EKSAMENSOPPGAVE

Emne: **IRE31613 Signalbehandling**

Lærer/telefon: Per Thomas Huth/90 95 56 59

Grupper:	**Dato:**	**Tid:**
12ELE-D, 12ELEY	**4. desember 2014**	09.00 – 13.00
Antall oppgavesider: 5 Inklusiv forside.	**Antall vedleggsider:** 2	
Sensurfrist: 9. Januar 2015		

Hjelpemidler:

Ifeachor & Jervis (2002):
Digital Signal Processing - A Practical Approach - 2. ed. Pearson Prentice Hall.

James H. McClellan et al (2003):
Signal Processing First. . Pearson Prentice Hall. (Totalt eller kopi av utdrag)

Artikler etc.:

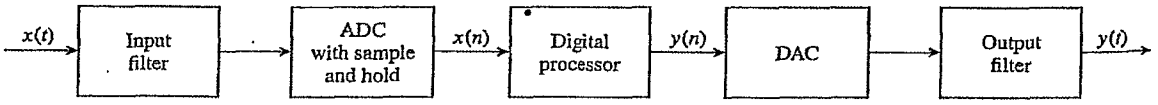
- Overview of Information Theory (chapt. 16)
- UNIK-kompendium (302)

Kalkulator, tegne og skrivesaker.

KANDIDATEN MÅ SELV KONTROLLERE AT OPPGAVESETTET ER FULLSTENDIG. Alle deloppgaver teller likt ved bedømming.

Oppgave 1

Figuren under viser prosessen hvor et analogt system digitaliseres for siden å bli behandlet digitalt.



a) Hvorfor blir det analoge signalet filtrert før punktprøving (Sampling) blir utført? Bruk et ideelt lavpassfilter med grensefrekvens 5kHz på inngangen. Tegn frekvensspekteret (frekvensresponsen) i området <-15kHz,+15kHz> for en punktprøvingsfrekvens på 12 kHz.

b) Tegn det samme for en punktprøvingsfrekvens på 8kHz. Hva skjer? Kommenter.

Isteden for det idealiserte filteret benyttes et filter med Butterworth karakteristikk:

$$A(f) = \frac{1}{\sqrt{1 + \left(\frac{f}{f_c}\right)^{2n}}}$$

Her er ***n*** ordenen på filteret. Vi har et 4. ordens filter med en grensefrekvens, $\mathbf{f_c}$ på 10kHz. Punktprøvingsfrekvensen er i utgangspunktet satt til 40kHz.

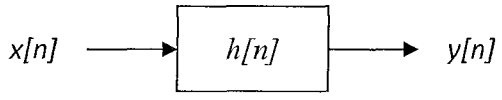
c) Finn Aliasing feilen ved grensefrekvensen.

d) Finn Aliasing feilen ved Foldingsfrekvensen. (Nyquist-frekvensen)

e) Finn minste punktprøvingsfrekvens som er nødvendig for at Signal til aliasing feilnivået skal være mindre enn 10 % ved grensefrekvensen.

f) Hva blir foldingsfrekvensen og aliasing feilen ved foldingsfrekvensen nå?

Vi tenker oss at den digitale prosessen som skal utføres er et LTI system på diskret form som vist i figuren under. Følgende er gitt: h[n] = {0,2,2,0,..} og x[n] = {0,1,1,0,0..}.



g) Finn y[n] ved hjelp av formelen som definerer foldning. Plott y[n].

h) Vis hvordan du kan løse samme foldning grafisk (Flip & Drag).

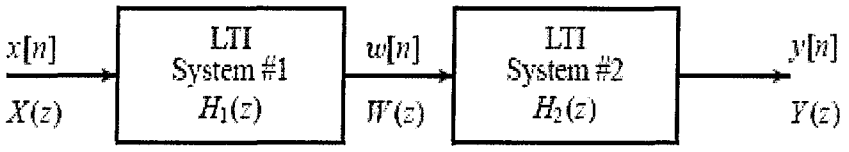
i) Finn et generelt utrykk som løser utrykket δ[n-a]*δ[n-b]. Svaret skal begrunnes eller utledes.

j) Bruk i) til å beregne y[n] direkte når h[n] = δ[n-1] + δ[n-2] + δ[n-3] og x[n] = δ[n-2] + δ[n-3].

