Kurs: FYS3230 Sensorer og måleteknikk	Gruppe:	Gruppe-dag:
Oppgave: **LABORATORIEØVELSE NR 1**		
Omhandler:		

Bruk av programsystemet Labview i måleteknikk
Innføring i Labview
Modelltilpasning med minste kvadraters metode
Digitale filtre og reduksjon av støy

Revidert, 10 mai 07, O. Sveen
Revidert 10 sept 07, H.Vangli
Revidert 2.nov 07 T.Lindem

Utført dato:	Utført av: Navn: email: Navn: email:
Godkjent:dato:	Godkjent av:
Kommentar fra veileder:	

1. Datamaskin og programvare.

Laboratorieøvelsene i FYS 3230 krever bruk av datamaskin og ferdige programmer laget i programmeringsspråket LabVIEW. Det anvendes metoder som kunne vært utført ved hjelp av programmering på hvilken som helst type datamaskin i hvilket som helst programmeringsspråk. Kursets innhold blir nødvendigvis noe styrt av de verktøy som er valgt, d.v.s av maskinvalg og valg av programmeringsverktøy. Litt av denne første laboratorie-veiledningen omhandler teknikker som er nødvendig å beherske for å bruke programmeringsverktøyet

Hvis man skulle bli spesielt interessert i de anvendte verktøy, må man være klar over at dette egentlig ikke er eksamens-stoff. HELP-menyer i LabVIEW vil kunne gjøre spesielt interesserte til de reneste virtuoser på de anvendte verktøy.

En del avsnitt i disse veiledningene er en fryktelig blanding av norsk og engelsk. Dette er gjort fordi vi tror det er dumt å oversette alle de engelske uttrykkene i bruksanvisningene for LabVIEW. Slik oversettelse vil øke vanskelighetene når man konsulterer manualene for å lære mer om bruken av verktøyet.

1.1 Oversikt

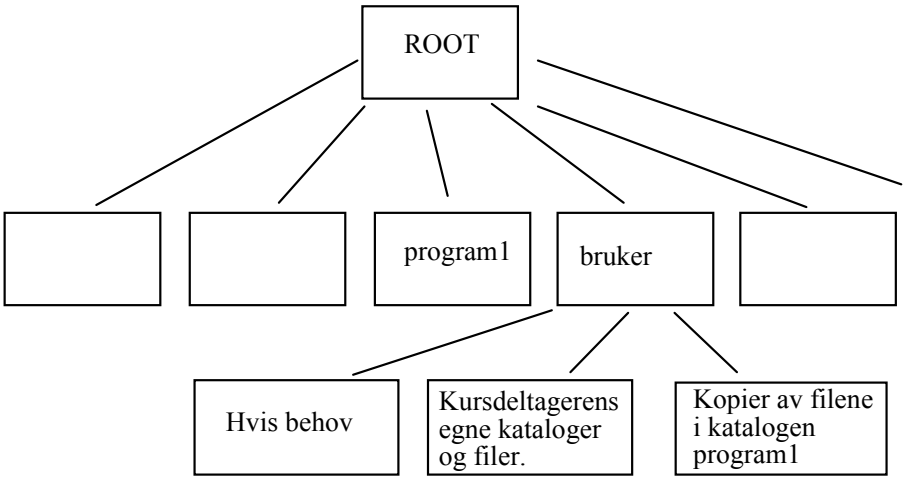
Vi skal bruke operativsystemet WINDOWS fra Microsoft. Som programmeringsspråk bruker vi **LabVIEW,** et grafisk programmeringsverktøy som finnes både for WINDOWS, MAC og SUN arbeidsstasjoner. Til dette kurset er det laget en rekke spesial-programmer som er lagret på harddisken.

Programmering i LabVIEW består i å lage moduler som kalles "Virtuelle Instrumenter" (VI). Et VI kan betraktes som en subrutine eller funksjon i konvensjonelle programmeringsspråk. Et VI tegnes på en dataskjerm ved hjelp av teknikker som er vanlige i tegne- og male-programmer. Instrumentet består av et tegnet frontpanel, og et tegnet koblingsdiagram. Frontpanelet inneholder "input" og "output" til/fra instrumentet fra/til brukeren eller andre VI. Koblingsdiagrammet beskriver koblinger og prosessering som gjøres inne i instrumentet, de må tegnes i det grafiske programmeringsspråket "**G**". Store programsystemer vil ofte bestå av flere VI. Data kan sendes mellom disse, på samme måte som data kan overføres mellom vanlige subrutiner.

LabVIEW blir levert med et stort bibliotek av ferdige VI som utfører funksjoner av allmenn interesse. For å lage et programsystem må man derfor velge ut de riktige biblioteks-instrumenter, og kople dem sammen med passende prosesseringsmoduler. Det er også mulig å koble seg til C-rutiner som da må være skrevet etter spesielle konvensjoner.

1.2 Filstruktur.

WINDOWS har et filsystem med trestruktur (som UNIX). Dette benyttes så vel på permanente disker som på løse disketter. Figuren nedenfor gir et eksempel på dette og viser hvordan forskjellige filkataloger er satt opp på harddisken på vår maskin.



ROOT katalogen er alltid øverst. Dette vil i Windows tilsvare den øverste mappen på den lokale harddisken(c:). Under denne finner vi to kataloger som er av viktighet i kurset – program1 - og – bruker- . Den første inneholder alle de programmene som er laget ferdig til bruk i kurset. Studentene bør prøve å unngå å ødelegge disse programmene. Den andre katalogen - bruker - er som navnet antyder beregnet til arbeid med laboratorieoppgavene. Når WINDOWS startes opp, vil alle programmene i katalogen -program1- bli kopiert til katalogen -bruker-. Eventuelle forandringer utført og lagret på programmene i -bruker- går derfor tapt hver gang systemet startes. For å lagre forandringer slik at de overlever til neste gangs pålogging, må man lagre filene på sin egen nettverksdisk el.l.. Man kan således ikke regne med at filer blir lagret i maskinen fra gang til gang. Er det noe du vil spare, må det overføres til din egen nettverksdisk

Hvis noen studenter ønsker å anvende tekstbehandlingsprogram for å skrive lab-rapportene, er WORD lagt inn på laboratoriemaskinene. EXCEL anbefales for en del av regneoppgavene.

