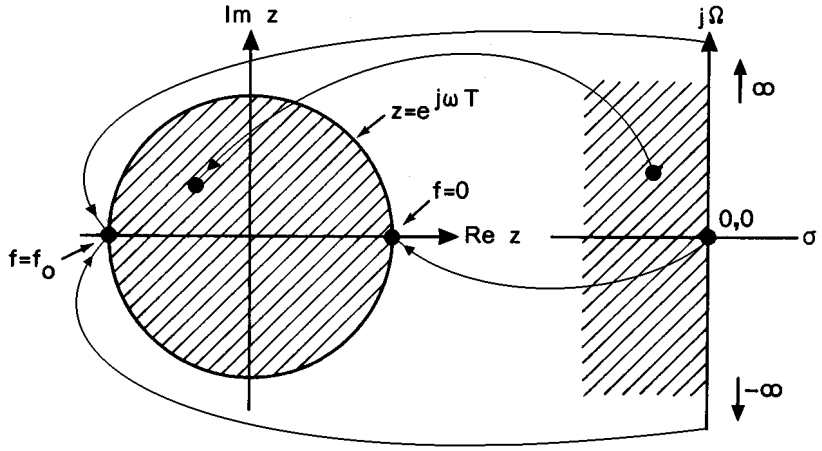
Hver metode har sine fordele og ulemper, der omtales senere. Men fælles for alle metoderne er, at transformationsprocessen formelt er den samme:

$$H(s) = \frac{\sum_{i=0}^{M} A_i s^i}{\sum_{i=0}^{N} B_i s^i} \xrightarrow{Z} H(z) = \frac{\sum_{i=0}^{M} a_i z^{-i}}{\sum_{i=0}^{N} b_i z^{-i}} \qquad \text{(a43)}$$

Det analoge filters poler og nulpunkter i s-planen overføres til z-planen, men uanset, hvilken type transformation man bruger, vil man aldrig kunne opnå nøjagtig reproduktion af analogfilterets fase- og amplitudekarakteristik. Årsagerne hertil afhænger af de forskellige typer af transformation.

Ved matched z-transformation og impuls invariant z-transformation får man en aliaseringsfejl. Det skyldes, at det digitale filters frekvensrespons gentager sig selv periodisk med samplingsfrekvensen, f_s, fordi s-planens $j\omega$-akse foldes uendeligt mange gange rundt på z-planens enhedscirkel. Denne fejl kan dog minimeres til en ubetydelighed ved at vælge en samplefrekvens, der er tilpas høj i forhold til filterets afskæringsfrekvens.

Ved bilineær z-transformation opstår aliaseringsfejlen ikke, fordi s-planens $j\omega$-akse kun foldes en halv gang rundt på z-planens enhedscirkel. Se Figur 48. Denne metode giver heller ikke ideel amplitude- og fasekarakteristik, idet overføringen af poler og nulpunkter ikke er lineær. Det kan der kompenseres for ved at lave en prewarping, det behandles senere.



***Figur 48**: Overførsel af s-planen til z-planen, ved bilineær z-transformation.*

Når man skal vælge, hvilken type transformation, man vil bruge, må man se på, hvilke overordnede mål man har sat sig. Drejer det sig om audiosignaler, må det overordnede mål være at opnå den bedst mulige tilnærmelse til det analoge filters amplitude- og fasekarakteristik. Hvad det angår, viser den impuls invariante z-transformation sig matched z-transformationen overlegen. Men bedst må siges at være den bilineære z-transformation, netop fordi man er fri for aliaseringsfejlen.

Bilineær z-transformation generelt

Det grundlæggende princip i bilineær z-transformation er substitution, idet s i det analoge filters overføringsfunktion erstattes med funktionsudtrykket $s = f(z)$.

