

MALMÖ HÖGSKOLA

Inbyggda system och signaler

Digital signalbehandling

Labuppgift 4

1504d

Namn 1 Leonard Holgersson

Namn 2 Hadi Deknache

Tommy Andersson
februari 2017

Design och jämförelse av FIR- och IIR-filter med hjälp av Matlab

Introduktion

I denna uppgift skall vi jämföra egenskaper hos FIR-filter och IIR-filter. För att bestämma filterkoefficienterna kommer Matlab att användas. Vi skall dels jämföra behovet av beräkningshastighet för att realisera specifika filterkrav avseende amplituddämpning, dels jämföra fasvridningens inverkan på kurvformen av en signal.

Programkod för FIR-filter

I labuppgift 3 (1504c) skrev du kod för FIR-filter. Vi skall använda den koden även i denna uppgift (använd varianten med *static float* för *xbuff[]* och *b[]*).

Programkod för IIR-filter

Utsignalen från ett IIR-filter ges av

$$y[n] = \sum_{k=0}^M b_k \cdot x[n-k] + \sum_{l=1}^N a_l \cdot y[n-l]$$

Första delen av högerledet kan ses som utsignalen från ett FIR-filter. Om vi vill göra kodandet så enkelt som möjligt kan vi använda koden för FIR-filtret och bilda och addera summan i andra delen av högerledet till utvärdet från FIR-filterdelen (dvs. Direktform I). Men se till att du har kvar FIR-filterkoden.

Skriv den nya delen av koden på samma sätt som för FIR-delen så att filtrets längd N lätt kan varieras t.ex. genom att använda **#define** N

Definiera en array för de fördröjda värdena av utsignalen, $y[]$, och en annan för koefficienterna $a[]$. Det är litet klurigt med indexen. Den första koefficienten är ju a_1 men alla arrayer i C börjar ju med index 0. Enklast är att placera a_1 i $a[1]$ och föregående utsignal i $y[1]$. Det betyder att $a[0]$ och $y[0]$ inte används men det gör ju inget, vi har gott om RAM. Båda arrayerna skall därför ha längden $N+1$.

Använd datatypen *static float*. Initiera arrayerna med 0 respektive aktuella värden för koefficienterna.

Bilda nu summan (andra delen av högerledet) och addera värdet från FIR-delen. Detta är den nya utsignalen. Denna skall nu sparas i $y[1]$ (inför nästa sampling), men först måste det ges plats genom att alla värden y -värden flyttas ett steg.

När allt är klart konverteras utvärdet till typ *uint32_t* och läggs i *outvalue*.

Test av IIR-koden

Testa att koden verkar fungera genom att prova den med filtret:

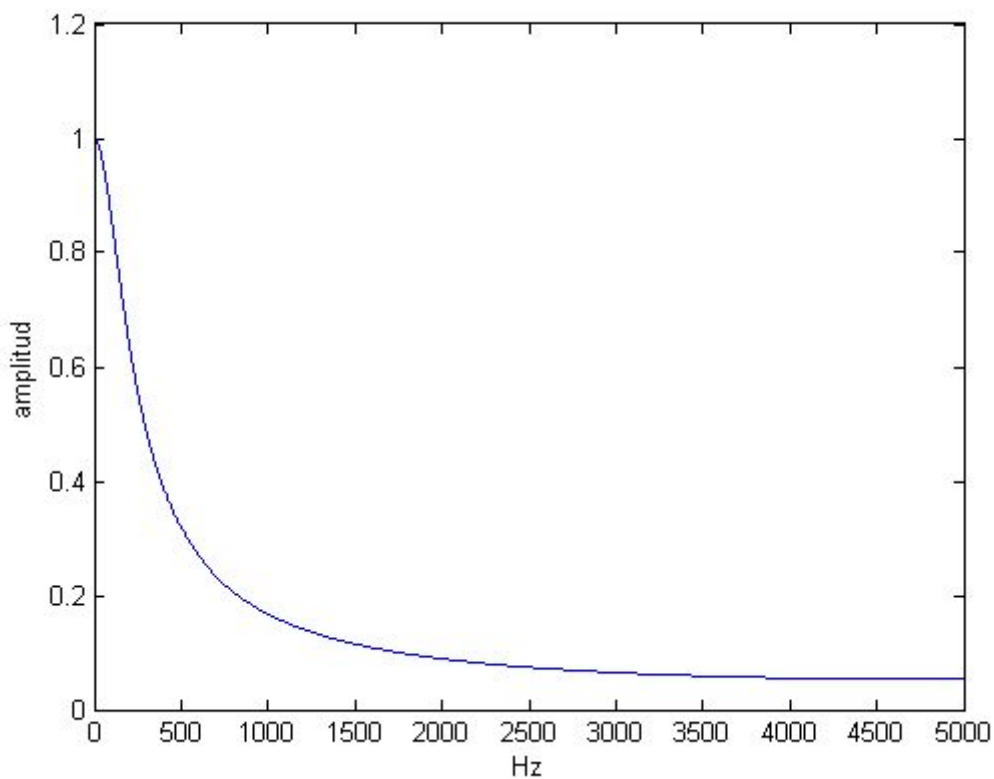
$$y[n] = 0.1 \cdot x(n) + 0.9 \cdot y[n-1]$$

För att kunna göra detta behöver vi känna $|H(e^{j\hat{\omega}})|$.