1.3 LabVIEW

LabVIEW programmer kalles Virtuelle Instrumenter (VIs). Et VI består av tre deler :

- Front Panelet
- Koblings Diagrammet
- Konnektoren (Som bare anvendes når programmet kalles opp som subrutine)

Trinn for trinn vil du nå bli hjulpet gjennom en prosedyre for å åpne et VI, og bli kjent med de grunnleggende mekanismer og begreper som gjelder for LabVIEW programmer.

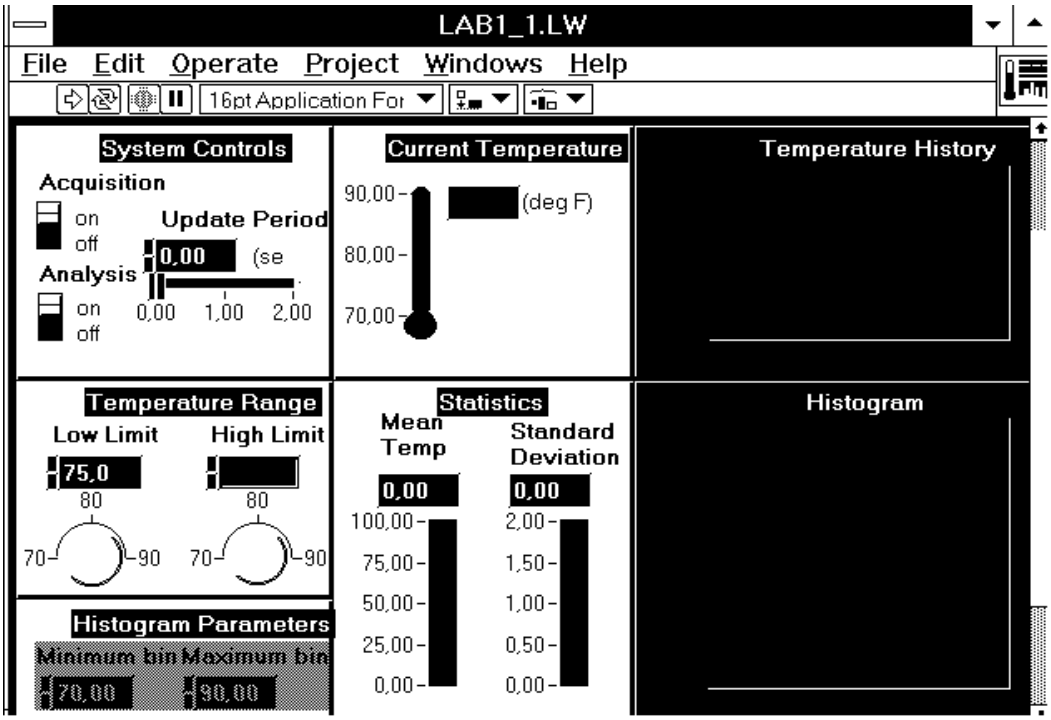
Hvis vi lukker et Koblings Diagram, vil bare Koblings Diagrammet bli lukket. Hvis vi lukker et Front Panel, vil både Front Panelet og Koblings Diagrammet bli lukket.

1.3.1 Front panelet

Logg deg inn på en av lab-pc'ene

1. Start opp LabVIEW ved å dobbeltklikke på **Snarvei til LabVIEW** som burde ligge på **Skrivebordet**. Hvis ikke dette fungerer, går inn på Start - All Programs - National Instruments LabVIEW 8.2. Etter en stund kommer det frem et **tomt Front Panel vindu**.

2. Trykk på **Browse** nederst i den gule Open-menyen. Trykk på "**My Computer**", dobbeltklikk på **c: Windows XP**, siden klikker du deg inn på mappen **bruker**. Velg så "All files" i "Files of type". Flere filer skal nå bli tilgjengelige. Start opp filen **laba_1.lw**. Filen blir da lest inn i LabVIEW, og etter noen sekunder vil Front Panelet vise seg som i følgende illustrasjon. Front Panelet inneholder flere typer kontrollorganer (knotter, skyvekontroller og brytere) som gir fra seg tall eller logiske variabler, og endel grafiske indikatororganer (skjermer og et termometer).



Både Front Panel vinduet og Kobling Diagram vinduet inneholder øverst oppe en "palett" med kommandoknotter og status indikatorer som må brukes for å kontrollere et VI. Paletten inneholder endel interessante "knotter"

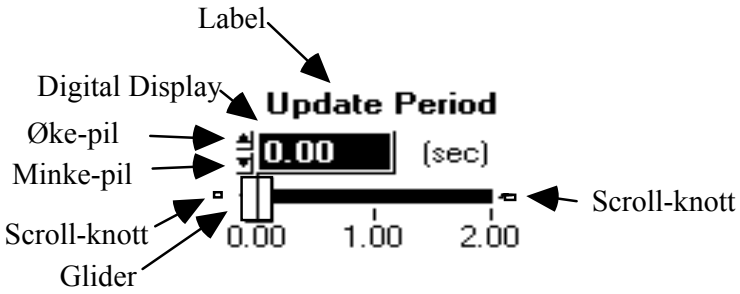
Et program (et VI) kjøres ved å klikke på Run knotten :  Run knotten skifter da utseende for å indikere at programmet kjører :  . Samtidig aktiviseres en Stopp knott i paletten, som kan klikkes for å stoppe programeksekveringen :  . Når et program ikke kjøres, inneholder palett-linjen dessuten felter for kontroll av fontstørrelser og grafisk editering.

Programmet LABA_1.LW er en simulering av et program som overvåker utviklingen av temperatur. Temperaturen måles, og vises frem på termometeret og på en grafisk fremstilling. Glidekontrollen "Update Period" styrer tidsintervallet mellom hver temperatur-lesing. Den grafiske fremstillingen viser også høy og lav grense for et temperaturområde, kontrollert av "skruknottene" merket Low Limit og High Limit. Hvis en målt temperatur er utenfor grensene for dette området, vil en alarm vise seg ved siden av termometeret.