Transformationsprocessen udtrykkes da således:

$$H(z) = H(s)\big|s = f(z) \qquad \text{(a44)}$$

Da det gælder at $z = e^{sT}$, kan s findes som

$$s = \frac{1}{T} \cdot \ln z \qquad \text{(a45)}$$

Substitueres der med (a45) i (a44) ville man teoretisk set opnå den ideelle transformation, men rent praktisk ville det blive urealiserbart, fordi man ikke kan finde et udtryk, der giver $\ln z$ på afsluttet form. Derfor må man bruge et tilnærmet udtryk for $\ln z$. Fra læren om rækkeudvikling ved vi, at $\ln z$ kan skrives som en uendelig række:

$$\ln z = 2 \cdot \left[\frac{z-1}{z+1} + \frac{1}{3} \cdot \left(\frac{z-1}{z+1} \right)^3 + \frac{1}{5} \cdot \left(\frac{z-1}{z+1} \right)^5 + \ldots \right] \qquad \text{(a46)}$$

Ved at iagttage (a46) ses det, at første led i udviklingsrækken er det mest betydende, hvorefter de følgende led hurtigt mister betydning, derfor kan man fjerne alle led efter det første, og stadig få en god tilnærmelse til $\ln z$.

$$\ln z \approx 2 \cdot \frac{z-1}{z+1} \qquad \text{(a47)}$$

Indsættes (a47) nu i (a45) fås substitutionsudtrykket, der anvendes ved bilineær z-transformation.

$$s = \frac{2}{T} \cdot \frac{z-1}{z+1} \qquad \text{(a48)}$$

Af praktiske grunde, kræves der ved implementeringen, at H(z) er noteret med negative potenser af z, derfor divideres der med z, således at (a48) bliver til:

$$s = \frac{2}{T} \cdot \frac{1-z^{-1}}{1+z^{-1}} \qquad \text{(a49)}$$

Den bilineære z-transformation kan nu udtrykkes ved (a44), som:

$$H(z) = H(s)\Big|_{s=\frac{2}{T}\cdot\frac{1-z^{-1}}{1+z^{-1}}} \qquad \text{(a50)}$$

Inden man med (a50) går i gang med at udføre den bilineære z-transformation, er man nødt til at se lidt på, hvorledes selve transformationen overfører s-planen til z-planen. En måde at gøre det på er at observere, hvorledes $j\omega$-aksen, der jo beskriver analogfilterets frekvensrespons, overføres til z-planens enhedscirkel.

Af praktiske grunde, som ses senere lader vi benævnelsen $j\Omega$ betegne det analoge s-plans imaginær variabel, mens $j\omega$ betegner det digitale s-plans imaginær variabel.

Først løses (a49) med hensyn til z:

$$z = \frac{\frac{2}{T}+s}{\frac{2}{T}-s} \qquad \text{(a51)}$$

substitueres s med $\sigma + j\Omega$ i (a51) fås:

$$z = \frac{\frac{2}{T}+\sigma+j\Omega}{\frac{2}{T}-\sigma-j\Omega} \qquad \text{(a52)}$$

For at vise, hvorledes højre og venstre halvplan af s-planen overføres til z-planen, skal vi se på den numeriske værdi af z i (a52), altså modulus.

$$|z| = \frac{\sqrt{\left(\frac{2}{T} + \sigma\right)^2 + \Omega^2}}{\sqrt{\left(\frac{2}{T} - \sigma\right)^2 + \Omega^2}} \qquad \text{(a53)}$$

Sættes σ i (a53) til henholdsvis større end én, lig med én, og mindre end én, bliver $|z|$ henholdsvis; større end én, hvilket viser at s-planens højreside kommer til at lægge uden for enhedscirklen i z-planen; lig med én, hvilket viser at s-planens origo kommer til at ligge i 1 på enhedscirklen; og mindre end én, hvilket viser at s-planens venstreside kommer til at ligge inden for z-planens enhedscirkel.

For at finde ud af, hvordan s-planens $j\Omega$-akse "lægges" ned på enhedscirklen i z-planen, må vi se lidt på argumentet til (a53), der benævnes $\angle z$. På $j\Omega$-aksen er $\sigma = 0$, det giver:

$$\angle z = \arctan\frac{\Omega T}{2} + \arctan\frac{\Omega T}{2} = 2 \cdot \arctan\frac{\Omega T}{2} \qquad \text{(a54)}$$

For $\sigma = 0$ bliver $\angle z = 0$, hvilket betyder, at s-planens origo overføres til (1,0) i z-planen. For σ gående mod $\pm\infty$ bliver argumentet $\angle z$ lig med $\pm 180°$. Det vil sige, at frekvenserne $\pm j\infty$ i s-planen overføres til (–1,0) i z-planen, hvilket svarer til den halve samplefrekvens, f_o; præcist som illustreret Figur 48