Frekvensfunktionen för ett IIR-filter ges av

$$H(e^{j\hat{\omega}}) = \frac{\sum_{k=0}^M b_k \cdot e^{-j\hat{\omega}k}}{1 - \sum_{l=1}^N a_l \cdot e^{-j\hat{\omega}l}}$$

Använd Matlab för att för att plotta $|H(e^{j\hat{\omega}})|$ (jämför med labuppgift 3) och **klistra in plotten här** (använd f som variabel och tänk på du kan behöva använda $./$ vid divisionen):



Använd vanliga mätuppställningen, kör koden, variera frekvensen och övertyga dig om att din kod fungerar.

Redovisa dina mätningar:

Frekvens [Hz]	Utsignalens toppvärde [V]
100	0,827
300	0,483
500	0,311
800	0,209
1000	0,169
2000	0,094
3000	0,079

När du gjort det klistrar du in C-koden för det kompletta IIR-filtret (dvs. den kod du själv skrivit):

```
void TC0_Handler(void)
{
    static float xbuff[M+1] = {0};
    static float b[M+1]={0.1};

    static float ybuff[N+1]= {0};
    static float a[N+1]= {0, 0.9};

    volatile uint32_t ul_dummy;
    uint32_t inval, outval;
    float value=0;

    /* Clear status bit to acknowledge interrupt */
    ul_dummy = tc_get_status(TC0, 0);          //The compare bit is cleared by
reading the register, manual p. 915

    /* Avoid compiler warning */
    UNUSED(ul_dummy);

    ioport_set_pin_level(CHECK_PIN,HIGH);      //put test pin HIGH

    adc_start(ADC);
    while((adc_get_status(ADC) & 0x1<<24)==0); //Wait until DRDY get high

    inval=adc_get_latest_value(ADC);           //get input value

    //-----FIR-filter koden här-----
    for (uint32_t k=M;k>0;k--)
    {
        xbuff[k]=xbuff[k-1];
    }

    xbuff[0]=(float)inval;
```

```

for (uint32_t x=0;x<M+1;x++)
{
    value+=b[x] * xbuff[x];
}

//outvalue = (uint32_t)(value);
//-----

//-----IIR-filter koden här-----

for (uint32_t x=1;x<=N+1;x++)
{
    value+=a[x] * ybuff[x-1];
}

for (uint32_t k=N-1;k>=1;k--)
{
    ybuff[k]=ybuff[k-1];
}

ybuff[0]=value;

outvalue = (uint32_t)(value);

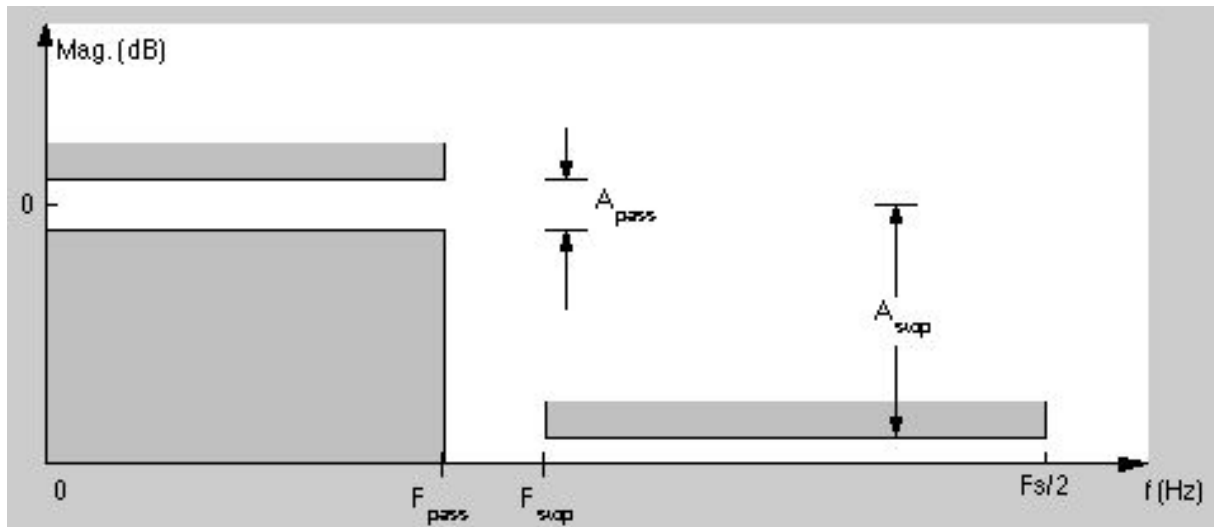
dacc_write_conversion_data(DACC,outvalue);    //send output value to DAC

ioport_set_pin_level(CHECK_PIN,LOW);          //put test pin LOW
}

```

Filterspecifikation

När man designar ett filter utgår man i regel från en kravspecifikation som innebär att filtret skall uppfylla vissa minimikrav. Figuren nedan visar hur en sådan specifikation kan se ut för ett lågpassfilter.