Når programmet startes, fortsetter det å gå inntil
- **Stop-knotten** ovenfor klikkes, eller
- "**Acquisition**"-bryteren på Front Panelet klikkes til "Off".

Det er også mulig å bruke "Analyses"-bryteren på Front Panelet til å slå analysering av innleste data av og på. Analyseringen består i å generere et histogram over innleste temperaturverdier, samt å generere løpende verdier for gjennomsnitt og standard avvik.

Når et program kjøres, kan man anvende en "Operating Tool", som ser slik ut : til å endre verdier slik som High Limit, Low Limit, og Acquisition eller Analyses knottenes stilling. "Klikk & Dra" i gliderkontrollens glider, eller i skruknottens omkrets. For å endre et tall, kan man gjøre et Klikk eller "Klikk & Dra" med musen og få en kursor plassert inne i tall-feltet. Deretter kan nye data innskrives med tastaturet. Avslutt med å klikke i en ubrukt del av Front Panelet, eller på "Enter"-knotten til venstre for Run knotten : : "Operating tool" kan også velges ved å velge i en "tools"-palett, som vises frem ved valg i "Windows"-menyen. Glidekontroller slik som Updata Period kan også endres ved å plassere "Operating tool" på glideren, og dra denne til en ny posisjon. Glidekontroller har også to "Scroll" knotter. Numeriske verdier kan alltid økes eller minkes ved å klikke på pilene like til venstre for tallet.



Nyinnskrevne verdier ikke blir gyldige før man klikker

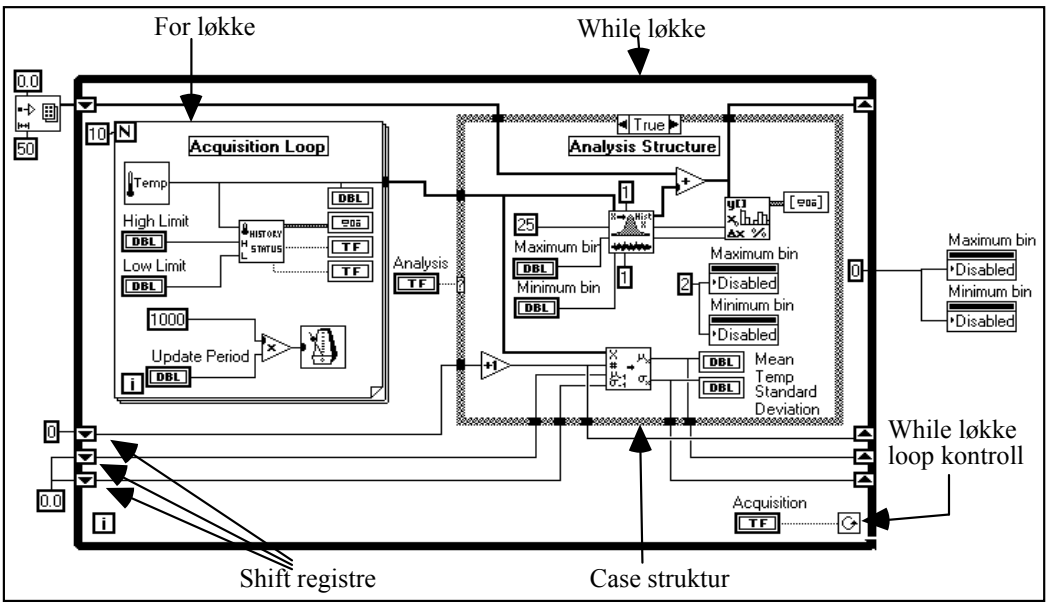
i en fri del av Front Panelet, eller på - knotten.

Eksperimenter med å endre de andre kontrollene på Front-Panelet mens programmet går. Stop eksekveringen ved å klikke på Acquisition bryteren.

1.3.2 Koblings Diagrammet

Nedenfor presenteres Koblings Diagrammet som tilsvarer det Front Panel som er kjent fra simuleringen av temperaturmåling. Denne korte gjennomgangen vil ikke være tilstrekkelig til å forstå dette Diagrammet ordentlig, en grundig forståelse vil være avhengig av flere følgende avsnitt.

Frembring Koblings Diagrammet ved å velge "**Show Diagram**" i "**Windows**"-menyen.



Hvert eneste Front Panel har et Koblings Diagram, de to er like uadskillelige som Knoll og Tott. Koblings Diagrammet utgjør programmet, og er skrevet i det grafiske programmerings-sproget **G**. **Koblings Diagrammet kan betraktes som kildekode.** Komponenter i Koblings Diagrammet tilsvarer For løkker, Case strukturer og aritmetiske operasjoner. Komponentene er koblet sammen med "ledninger" som illustrerer hvordan data flyter i Diagrammet.

Den ytterste strukturen er en **While løkke**, som vil fortsette å bli utført inntil Acquisition bryteren på Front Panelet blir slått av. På kanten av While løkken sees en del terminaler med piler inni. Disse terminalene kalles "**Shift Registers**", og brukes til å lagre verdier fra en eksekvering av løkken til den neste. De 4 verdierne som lagres her er selve histogrammet, serie nummer, gjennomsnittsverdi og standard avvik.

Inne i den ytre While løkken finnes en **For løkke** og en **Case struktur**. Datainnsamlingen foregår inne i For løkken. 10 temperatur-avlesninger utføres, og hver avlesning angis på termometeret og plottes opp. Hver avlesning vil også sjekkes mot den høye og lave grense satt på Front Panelet, og resulterer i varsel om temperaturer utenfor grensene.

Når 10 avlesninger er utført, aktiveres Case strukturen for eventuelt å analysere de 10 datapunktene. Hvis Front Panelets "analyses" bryter er av, vil ingen analyse utføres. Dette kan ses ved å klikke på en av pilene ved siden av ordet "*True* " øverst på kanten av strukturen. Vi får da se Casens "*False* " avsnitt. Ingen analysering av data skjer her, og grunnlaget for histogrammet settes lik null. Et tilsvarende museklikk bringer "*True* " avsnittet tilbake. Vi ser at de målte data sendes inn i to andre VI, ett som genererer histogrammet, og ett annet som genererer løpende gjennomsnitt og standard avvik.
Ikke fortvil om du ikke forstår alt som ovenfor har vært anført. Dette er ment som en grov illustrasjon av ting det er rimelig enkelt å få til i et LabVIEW program. Det finnes utfyllende informasjon tilgjengelig gjennom pekere i oppstartsbildet til LabVIEW. HELP-funksjoner kan også brukes til å lære seg mere.