Vi bytter ut prosessen med en LTI prosess som er vist under. System 1 har følgende output:

$w[n] = (x[n] + x[n-1] + x[n-2])/3.$

k) Hva slags krets (Funksjon) er dette? Hva blir $H_1(z)$?



La $H_2(z) = H_1(z)$

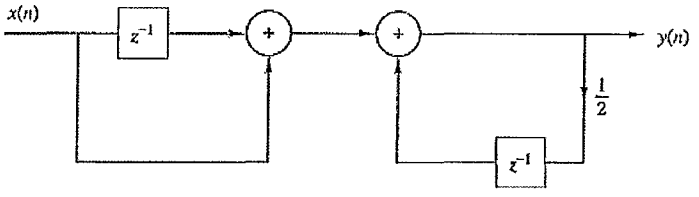
l) Finn y(n).

m) Finn poler og nullpunkt i H(z), for totalsystemet, og plott dem. Er totalsystemet stabilt?

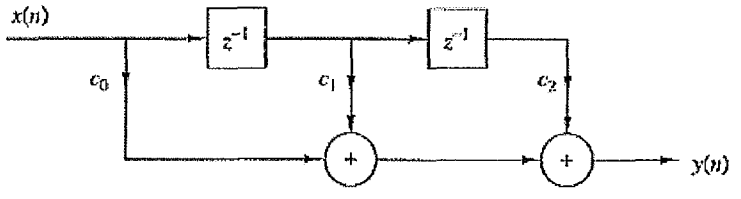
n) Finn et utrykk for $H(e^{j\hat{\omega}})$.

o) Plott $H(e^{j\hat{\omega}})$ med tallverdi (magnitude) for området $-\pi < \hat{\omega} < +\pi$.

Vi ser videre på følgende systemer:



p) Finn y(n) for systemet skissert over. Hva slags system er dette?



q) Finn y(n) for systemet skissert over. Hva slags system er dette?

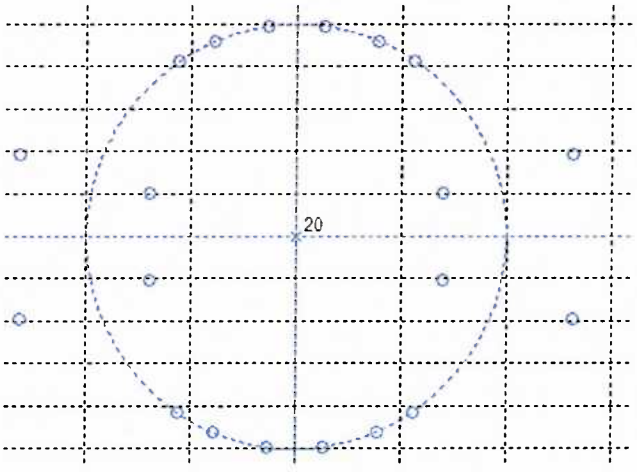
r) Lag en ny og tilsvarende skisse som beskriver **totalsystemet** foran [LTI filteret behandlet i oppgave k) til o)] og sett verdi på koeffisientene.

Oppgave 2

a) Et digitalt filter har et Pole/Zero-plott som vist i figur 2A. Beskriv filteret ut fra denne lista:

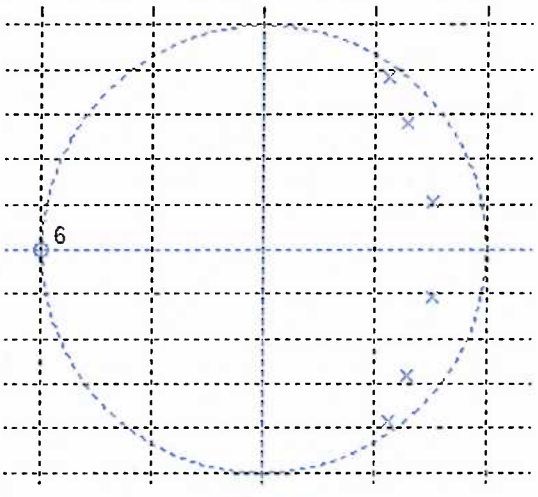
- Er dette FIR eller IIR filter?
- Er filtertypen lavpass, høypass, båndpass eller båndstopp?
- Er filteret stabilt?

Grunngi svarene



Figur 2A

b) Et annet digitalt filter har et Pole/Zero-plott som vist i figur 2B.
Beskriv også dette filteret ut fra lista i forrige punkt. Grunngi svarene.