Kravet är amplitudfunktionen (|frekvensfunktionen|) skall rymmas inom "korridoren" i figuren där A_{pass} , A_{stop} , F_{pass} och F_{stop} är specificerade.

Figuren är hämtad från *fdatool* som är ett hjälpmedel i Matlabs Signal Toolbox för filterdesign.

I denna laboration vill vi designa lågpasfilter med följande kravspecifikation:

$A_{\text{pass}} = 1 \text{ dB}$, $A_{\text{stop}} = 40 \text{ dB}$, $F_{\text{pass}} = 750 \text{ Hz}$, $F_{\text{stop}} = 1250 \text{ Hz}$, sampelfrekvens 10000 Hz

$A_{\text{stop}} = 40 \text{ dB}$ betyder att dämpningen är 40 dB dvs förstärkningen är -40 dB.

Vi skall se vad det innebär i "vanliga" siffror.

Inom passbandet får amplituden variera högst $\pm 0.5 \text{ dB}$.

Vad innebär det i %?
(redovisa beräkningarna)

$$0,5 \text{ dB} = 20 \cdot \log \frac{a}{b} \quad \frac{a}{b} = ? \quad \frac{0,5}{20} = \log \frac{a}{b}$$

$$10^{(0,5/20)} = a/b \Rightarrow a/b \approx 1,06 \text{ ger } 6\%$$

$$-0,5 \text{ dB} = 20 \cdot \log a/b \quad \frac{a}{b} = ? \quad \frac{-0,5}{20} = \log \frac{a}{b}$$

$$10^{(-0,5/20)} = a/b \Rightarrow a/b \approx 0,94 \text{ vilket ger } -6\%$$

I stoppbandet skall amplituden ha minskat med 40 dB

Det innebär att den har minskat till% av insignalen.
(redovisa beräkningarna)

$$-40 \text{ dB} = 20 \cdot \log(a/b) \rightarrow \frac{-40}{20} = \log a/b$$

$$10^{(-40/20)} = 0,01 \text{ dvs } 1\%$$

Vi skall använda dels ett FIR-filter av sk equirippletyp (innebär att "ripple" är lika stort i hela stoppbandet) dels ett IIR-filter av typ "Elliptic". Detta är den typ av IIR-filter som ger den brantaste lutningen för ett givet ordningstal på filtret (och har mest olinjär fas...).

FIR-filter - design

Starta Matlab och ge kommandot *fdatool*.

Ställ in kravspecifikationen enligt ovan. Sätt *Filter order* till *Minimum* och *Density Factor* till 20 (Density Factor är ett mått på räknenoggrannheten när Matlab iterativt beräknar

filterkoefficienterna).

Klicka på *Design Filter*.



Undersök nu vilken information du kan få ut om filtret genom att klicka på symbolerna

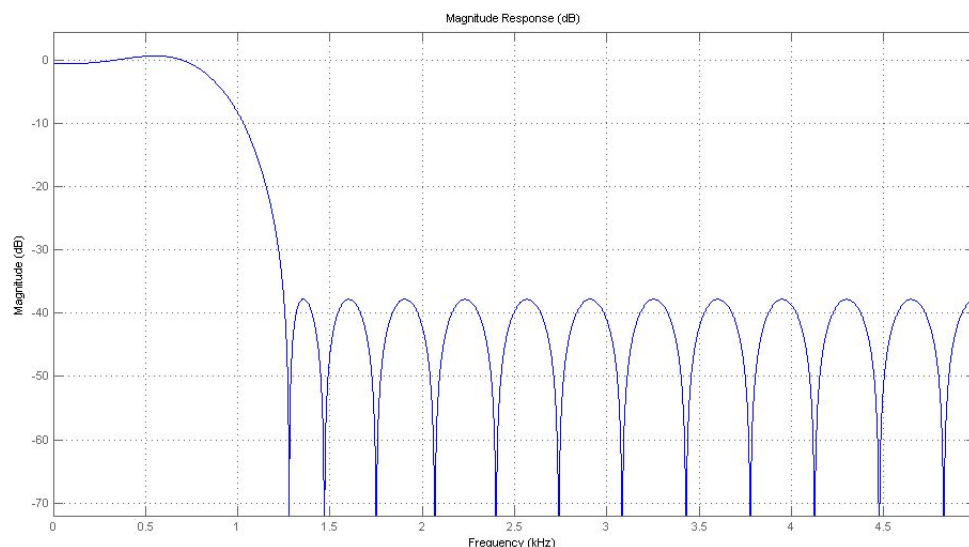


Märkligt nog händer det att *fdatool* inte alltid bestämmer filterkoefficienterna så kravspecifikationen blir helt uppfylld. Vi blundar för det i detta fall och nöjer oss med resultatet vi får...