2. Modell-tilpasning. Minste kvadraters metode.

På grunnlag av et sett målinger ønsker vi ofte å foreta en sammenligning med en modell. Modellen kan være en teoretisk formel, eller det kan være vi ønsker å beskrive målingene ved en enkel funksjonssammenheng (f.eks. et polynom). I alle tilfelle må vi velge et kriterium (*Figure-of-merit function*) som måler hvor godt data og modell med et visst valg av parametre stemmer overens. Målingene vil typisk være beheftet med målefeil slik at de aldri eksakt vil stemme med modellen. Vi bør derfor på grunnlag av statistisk teori kunne avgjøre hvor store avvik vi kan godta, dvs. om modellen er brukbar eller må forkastes.

Et program som foretar en tilpasning av målte data til en modell bør derfor kunne gi

- verdier for parametre
- estimat av usikkerhet i parametre
- statistisk mål for nøyaktigheten i tilpasningen (*Goodness-of-fit*)

Den vanligste metoden for tilpasning er å bestemme parametrene slik at summen av det kvadratiske avvik mellom målte verdier og beregnede verdier blir minst mulig (minste kvadraters metode). Hvis målefeilene er uavhengige og normal-fordelte med konstant standard avvik, sier statistisk teori at dette gir det beste estimat av parametrene (*maximum likelihood estimation*). Målefeilene er ofte ikke helt normal-fordelt, men metoden gir likevel en brukbar indikasjon på hvor sannsynlig det er at den valgte modell passer med målingene.

2.1 Minste kvadraters metode.

Anta at vi har N målte verdier (x_i, y_i) og en modell med M justerbare parametre som beskriver sammenhengen mellom x og y . Vi ønsker å bestemme parametrene a_1, a_2, ...a_M slik at vi får minimum for

$$\chi^2(chi-square) = \sum_{i=1}^{N}\left(\frac{y_i - y(x_i;a_1,\ldots,a_M)}{\sigma^2}\right)^2 ,$$

hvor σ_i er usikkerheten (standardavviket) for den målte verdien y_i .

Metodene for å finne minst mulig *chi-square* involverer matematikk og statistisk teori som er forsøkt forklart i heftet med forelesningsreferater. Ved tilpasning til en rett linje (y=Ax+B), vil standard avviket for A og B være mulig å finne ut ved hjelp av matriser. Hvis de målte verdier ikke har kjente standard avvik, kan det likevel anvendes en standard-prosedyre for å estimere standard avvik for de utregnede parametrene.

I LabVIEW finnes det ingen innebygde muligheter for å regne ut standard avvik i parametre eller estimater for å bestemme sannsynligheten for at en modell en den "riktige" å bruke for å tilpasse de foreliggende data. LabVIEW krever at modellen er lineært avhengig av parametrene. Hvis man ikke kan leve med disse restriksjonene, finnes det mange andre algoritmer som gjør slik kurvetilpasning. Den ivrige leser kan f.eks. anbefales boken Numerical Recipes. The Art of Scientific Computing, Cambridge Univ. Press, 1992, av W.H.Press et al.

2.2 Eksempel på parameter-tilpasning til bølgeform beskrevet av polynom.

Start opp LabVIEW, og bruk **Open**-kommandoen til å lese inn les inn programmet **LAB1_1.LW**. Programmet er laget med 21 par av (x,y)-verdier. Programmet bestemmer parametrene (polynom-koeffisientene: A, B, C,.....,H) slik at *chi-square* blir minst mulig.

I den øverste grafen vises (x,y)-verdiene samt den best tilpassede rette linje y=Ax+B. χ^2 er indikert, og forteller om avviket i tilpasningen.
I den midterste grafen vises de samme (x,y)-verdiene sammen med den best tilpassede linje beskrevet av et 2.-gradspolynom ($y=Ax^2+Bx+C$). Tilsvarende χ^2 vises her også.
I den nederste grafen vises de samme (x,y)-verdiene sammen med den best tilpassede linje beskrevet av et N^{te}-gradspolynom hvor N<8. χ^2 vises også.

Tilfeldig (normalfordelt) støy kan innblandes i y-verdiene. Standard avviket for amplituden på denne støyen styres av en glide-kontroller på front panelet. En annen glide-kontroller bestemmer hvor lang pause programmet skal vente mellom hver gang det kjøres. N i den nederste tilpasningen kan også varieres.

Oppgave 1.1

- Kjør programmet. Litt "bakgrunnstøy" er lagt inn slik at grafene "lever". De tre grafene fremstilles med de samme rådata, og kurven som tilsvarer de "beste" parameterverdiene for første, annen og N^{te} grads polynom. χ^2 angis også, den indikerer tilpasningens nøyaktighet. Ta en utskrift. Hvilken tilpasning er best? Hvorfor?

- Sett intervallet (perioden) for repetisjon slik at du klarer å følge med på hva som foregår (kanskje 1 sekund), og gjør N mindre. Hva skjer med tilpasningen?
  Finnes det verdier for N hvor den nederste grafen blir lik de to øverste?
  Du kan endre en enkelt y-verdi ved å skrive endringen inn i tabellen til venstre på skjermen. Prøv å øke y-verdi nr. 10 med noen hundre, og se på de ulike tilpasningene.

- Sett støyens normalfordeling til 50. Tilfeldige verdier med standad avvik 50 legges til y-verdiene. Kjør langsomt, så du rekker å skrive ned A og B fra den øverste grafen. Ta 20 kjøringer, og regn ut standard avviket for A og B.
  PS: Standardavvik er gitt ved:
  $$\sigma = \sqrt{\frac{1}{N} * \sum_{i=1}^{N} (x_1 - \bar{x})^2}$$
- Gjør det samme med støyens normalfordeling satt lik 25. Hvordan endrer A og B's standardavvik seg når støyen på rådata halveres?