Figur 2B

I **vedlegg 1**, finner du en komplett programliste for et filterdemoprogram som skal kunne kjøres på en ***mbed*** LPC1768 mikrokontrollermodul. Programmet importerer headerfiler for digitale filtre som er konstruert vha. MATLAB/FDATOOL, for å kunne realisere et spesifikt filter.

c) Forklar hva du må forandre i programmet for å kunne kjøre filteret med en samplingshastighet på 16 kHz.

d) Anta at du måler med oscilloskop direkte (uten analogt rekonstruksjonsfilter) på den analoge utgangen, som representerer resultatet av filtreringen. Anta videre at inputsignalet er et 2 kHz sinus-signal med en amplitude på 1 V og med en offset på 1.65 V. Skisser hvordan utsignalet vil se ut i forhold til innsignalet. Gå ut fra at den aktuelle frekvensen er i filterets passbånd.
Hint! Sjekk forholdet mellom samplingsfrekvens og signalfrekvens før du tegner.

e) Du ønsker med en skop-probe å måle hvor lang tid hele prosessen med DA-omforming + AD-omforming + filterberegning tar for ett sample. Vis hva du må gjøre i programmet ditt for å få til dette.

f) Anta at du ved målinger finner at AD-omforming i dette enkle oppsettet tar 40 mikrosekunder. Tilsvarende for DA-omformeren er 10 mikrosekunder. Beregningen av en gitt filteralgoritme tar 100 mikrosekunder. Hva blir i dette tilfellet høyeste teoretiske samplingshastighet, hvis man legger inn en sikkerhetsmargin på 10 %.

Vedlegg 1

Programlister for filterdemo

Fil: Filterdemo.cpp

```
#include "mbed.h"
#include "FIR.h"
#include "lp_fir_ca1_5_av_fs.h"

AnalogOut ana_out(p18); // DAC
AnalogIn ana_in(p16);   // ADC
Ticker io_cntrl;
volatile float osig = 0.0; // Signal sent to DAC
volatile float isig = 0.0; // Signal read from ADC
FirConfig Cfg;

void output(void);          // Send signal DAC
void inout(void);           // Ticker funksjon
void input(void);           // Les fra ADC
void compute(void);         // Utfør DSP-algoritme

int main()
{
    // Set up filter from FDATOOL-exported headerfile
    fir_init(BL, &B[0], &Cfg);
    io_cntrl.attach_us(&inout, 125);

    while(1) {
        /* Do Nothing */
    }
}

void inout(void)
{
    output();
    input();
    compute();
}

void input(void)
{
    isig = ana_in - 0.5;
}

void output(void)
{
    ana_out = osig + 0.5; // scaling and offset
}

void compute()
{
    osig = fir(isig, &Cfg);
}
```

Fil: FIR.h

```
#include <stdio.h>
#include <stdlib.h>

typedef struct
{
    const float *h;  // Peker til impulsrespons
    float *dly;       // Peker til forsinkelseslinje
    int N;              // Filterlengde
} FirConfig;

void fir_init(short N, const float *h, FirConfig *Cfg);
float fir(float inp,  FirConfig *Cfg);
```

Fil: FIR.cpp

```
#include "FIR.h"

void fir_init(short N, const float *h, FirConfig *Cfg)
{
    short i;
    Cfg->h    = h;
    Cfg->N    = N;
    Cfg->dly  = static_cast <float *> (malloc(N * sizeof(float)));
    if (Cfg->dly == NULL) {
        printf("FIR Buffer allocation FAILED!\n");
        exit(1);
    }
    else {
        printf("FIR Buffer allocation OK!\n");
        printf("... Filling buffer with zeros\n");
        for (i = 0; i < N; i++) {
            Cfg->dly[i] = 0.0;
        }
    }
}

float fir(float inp,  FirConfig *Cfg)
//implements FIR - with newest sample in dly[0]
{
    float *dly = Cfg->dly;
    const float *h = Cfg->h;
    const int N = Cfg->N;
    short i;
    float yn = 0;

    dly[0] = inp;                 //newest sample x(n) @top of buffer
    for (i = N-1; i > 0; i--)     //loop
    {
        yn += dly[i] * h[i];      //y(n)=x[n-i]*h[i]
        dly[i] = dly[i-1];        //move data down to update delays
    }
    yn += dly[0] * h[0];          //y(n)=x(n)*h(0)
    return yn;                    //return y(n) at time n
}
```