Vi vill förstås dels kunna hämta koefficienterna b_k för att stoppa in dem i koden men vi vill även kunna kopiera den beräknade amplitudfunktionen så vi kan jämföra med vad det blir i verkligheten. Även fasfunktionen, fördröjningen och pol-nollställeplaceringen är intressanta att spara.


Amplitudfunktion

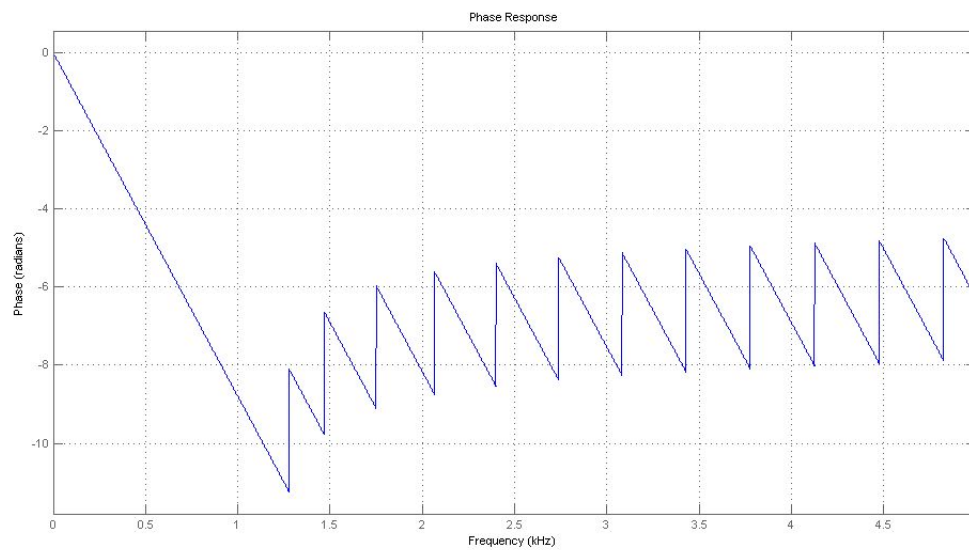
Enklast är nog att klicka på symbolen . Det öppnas nu ett nytt fönster. Välj  (Alternativt kan man gå in under menyn *Analysis* och välja *Magnitude response*). I detta läge kan man om man vill redigera plotten men vi nöjer oss med den som den ser ut. Välj nu *File/Export...* Spara filen t.ex. i jpeg-format i hemkatalogen med lämpligt namn och fyll tillägg. **Hämta filen till detta word-dokument och sätt in den här (*Insert/Picture/From file...*):**



Tyvärr lite krångligt...

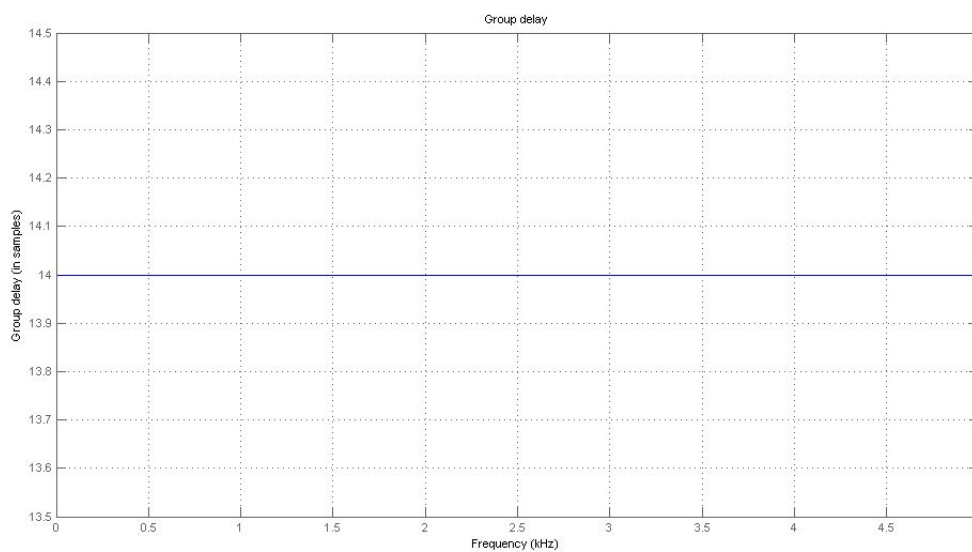
Fasfunktion

Gör på motsvarande sätt för fasfunktionen (symbolen ) och **sätt in diagrammet här**