2.3 Måling av motstanden til gløde-filamentet i en lampe.

Oppgave 1.2

- Mål motstanden til filamentet med digital-voltmeteret (DVMT). Kobl lampen til en strømkilde som kan gi 0 til 12 volt. Bruk meteret på strømkilden til å måle strømmen som går i filamentet når spenningen er 1, 2, 3,...., 10 volt.

- Regn ut *motstand* = spenning / strøm som funksjon av *strøm*, ta med DVMT-verdien som svarer til strøm ≈ 0. Motstanden er som du ser ikke konstant, men øker tilnærmet lineært med strømmen. Ved hjelp av LabVIEW-programmet **LAB1_2.LW** skal du lage en tilpasning til en rett linje.

- Før programmet startes, skriver du inn de utregnede par av (strøm, motstand)-verdier, ialt 11 tallpar (motstanden målt med DVMT svarer til strøm = 0).

- Kjør programmet. Parametrene a0 og a1 regnes ut for modellen
  motstand = a0*strøm + a1.
  Hvilke verdier får du for parametrene ? Stopp programmet, og ta en utskrift av Front Panelet for å vedlegge journalen.

- Vi regner at motstanden til filamentet øker tilnærmet lineært med temperaturen med en temperaturkoeffisient lik 0.005 / °C. Hvor stor blir temperaturen til filamentet når strømmen er 0.15 amp.?

3. Digital Filtrering

Med et signal forstår vi normalt en målbar fysisk størrelse som forandrer seg som funksjon av tiden. Hvis signalet er definert ved alle relevante tidspunkt, kaller vi det et analogt signal. Hvis vi tenker oss at vi måler signalet bare ved visse tidspunkt ("sampler" det) med en analog til digital omvandler, får vi en sekvens med tall som vi kaller et digitalt signal. Vanligvis vil vi måle med konstante samplingsintervall (tidsintervall) svarende til en konstant samplingsfrekvens.

Digital signalbehandling består i å analysere og manipulere det digitale signalet, for eksempel

a) produsere en ny sekvens av tall der frekvensinnholdet er modifisert i forhold til det opprinnelige signalet (digital filtrering).

b) produsere en sekvens av tall som gir uttrykk for frekvensinnholdet i det gitte signalet (diskret fouriertransformasjon).

Hvis tidsintervallet mellom hvert sampel er 1 sek, er samplingsfrekvensen f_S = 1Hz. Den høyeste frekvensen som gjengis korrekt, er gitt ved Nyquist-frekvensen f_S / 2 = 0.5Hz. Høyere frekvenser vil (hvis de ikke er fjernet av et lavpass filter foran AD-konverteren), bli speilet om denne frekvensen ned i lavfrekvensområdet. For eksempel vil en signalfrekvens på 1.25 * 0.5Hz gi et "alias-signal" med frekvens 0.75 * 0.5Hz.

Videre har vi at hvis vi tar N (f.eks. 100) sampler av et analogt signal, og samplingsfrekvensen er f_S (f.eks. 1 Hz), vil vi få med akkurat en hel periode av et sinus signal med frekvens f_S / N = 0.01Hz.

3.1 Ikke-rekursive filter. Utjevning.

Et filter hvor beregningen av de nye verdiene ikke tar hensyn til tidligere beregnede verdier, kalles et ikke-rekursivt filter.

Et vanlig resultat av en måleserie er at det digitale signalet (sekvensen av samplede verdier) inneholder støy som vi ønsker å redusere, i tillegg til det signalet vi er interessert i. Hvis signalet vi ønsker å fremheve er lavfrekvent, kan vi prøve en form for lavpass filter.

En måte er å bruke en form for utjevning eller glatting. Vi lager da et nytt digitalt signal (sekvens av verdier) hvor hver verdi er en eller annen form for midling over noen naboverdier i den opprinnelige sekvensen. En sekvens inneholder N = 512 verdier (sampler). Hvis vi midler over 5 naboverdier, får vi

$$y(n) =(u(n-2) + u(n-1) + u(n) + u(n+1) +u(n+2)) \ / 5$$

for n = 2 til 509 hvor $u(n)$ er en verdi i den opprinnelige sekvensen og $y(n)$ en verdi i den nye. For de to første og de to siste verdiene må vi modifisere formelen, men for enkelhets skyld setter vi disse verdiene lik de gamle, dvs.

$y(0) = u(0)$ $\quad$ $y(1) = u(1)$
$y(510) = u(510)$ $\quad$ $y(511) = u(511)$

Dette er et eksempel på et ikke-rekursivt filter som generelt kan skrives som

$$y(n) = \sum_{k=-K}^{K} c(k) * u(n-k)$$

hvor K angir hvor langt ut til hver side vi går i filtreringen, og $c(k)$ er filterkoeffisientene. I vårt tilfelle er K = 2 og alle koeffisientene 1 / 5. Vi kan se på filteringen som en prosess hvor vi ser på den opprinnelige sekvensen gjennom et vindu (filteret). Etter å ha beregnet den nye verdien, forskyver vi vinduet et step og beregner neste filtrerte verdi, osv.

I stedet for å ta middelverdien av noen punkter (som svarer til tilpasning med en rett linje), brukes det ofte mer kompliserte polynomer. Vi kunne for eksempel bruke minste kvadraters metode og finne en tilpasning til et tredjegrads polynom gjennom fem punkter. I stedet for koeffisientene {1, 1, 1, 1, 1} / 5 (midling over 5 punkter) får vi da {-3, 12, 17, 12, -3} / 35. I litteraturen er det angitt mange forskjellige koeffisient sett for glatting over flere punkter.

Oppgave 1.3

- Programmet **LAB1_3.LW** beregner 280 verdier for et sinus signal med amplitude 1 og med overlagret støy. Støyen har Gauss-fordelt (normal-fordelt) amplitude med middelverdi 0, og standard avvik som kontroleres av Front Panelet. Programmet viser de opprinnelige verdiene og resultatet etter midling over 5 punkter. Kjør programmet med ulike standardavvik, og studer filter-virkningen.