Ud fra ovenstående kan man slutte, at frekvensområdet $0 < f < \infty$ på det analoge filters frekvensakse transformeres over i frekvensområdet $0 < f < f_o$ for det digitale filter. Det betyder, at amplitude-karakteristikken for det digitale filter vil være en stærkt frekvensmæssigt sammenpresset udgave af analogfilterts amplitude-karakteristik. Denne sammenpresning kaldes for warping. For at kunne kompensere for denne warping, må man kende forholdet mellem Ω og ω, dette forhold søges derfor fundet. For z-planens enhedscirkel gælder det, at $z = e^{j\omega T}$. Indsættes det i (a49) fås:

$$s = \sigma + j\Omega = \frac{2}{T} \cdot \frac{1 - e^{-j\omega T/2}}{1 + e^{-j\omega T/2}} = \frac{2}{T} \cdot \frac{e^{j\omega T/2} - e^{-j\omega T/2}}{e^{j\omega T/2} + e^{-j\omega T/2}} = \frac{2}{T} \cdot \frac{2 \cdot j \cdot \sin(\omega T/2)}{2 \cdot \cos(\omega T/2)} \qquad \text{(a55)}$$

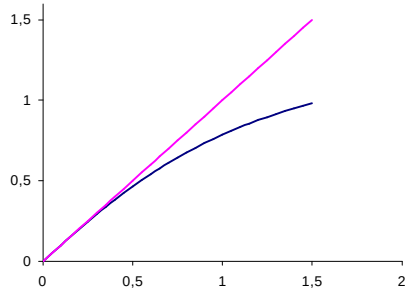
For frekvenser på $j\Omega$-aksen og frekvenser på enhedscirklen gælder, at $\sigma = 0$, det medfører:

$$j\Omega = \frac{2}{T} \cdot j \tan\frac{\omega T}{2} \Leftrightarrow \Omega = \frac{2}{T} \cdot \tan\frac{\omega T}{2} \qquad \text{(a56)}$$

Udtrykkes (a56) ved ω fås:

$$\omega = \frac{2}{T} \cdot \arctan\frac{\Omega T}{2} \qquad \text{(a57)}$$

Forholdet mellem Ω og ω, ses afbilledet i Figur 49



***Figur 49**: forhold mellem Ω (y-akse) og ω (x-akse)*

Efter at have fundet forholdet mellem Ω og ω er vi nu istand til at kompensere for frekvenswarpingen; denne proces kaldes prewarping.

Prewarping

Når man foretager en prewarping, laver man en slags forskydning af det analoge prototype filters amplitudekarakteristik, før man foretager z-transformationen. Forskydningen laves i

modsat retning af, selve warpings-effektens virkning, således at man opnår den ønskede amplitudekarakteristik, når z-transformationen er udført.

For at realisere prewarpingen, skal der indbygges en passende frekvensskalering i selve den bilineære z-transformation, det gøres ved at erstatte $\frac{2}{T}$ i (a49) med en prewarpingskonstant C. (49) kan da skrives som:

$$s = C \cdot \frac{1 - z^{-1}}{1 + z^{-1}} \qquad \text{(a58)}$$

Ydermere kan (a56) skrives, som:

$$\Omega_a = C \cdot \tan \frac{\omega_a T}{2} \qquad \text{(a59)}$$

hvor Ω_a er analogfilterets normerede afskæringsfrekvens, og ω_a digitalfilterets specificerede afskæringsfrekvens.

Ligestillet hermed gælder (a59), der gælder for stopbåndsfrekvensen:

$$\Omega_s = C \cdot \tan \frac{\omega_s T}{2} \qquad \text{(a60)}$$

Løses (a60) med hensyn til C fås:

$$C = \cot \frac{\omega_a T}{2} \qquad \text{(a61)}$$

Slutteligt kan (a61) skrives som:

$$H(z) = H(s)\Big|_{s=C\cdot\frac{1-z^{-1}}{1+z^{-1}}} \tag{a62}$$

For at illustrere, hvorledes (a62) anvendes i praksis, vil der i det følgende blive fremsat et beregningsudtryk, der fremstiller en bilineær z-transformeret 1. og 2. ordens overføringsfunktions koefficienter udfra, en analog prototypefunktions koefficienter.