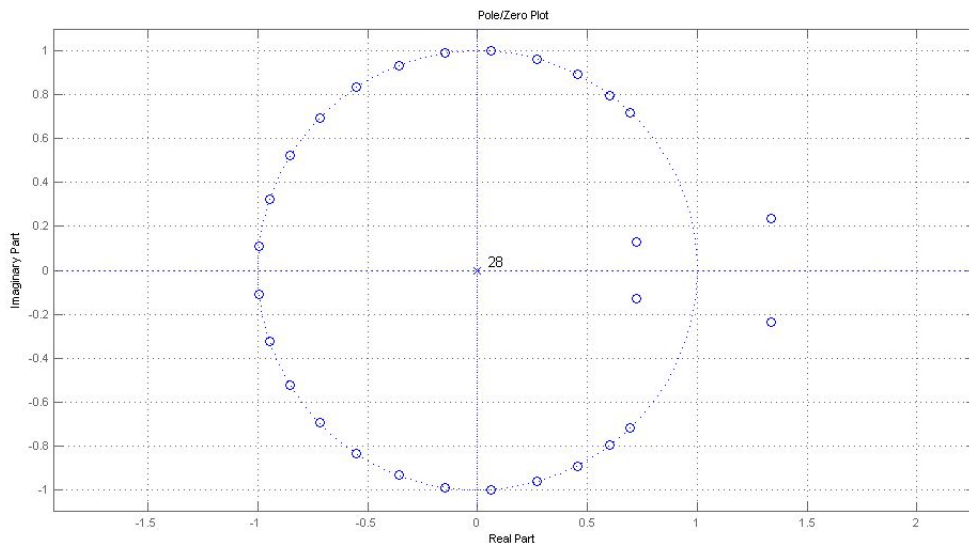
Fördröjning

Samma sak för fördröjningen (symbolen ) i filtret



Pol-nollställeplacering

Och här sätts pol-nollställeplaceringen (symbolen ) in



Filterkoefficienterna

Gå tillbaka till grundfönstret i *fdatool*. Välj *Targets/Generate C-header...* Här kan man välja i vilket format man vill ha koefficienterna. Eftersom vi tänker använda float i beräkningarna exporterar vi enklast i formatet double precision floating point. Välj ett filnamn och spara i hemkatalogen.

Välj nu *Insert/File...* i Word och sätt in dokumentet här:

```

/*
 * Filter Coefficients (C Source) generated by the Filter Design and
Analysis Tool
 *
 * Generated by MATLAB(R) 8.0 and the Signal Processing Toolbox 6.18.
 *
 * Generated on: 15-Feb-2017 12:25:45
 *
 */

/*
 * Discrete-Time FIR Filter (real)
 * -----
 * Filter Structure   : Direct-Form FIR
 * Filter Length      : 29
 * Stable             : Yes
 * Linear Phase       : Yes (Type 1)
 */

/* General type conversion for MATLAB generated C-code */
#include "tmwtypes.h"
/*
 * Expected path to tmwtypes.h
 * C:\Program Files\MATLAB\R2012b\extern\include\tmwtypes.h
 */

```

```

const int BL = 29;
const real64_T B[29] = {
    0.01080096024942, 0.009150882313605,
    0.007511904549456, 0.0005792030769843,
    -0.01127376170725, -0.02515190942132, -0.03590095692545,
    -0.03739762488425,
    -0.02453046490484, 0.00471963858602, 0.04788555515624,
    0.09797523269544,
    0.1449637668957, 0.1784338566514, 0.1905580972352,
    0.1784338566514,
    0.1449637668957, 0.09797523269544, 0.04788555515624,
    0.00471963858602,
    -0.02453046490484, -0.03739762488425, -0.03590095692545,
    -0.02515190942132,
    -0.01127376170725, 0.0005792030769843, 0.007511904549456,
    0.009150882313605,
    0.01080096024942
};