- Angi omtrentlig verdi for reduksjon av støyen. Bruk øyemål ! Det går an å vise at standardavvik for normalfordelt støy minker med en faktor som er lik kvadratroten av summen av kvadratene av filterkoeffisientene. For 5 punkts midling skulle dette gi en faktor på

$$\sqrt{5*(0.2)^2} = \sqrt{5*0.04} = \sqrt{0.2} \approx 0.45.$$

Ser det ut til å stemme ?

3.2 Overføringsfunksjon for et digitalt filter.

For analoge signaler kan vi regne ut en overføringsfunksjon (systemfunksjon) som multiplisert med inngangssignalet gir utgangssignalet. Hvis vi er interessert i frekvensavhengigheten (stasjonære signaler), tegner vi gjerne opp et Bode-plot som viser forsterkning som funksjon av frekvens.

For digitale filter kan vi også finne en overføringsfunksjon som fremstilt grafisk som funksjon av frekvensen, svarer helt til et Bode-plot. Som eksempel kan vi bestemme overførings-funksjonen for et filter som en 5 punkts midling slik:

Vi starter med å se på hva som hender hvis inngangssignalet er en enkel frekvens

$$u(n) = e^{j\omega n}$$

Dette gir

$$y(n) = \left(e^{j\omega(n-2)} + e^{j\omega(n-1)} + e^{j\omega n} + e^{j\omega(n+1)} + e^{j\omega(n+2)}\right)/5$$

$$= e^{j\omega n}\left(e^{-j2\omega} + e^{-j\omega} + 1 + e^{j\omega} + e^{j2\omega}\right)/5$$

$$= e^{j\omega n}\left(2\cos(2\omega) + 2\cos(\omega) + 1\right)/5$$

Inngangsfrekvensen blir altså multiplisert med en faktor som avhenger av frekvensen og filter koeffisientene. Et plot av denne funksjonen viser hvordan hver enkelt frekvens blir forsterket (forminsket) på grunn av filteret. (Vi plotter da gjerne som funksjon av "vanlig" frekvens (Hz) i stedet for vinkelfrekvens (ω).)

I stedet for å utlede et slikt analytisk uttrykk for overføringsfunksjonen kan vi beregne noen utgangsverdier når vi sender inn et signal med bare en frekvens. Forholdet mellom en utgangsverdi og en inngangsverdi med samme nummer i sekvensen, er overførings-funksjonens verdi for denne frekvensen. Ved å gjøre dette for en del frekvenser kan vi tegne opp kurven for funksjonen. Som eksempel kan vi tenke oss at vi sampler et cosinus signal med peak-to-peak amplitude 2 og sampelfrekvens 1 Hz, dvs. 1 sek mellom hvert sampel. Hvis signal frekvensen er 0.25 Hz, får vi 4 sampler pr. periode. Bruker vi for enkelthets skyld bare 3 punkts midling, får vi med sample nr. 0 på bølgens høyeste topp:

Inn	Ut	Forsterkning
u(0) = 1		
u(1) = 0	y(1) = (1+0-1)/3 = 0	0
u(2) = - 1	y(2) = (0-1+0)/3 = -0.33	0.33
u(3) = 0	y(3) = (-1+0+1)/3= 0	0
u(4) = 1	y(4) = (0+1+0)/3 = 0.33	0.33
osv. 0		

Med signalfrekvens 0.5 Hz får vi to sampler per periode og resultatene

Inn	Ut	Forsterkning
u(0) = 1		
u(1) = -1	y(1) = (1-1+1)/ = 0.33	-0.33
u(2) = 1	y(2) = (-1+1-1)/3 = -0.33	-0.33
u(3) = -1	y(3) = (1-1+1)/3 = 0.33	-0.33
osv. 1		

Ved å undersøke flere frekvenser kan en se at funksjonen har ett nullpunkt mellom 0.25 Hz og 0.50 Hz. Nullpunktet for 3 punkts midling ligger ved ca.0.33 Hz. Hvis frekvensen går mot 0, går funksjonen mot 1.

Oppgave 1.4

I programmet **LAB1_4.LW** bestemmer vi verdien for overføringsfunksjonen for 5 punkts midling ved forskjellige frekvenser ved å bruke inngangssignalet

*u(i)=cos(i*j*2*π/100) for i = 0 til 99.*

Hvis j = 1 inneholder sekvensen på 100 punkter en hel periode, hvis j = 2 blir det to perioder osv. Vi kan normalisere ved å tenke oss at tidsintervallet mellom to sampler alltid er en tidsenhet (f.eks. ett sekund). (Hvis det ikke er det, kan vi alltid transformere.) med j = 1 får vi en hel periode i løpet av 100 tidsenheter, dvs. signalfrekvensen er 0.01 Hz. Med j=50 får vi 50 perioder, dvs. frekvens 0.5 Hz. Dette

er Nyquist-frekvensen som er den høyeste frekvensen det er aktuelt å måle forsterkningen for.

Programmet regner ut forsterkningen for verdiene:

j	=1	dvs. frekvens 0.01
	=2	0.02
	=3	0.03
	.	
	.	
	=50	0.50

og viser resultatene grafisk.

Som vi så i regneeksemplet for 3-punkts midling, er det ikke nødvendig å regne ut alle punktene. Programmet stopper ved hver frekvens på i = 20, og som forsterkning brukes y(9)/u(9) .

- Kjør programmet. Det tegnes opp en kurve over forsterkningen som funksjon av frekvensen (filterets overføringsfunksjon). Med programmets opprinnelige innstillinger fås 5-punkt midling. På Front Panelet kan du endre antallet punkter hvert enkelt punkt utregnes fra. Det maksimale antallet er 9, da må alle koeffisientene fra -4 til +4 være ulik null. Hvis bare koeffisientene fra -3 til +3 er utfylt, mens -4 og +4 er null, vil vi ha 7 punkter med i utregningen. Hvis bare koeffisientene fra -2 til +2 er utfylt, mens -4 til -3 og +3 til +4 er null, vil vi ha med 5 punkter.

- Når frekvensen øker, faller overføringsfunksjonen mot 0. Hvordan er dette fallet avhengig av antall punkter det midles over ? Ved høye frekvenser får vi alltid svingninger omkring nulllinjen. Det samme gjelder for koeffisienter som svarer til LSQ-tilpasning med polynomer av høyere grad.