1. ordenskoefficienter

$H(s)$ kan noteres som:

$$H(s) = \frac{A_1 s + A_0}{B_1 s + B_0} \tag{a63}$$

$H(z)$ kan skrives som:

$$H(z) = \frac{a_0 + a_1 z^{-1}}{1 + b_1 z^{-1}} \tag{a64}$$

Ved at udføre en bilineær z-transformation på (a64) fås $H(z)$ som:

$$H(z) = H(s)\Big|s = C\cdot\frac{1-z^{-1}}{1+z^{-1}} = \frac{A_1 C\cdot\frac{1-z^{-1}}{1+z^{-1}} + A_0}{B_1 C\cdot\frac{1-z^{-1}}{1+z^{-1}} + B_0} \tag{a65}$$

efter et vist reduktions- og omskrivningsarbejde fås:

$$H(z) = \frac{\frac{A_0 + A_1 C}{B_0 + B_1 C} + \frac{A_0 - A_1 C}{B_0 + B_1 C}\cdot z^{-1}}{1 + \frac{B_0 - B_1 C}{B_0 + B_1 C}\cdot z^{-1}} \tag{a66}$$

Ved sammenligning mellem (a64) og (a66) findes koefficienterne til den digitale overføringsfunktion som:

$$a_0 = \frac{A_0 + A_1 C}{B_0 + B_1 C} \tag{a67}$$

$$a_1 = \frac{A_0 - A_1 C}{B_0 + B_1 C} \tag{a68}$$

$$b_0 = 1 \tag{a69}$$

$$b_1 = \frac{B_0 - B_1 C}{B_0 + B_1 C} \tag{a70}$$

2. ordenskoefficienter

Den analoge 2. ordens overføringsfunktion kan parallelt til 1. ordens overføringsfunktionen noteres som:

$$H(s) = \frac{A_2 s^2 + A_1 s + A_0}{B_2 s^2 + B_1 s + B_0} \tag{a71}$$

ligeledes kan $H(z)$ kan noteres som:

$$H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}} \tag{a72}$$

Ved at udføre en bilineær z-transformation på (a71), jævnfør udregningerne til 1. ordens overføringsfunktionen, opnås $H(z)$ som:

$$H(z) = H(s)\Big|_{s = C \cdot \frac{1 - z^{-1}}{1 + z^{-1}}} = \frac{A_1 C^2 \cdot \left(\frac{1 - z^{-1}}{1 + z^{-1}}\right)^2 + A_1 C \cdot \frac{1 - z^{-1}}{1 + z^{-1}} + A_0}{B_1 C^2 \cdot \left(\frac{1 - z^{-1}}{1 + z^{-1}}\right)^2 + B_1 C \cdot \frac{1 - z^{-1}}{1 + z^{-1}} + B_0} \tag{a73}$$

efter et lidt mere omfattende reduktions- og omskrivningsarbejde fås:

$$H(z) = \frac{\frac{A_0 + A_1C + A_2C^2}{B_0 + B_1C + B_2C^2} + \frac{2\left(A_0 - A_2C^2\right)}{B_0 + B_1C + B_2C^2} \cdot z^{-1} + \frac{A_0 - A_1C + A_2C^2}{B_0 + B_1C + B_2C^2} \cdot z^{-2}}{1 + \frac{2\left(B_0 - B_2C^2\right)}{B_0 + B_1C + B_2C^2} \cdot z^{-1} + \frac{B_0 - B_1C + B_2C^2}{B_0 + B_1C + B_2C^2} \cdot z^{-2}} \qquad \text{(a74)}$$

Ved sammenligning mellem (a72) og (a74) findes koefficienterne til den digitale overføringsfunktion som:

$$a_0 = \frac{A_0 + A_1C + A_2C^2}{B_0 + B_1C + B_2C^2} \qquad \text{(a75)}$$

$$a_1 = \frac{2\left(A_0 - A_2C^2\right)}{B_0 + B_1C + B_2C^2} \qquad \text{(a76)}$$

$$a_2 = \frac{A_0 - A_1C + A_2C^2}{B_0 + B_1C + B_2C^2} \qquad \text{(a77)}$$

$$b_0 = 1 \qquad \text{(a78)}$$

$$b_1 = \frac{2\left(B_0 - B_2C^2\right)}{B_0 + B_1C + B_2C^2} \qquad \text{(a79)}$$

$$b_2 = \frac{B_0 - B_1C + B_2C^2}{B_0 + B_1C + B_2C^2}$$

(a80)

8.5 E) Invers z-transformation

Invers z-transformation ved partialbrøksopløsning.