```

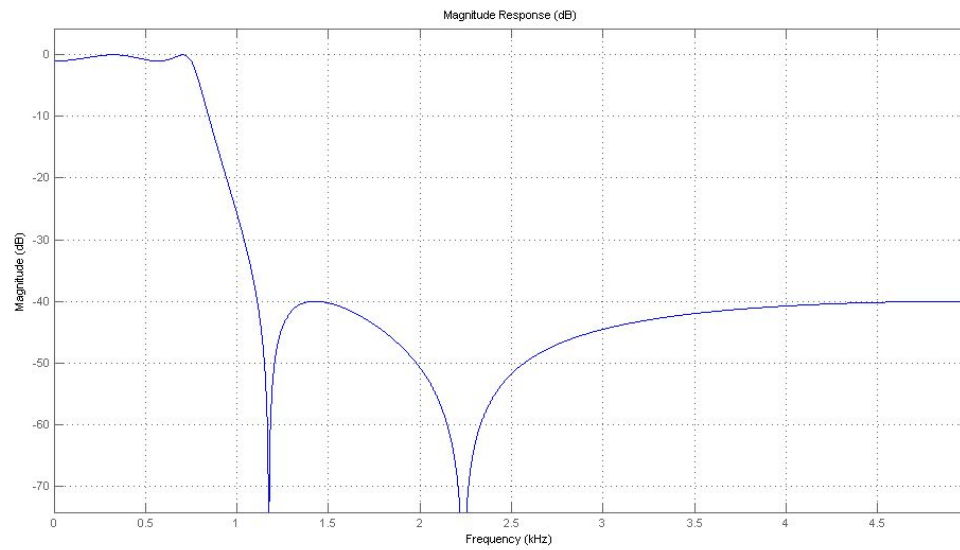
IIR-filter - design

Nu gör vi motsvarande för IIR-filter. Kolla att kravspecifikation fortfarande stämmer i *fdatoool*. Klicka för IIR-filter och välj *Elliptic* på rullgardinsmenyn. Kör *Design Filter*. Som framgår i rutan upp till vänster väljer Matlab i första hand att beräkna koefficienterna för kaskadkoppling av 2:a ordningens sektioner. Vi vill göra kodningen enkel och vill bara ha en sektion. Välj därför *Edit/Convert to single section*. Matlab anger Direktform II, vi tänker använda Direktform I men det gör ingenting, koefficienterna har samma värden i dessa två former.

Använd nu samma teknik som vid FIR-filterdesignen för att hämta amplitudfunktion, fasfunktion, fördröjning, pol-nollställeplacering och koefficienterna.