Oppgave 1.5

Det går an å modifisere koeffisientene slik at de oppfyller kravet at et signal med frekvens lik Nyquist-frekvensen blir helt undertrykket. Dette reduserer svingningene omkring nullinjen ganske godt.

- Modifiser Front Panelets koeffisienter i **LAB1_4.LW** slik at du bruker 5 punkts filtrering med koeffisientene 0.0729, 0.25, 0.354, 0.25, 0.0729 for C(-2), C(-1). C(0), C(1) og C(2) .
- Ta en utskrift av overføringsfunksjonen. Hvordan er dette filteret ulik 5-punkts midling i oppgave 1.4 ?

4. Rekursive filter.

Et filter hvor de nye verdiene beregnes etter en formel

$$y(n) = \sum_{k=0}^{K} c(k) * u(n-k) + \sum_{k=0}^{M} d(k) * y(n-k)$$

kalles et rekursivt filter fordi vi ved beregningen av *y(n)* også tar med tidligere beregnede verdier (*y(n-1), y(n-2)* osv.). Filteret har derfor en slags hukommelse, dvs. utgangsverdiene påvirkes av tidligere beregnede verdier. Vi skal se at det digitale filteret som svarer til et 1. ordens analogt lavpass filter, er et slikt rekursivt filter.

4.1 Digitalt lavpassfilter.

For å finne formelen for et digitalt lavpass filter starter vi med differensial ligningen for et 1. ordens analogt lavpass filter:

$$v_2(t) + R * C * \frac{dv_2(t)}{dt} = v_1(t)$$

med $v_1(t)$ inngangssignal, $v_2(t)$ utgangssignal og $R*C$ tidskonstanten T. På samme måte som i lab.oppg.1 bruker vi tilnærmelsen

$$\frac{dv_2}{dt} = \left(v_2(n) - v_2(n-1)\right) / S$$

dvs. den deriverte settes lik forskjellen mellom to nabo sampler dividert med sampel intervallet (tidsintervallet). Dette gir differens ligningen

$$v_2(n) = b * v_2(n-1) + (1-b) * v_1(n)$$

med

$$b = (T / S) / (1 + T / S)$$

hvor $T = R*C$ er filterets tidskonstant og S er sampel intervall (tidsintervall). Dette er et rekursivt filter, dvs. utgangsverdien avhenger både av inngangsverdi og utgangsverdi, og er altså den digitale ekvivalenten til et 1. ordens analogt lavpass filter. Forholdet mellom tidskonstanten T og sampel intervallet S bestemmer hvordan det digitale filteret virker.

For det analoge filteret vil en liten tidskonstant T svare til en høy verdi for grensefrekvensen. Tilsvarende ser vi for det digitale filteret at en liten verdi for *T/S* gir en liten verdi for b og liten virkning av det digitale filteret. For b gjelder at $0 < b < 1$. Med $b = 0$ får vi ingen filtrering, den nye sekvensen er identisk med den opprinnelige.

Grensesekvens for et filter defineres vanligvis ved -3dB punktet, dvs at amplituden er redusert til ca 0.707 av max verdi. Dette kalles også "Turn Over Frequency" f_0, og er gitt ved $f_0 = 1/(2\pi RC)$

$T = R*C$, slik at man kan beregne T når grensefrekvens er oppgitt.

Oppgave 1.6

Programmet **LAB1_5.LW** er nokså likt LAB1_3.LW, men bruker den digitale ekvivalenten til et 1. ordens analogt lavpass filter. Som inngangssekvens bruker vi 280 samplede verdier av et sinus signal med støy. Vi tenker oss at sampel frekvensen er 1 Hz, dvs. $S = 1$ sekund. Sekvensen dekker akkurat 2 perioder av sinus-signalet.

- Hvilken frekvens har signalet ?

Programmet regner ut sekvensen y(n) for n = 1 til 280 (y(n) er satt lik 0 før start) for oppgitt verdi av T / S som er forholdet mellom tidskonstant og sample intervall

- Hvilken verdi av T svarer til at filterets grensefrekvens blir lik sinus signalets frekvens ?

- Kjør programmet med denne verdien,og f.eks. 10 ganger større og mindre verdi, og se hvordan signalet ut av filteret blir forandret. Ved å fjerne innlagt støy (glidekontroll på Front Panelet) kan du lettere se hvordan forsterkningen varierer.

4.2 Beregning av responsen på grunnlag av filterets impulsrespons.

En digital enhetsimpuls er definert som sekvensen

$$\text{delta(n)} = \delta(n) = 1 \text{ for } n = 0$$
$$= 0 \text{ for } n \neq 0.$$

Dette blir en sekvens av '0', med en enslig '1' der hvor n=0.
Vi vil bruke denne sekvensen som inngangssignal i differensligningen

$v_2(n) = a^* v_1(n) + b^* v_2(n\text{-}1)$, hvor $a = 1/(1 + T/S)$ og $b = (T/S)/(1 + T/S)$,

altså som inngangsignal til det rekursive digitalfilteret vi nettopp har anvendt.

Med

$$v_1(n) = 1 \text{ for } n = 0$$
$$= 0 \text{ for } n \neq 0,$$

får vi sekvensen

$v_2(0) = a$
$v_2(1) = a * b$
$v_2(2) = a * b * b$
$v_2(3) = a * b * b * b$
osv.

Siden denne sekvensen er filterets utgang ved en impuls som inngang, kalles sekvensen for filterets impulsrespons.

Da b er positiv og mindre enn 1, er dette en sekvens av verdier som avtar eksponentielt mot 0. Dess mindre b er, dess mindre er tidskonstanten i forhold til sampel intervallet og dess fortere avtar verdiene mot 0.

Dette viser hvordan et 1. ordens lavpassfilter reagerer på en enhets impuls. Filterets **impuls respons** betegnes i det følgende med

$\{h(n)\}$ (obs.: sekvens av verdier).

Vi skal nå vise at hvis vi kjenner systemets impuls respons, kan vi regne ut hvordan det reagerer på et vilkårlig inngangssignal.