Efter at have foretaget den bilineære z-transformation, og derved fundet den digitale overføringsfunktion H(z), er det nu muligt at finde udgangssekvensen y(n) for systemet. Denne udgangssekvens findes ved at invers z-transformere overføringsfunktionen.

Proceduren ved en invers z-transformation, der minder meget om invers Laplace-transformation, er som følgende:

Y(z) der er udtrykt ved X(z) gange H(z) omskrives så nævneren fås på faktoriseret form. Et krav til Y(z) er, at udtrykket er givet ved positive potenser af z:

$$Y(z) = \frac{T(z)}{N(z)} = \frac{T(z)}{(z-p_1)(z-p_2)...(z-p_N)} \qquad \text{(a81)}$$

hvor $p_1, p_2, \ldots p_N$ er nævnerpolynomiets rødder, hvilket også svare til Y(z)'s poler. T(z) er kort for tællerpolynomiet.
Herefter foretages en division med z, på begge sider af lighedstegnet, og funktionen noteres på partialbrøksopløst form.

$$\frac{Y(z)}{z} = \frac{T(z)}{zN(z)} = \frac{k_1}{(z-p_1)} + \frac{k_2}{(z-p_2)} + ... + \frac{k_N}{(z-p_N)} \qquad \text{(a82)}$$

Partialbrøkens tællerkoefficienter k_i beregnes som:

$$k_i = (z-p_i) \cdot \left. \frac{Y(z)}{z} \right|_{z=p_i} \qquad \text{(a83)}$$

$\frac{Y(z)}{Y}$ noteres nu på partialbrøksopløst form med de beregnede tællerkoefficienter, hvorefter

man multiplicerer alle led med z for at korrigere for divisionen med z tidligere i processen. Når det er gjort, kan de enkelte partialbrøker inverstransformeres ved hjælp af de givende tabeller. y(n) vil da være udtrykt som summen af de inverstransformerede.

9. Litteratur- og kildeliste

[DSP First]

Titel: DSP First - A Multimedia Approach

Forfattere: James H. McClellan, Ronald W. Schafer, Mark A. Yoder

Udgivet: Prentice-Hall, Inc, 1998

ISBN: 0-13-243171-8

[Hansson]

Artikel: Elektronik i dag - er software

Forfatter: Leif Hasson

Udgivet: Ingeniøren, udgave 19, 1997

[Hüche]

Titel: Digital Signal Behandling

Forfatter: Erik Hüche

Udgivet: Teknisk Forlag A/S, 1986

ISBN: 87-571-0874-9

[Ramskov]

Artikel: Måleinstrumenter ramt af PC'erne

Forfatter: Jens Ramskov

Udgivet: Ingeniøren, udgave 15, 1998

[Sound Pro]

Artikel: Computer busses & the audio industry

Forfatter: Dave Shapton

Udgivet: Sound Pro, december 1997

ISSN: 1368-3470

[**ITcenter]**

Titel: Forslag til Virtuelt IT-forskningscenter

Udgivet: Forskningsministeriet, maj 1995

URL: www.fsk.dk/fsk/publ/itcenter/

Dato: 18/4/98

[**Medialab]**

Titel: Multimedier i Danmark

Udgivet: Forskningsministeriet, marts 1996

URL: www.fsk.dk/fsk/publ/medialab/kap5.html

Dato: 18/4/98

[Supplerende Litteratur]

Titel: Discrete-Time Signal Processing

Forfatter: Alan V. Oppenheim & Ronald W. Schafer

Udgivet: Prentice Hall, 1989

ISBN: 0-13-216771-9

Titel: Soundblaster Programming Information ver. 0.9

Forfatter: André Baresel & Craig Jackson

Filnavn: sblaster.doc

Titel: Programming the SoundBlaster 16 DSP ver. 3.2

Forfatter: Ethan Brodsky

Filnavn: sb16doc.txt

Titel: DMA intro

Forfatter: Draeden of VLA

Filnavn: dmaintro.txt

Titel: Art of assembly, kapitel 17.3

Forfatter: Randall Hyde

URL: http://webster.ucr.edu/Page_asm/ArtOfAssembly/CH17/CH17-3.html

Dato: 21/5/98