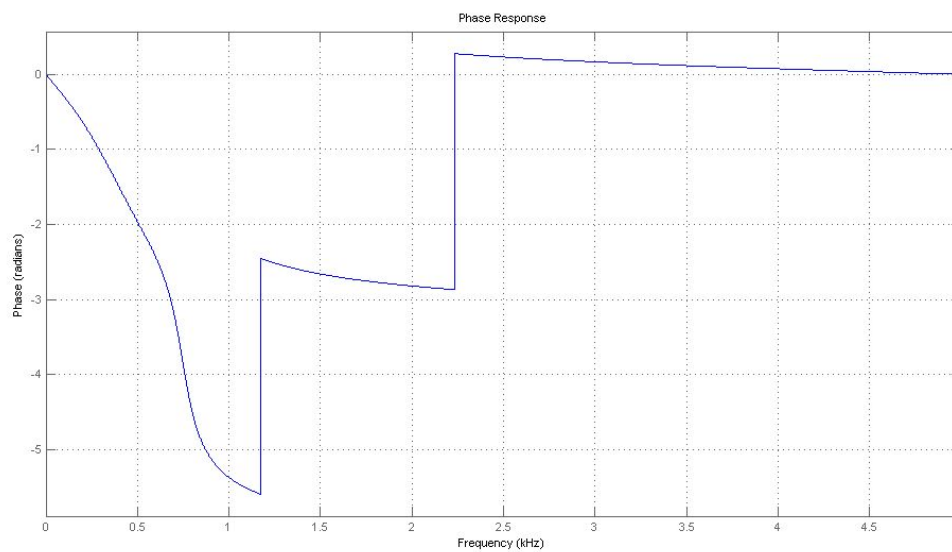
Amplitudfunktion

Sätt in figur



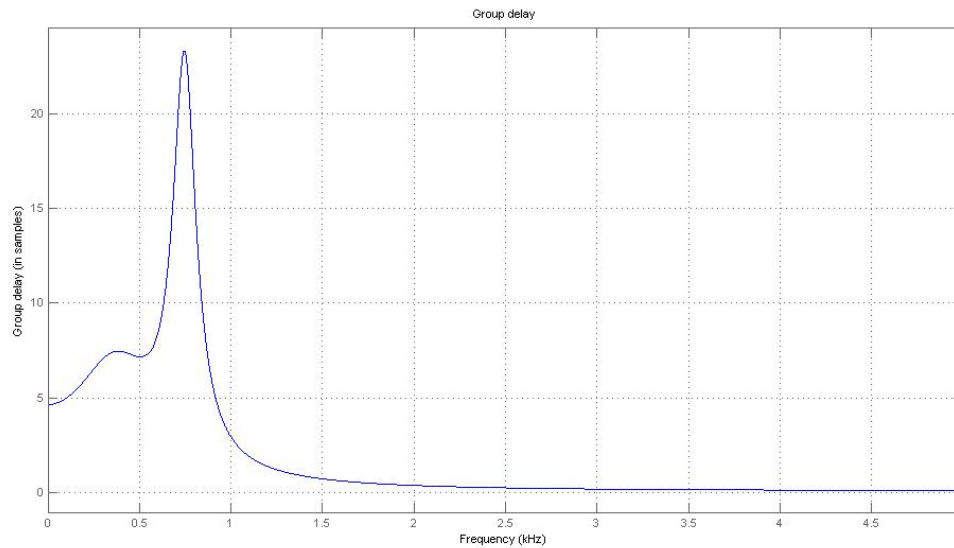
Fasfunktion

Sätt in figur



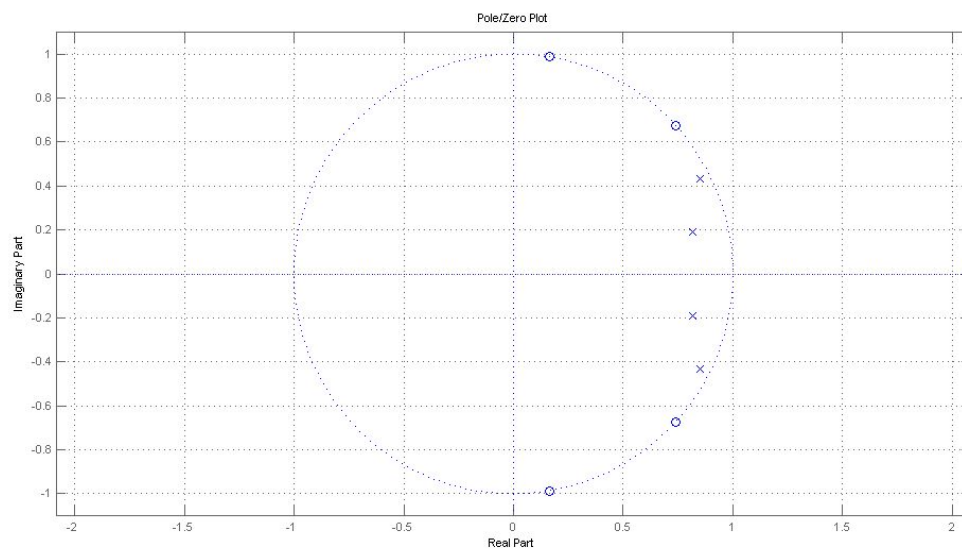
Fördröjning

Sätt in figur



Pol-nollställeplacering

Sätt in figur



Filterkoefficienter

Sätt in textdokument

```

/*
 * Filter Coefficients (C Source) generated by the Filter Design and
Analysis Tool
 *
 * Generated by MATLAB(R) 8.0 and the Signal Processing Toolbox 6.18.
 *
 * Generated on: 15-Feb-2017 14:20:10
 *
 */

```

```

/*
 * Discrete-Time IIR Filter (real)
 * -----
 * Filter Structure      : Direct-Form II
 * Numerator Length     : 5
 * Denominator Length   : 5
 * Stable                : Yes
 * Linear Phase         : No
 */

/* General type conversion for MATLAB generated C-code */
#include "tmwtypes.h"
/*
 * Expected path to tmwtypes.h
 * C:\Program Files\MATLAB\R2012b\extern\include\tmwtypes.h
 */
const int NL = 5;
const real64_T NUM[5] = {
    0.01488697472657, -0.02695899404537, 0.03705935223574,
    -0.02695899404537,
    0.01488697472657
};
const int DL = 5;
const real64_T DEN[5] = {
    1, -3.338693232847, 4.401916486793,
    -2.691625646031,
    0.6428936122854
};

```

Test av filterna med sinusformad signal

Nu är det dags att testa om filterna fungerar enligt kravspecifikationen och även att undersöka beräkningstiden.

Använd samma uppkoppling och samma inspanning som i tidigare labbar.

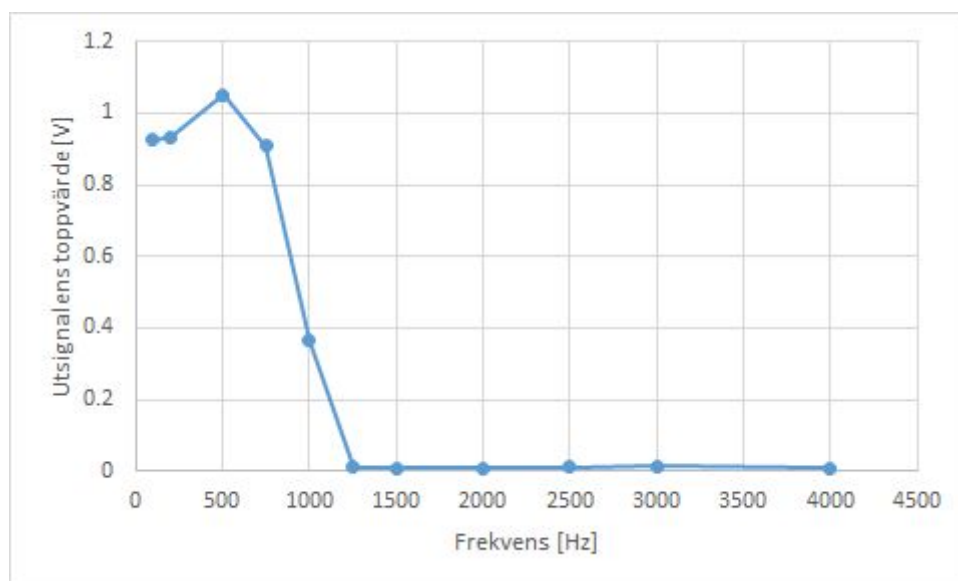
FIR-filter

Kopiera koefficienterna från den insatta texten ovan (FIR-filter design - filterkoefficienter) i initieringen av $b[]$ i programkoden och skriv in rätt värde på M . Ladda programmet och testa att kravspecifikationen verkar uppfylld genom att variera frekvensen över det aktuella området.