Et generelt digitalt signal (dvs. en sekvens av verdier) kan skrives som en sum av enhets impulser med forskjellig vekt og forsinkelse.
Eksempel:

$$x(n) = a_0\ \delta(n) + a_1\ \delta(n-1) + a_2\ \delta(n-2) + a_3\ \delta(n-3).$$

En vilkårlig sekvens kan derfor skrives som

$$x(n) = \sum_{k=0}^{\infty} x(k) * \delta(n-k)$$

Hvis vi har et system (f.eks et filter) som er lineært og skift invariant gjelder følgende:

Hvis: $\{\delta(n)\}$ gir responsen $\{h(n)\}$

vil en forsinket enhetsimpuls $\{\delta(n-k)\}$ gi responsen $\{h(n-k)\}$.

Tilsvarende gjelder for en vilkårlig sekvens:

Hvis	$x(n)$	gir responsen	$y(n)$
vil	$x(n-k)$	gi responsen	$y(n-k)$.

Hvis transformasjonen av $x(n)$ fra $y(n)$ skrives symbolsk som

$$y(n) = T[x(n)]$$

får vi derfor

$$y(n) = T\left[\sum_{k=0}^{n} x(k) * \delta(n-k)\right] = \sum_{k=0}^{n} x(k) * T[\delta(n-k)]$$

dvs.

$$y(n) = \sum_{k=0}^{n} x(k) * h(n-k)$$

Et lineært skift invariant system er derfor fullstendig karakterisert ved {*h(n)* }, responsen til en enhets impuls. Responsen er gitt ved *folding (convolution)* av inngangssekvensen *x(n)* med systemets impuls respons (sekvensen) {*h(n)* } . Ligningen ovenfor kalles *convolution sum* . Ordet *folding* kommer av at vi ved utregningen kan tenke oss de to sekvensene skrevet på to linjer, den ene under den andre. Vi tar så og folder den ene mot venstre om origo. Sekvensen som er foldet forskyves så mot høyre ett trinn av gangen. I hver posisjon skal vi så summere produktene av de to samplene som står rett overfor hverandre for å beregne *y(n)* .

Eksempel: Beregne y(2)

```
                x(0)   x(1)   x(2)   x(3).....
.....h(3)       h(2)   h(1)   h(0)
```

y(2) = x(0)*h(2) +x(1)*h(1)+x(2)*h(0).

Foldingssummen kan også skrives som

$$y(n) = \sum_{k=0}^{n} h(k) * x(n-k)$$

Det blir jo det samme hvilken sekvens som foldes.

Eksempel:
Inngangssekvens (1 1 1 0 0 0 0 0 0) inn på filteret med *T/S* = 4, dvs. *b* =0.8 .

v_1(n)	h(n)	respons
1	0.2	0.2
1	0.16	0.2 + 0.16 = 0.36
1	0.128	0.2 + 0.16 + 0.128 = 0.488
0	0.1024	0.16 + 0.128 + 0.1024 = 0.3904
0	0.0819	0.3127
0	0.0655	0.2502
0	0.0524	0.1998
0	0.0419	0.1598
0	0.0336	0.1279

Dette svarer til et analog lavpass filter påtrykt en firkantpuls.

Hvis systemet ikke svarer til et vanlig filter, men har en impuls respons som f.eks. er gitt ved sekvensen 111000000 , vil inngangssignalet ovenfor gi sekvensen 123210000 , dvs. en trekantet form.

For lange sekvenser blir det mye regnearbeid. Det er en nær sammenheng mellom Fourier transformasjon og "folding".

"Folding" er et viktig begrep i alle slags målinger. Hvis det ikke er periodiske forløp (funksjoner),må en være forsiktig når en tolker resultatene.For å unngå randeffekter kan det være nødvendig å forlenge sekvensene med nuller. Alle målinger påvirkes av måleapparaturen, og de målte verdiene er de opprinnelige verdiene "foldet" med responsfunksjonen til måleapparaturen. En optisk spektrograf eller en magnetisk

spektrograf for ladede partikler vil for eksempel gi en utsmøring, en fordeling som skyldes spalt åpninger osv. Tilsvarende gjelder for en detektor for ladede partikler. Selv om partiklene har en veldefinert energi, vil vi måle en Gaussformet fordeling. I prinsippet kan vi da gjøre to ting. Vi kan enten starte med en eller annen modell ligning for inngangsverdiene med justerbare parametre, folde den med responsfunksjonen og tilpasse parametrene slik at vi får best mulig overensstemmelse med målingene.

Den andre metoden er å starte med målingene og prøve å fjerne innflytelsen av apparaturen ved den omvendte prosessen av folding (*unfolding,deconvolution*). Hva som er den beste fremgangsmåten, vil variere.

Program som er laget for folding, kan som regel benyttes for den inverse prosessen også.

Oppgave 1.7 :

I denne oppgaven skal vi illustrere teorien ovenfor som altså sier at utgangssekvensen (responsen) til et filter er inngangssekvensen "foldet" med impulsresponsen til filteret. Impulsresponsen spiller altså samme rolle i det digitale tilfellet som overføringsfunksjonen i Laplace teorien i det analoge tilfelle.

Som eksempel bruker vi det digitale 1. ordens filteret. Vi viste tidligere at det var lett å regne ut impulsresponsen til dette filteret. Det ble en sekvens som starter med en verdi og avtar eksponentielt mot null: *a, a * b, a * b * b,* osv. hvor

$$a = \frac{1}{1 + T/S} \quad \text{og} \quad b = \frac{T/S}{1 + T/S} = 1 - a$$

Programmet **LAB1_6.LW** genererer først et sinus signal med støy, 256 sampler. Dette signal vises frem grafisk.

Deretter genereres impulsresponsen for den verdien for T/S som du velger. Bruk verdier for T/S som du brukte i oppgave 1.6. Impulsresponsen vises også grafisk på skjermen.

- Hva slags forløp har kurven for impulsresponsen ?

De to sample sekvensene foldes sammen, og svaret vises grafisk i samme fremstilling som den opprinnelige sinuskurven. Svaret er korrigert med en faktor (roten av antall verdier) .

- Ser resultatet ut til å stemme med det du så i oppg. 1.6 ?

- Hvordan skal det bli hvis du bruker en svært liten verdi for T/S ?