Redovisa med tabell med mätvärden och diagram (missa inte att mäta vid de kritiska frekvenserna i kravspecifikationen)

<i>Frekvens (Hz)</i>	<i>Amplitud (V)</i>
100	0.926
200	0.933
500	1.05

750	0.909
1000	0.365
1250	0.012
1500	0.009
2000	0.009
2500	0.011
3000	0.013
4000	0.009



Kommentar:

Mätningarna av plotten ser bra ut och stämmer någorlunda för FIR-filtret. Kollar man på plottarna med matlab är dem någorlunda lika.

Anslut nu oscilloskopets ena kanal till testutgången Digital Pin 22 som ju ställs hög under beräkningstiden och mät upp pulsbredden som alltså svarar mot beräkningstiden.

Beräkningstid: 77.64µs

IIR-filter

Koefficienterna från den insatta texten ovan i avsnittet IIR-design – Filterkoefficienter består av två uppsättningar som svarar mot täljaren (Numerator) och nämnaren (Denominator) i överföringsfunktionen för IIR-filtret. Koefficienterna i täljaren är b_k -koefficienterna och används för initieringen av $b[]$ i programkoden.

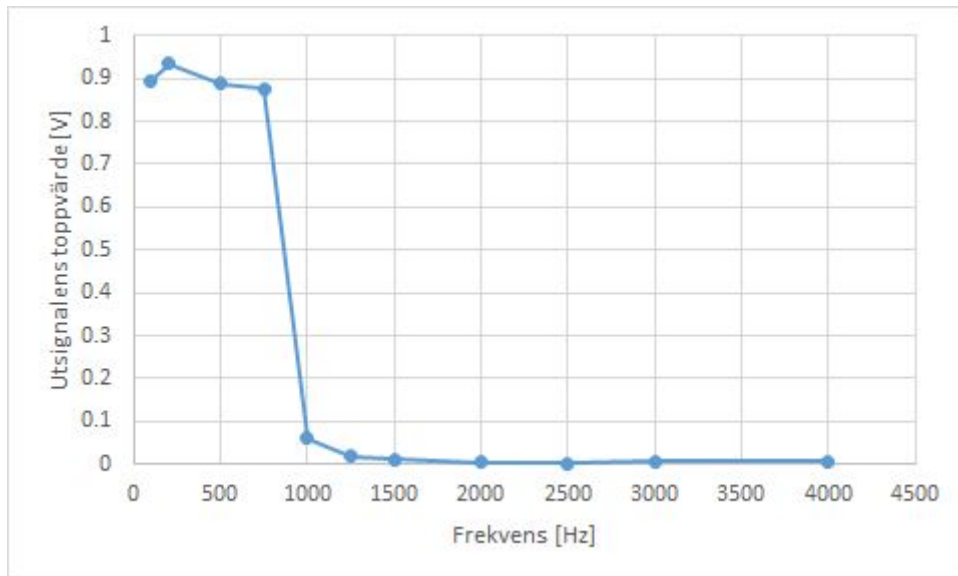
Nämnarkoefficienterna börjar med en etta, denna behöver vi inte. Enklast är dock att använda

hela uttrycket för att initiera $a[]$. Ettan hamnar ju då i $a[0]$ som vi ju inte använder. Dock måste man **byta tecken på a -koefficienterna**. Definitionen är nämligen inte helt lika i Matlab med den som vi (och kursboken) använder (kolla gärna helpfunktionen i Matlab...).

Ladda nu programmet och testa att kravspecifikationen verkar uppfyllt genom att variera frekvensen över det aktuella området.

Redovisa med tabell med mätvärden och diagram (missa inte att mäta vid de kritiska frekvenserna i kravspecifikationen)

<i>Frekvens</i>	<i>Amplitud</i>
100	0.895
200	0.934
500	0.887
750	0.877
1000	0.06
1250	0.02
1500	0.011
2000	0.005
2500	0.003
3000	0.007
4000	0.008



Kommentar:

Mätningarna av plotten ser bra ut och stämmer någorlunda för IIR filter och dess egenskaper. Kollar man på plottarna med matlab är dem någorlunda lika.

Anslut nu oscilloskopets ena kanal till testutgången Digital Pin 22 och mät upp beräkningstiden.

Beräkningstid: 26.46μs

Test av filterna med icke-sinusformad signal

Om allt fungerat som det skall uppfyller både FIR- och IIR-filterna kravspecifikationen som vi satte upp.

Beroende på situationen kan det emellertid också finnas andra krav.

Om man vill ta bort någon störning från t.ex. en EKG-signal är det viktigt att den ursprungliga kurvformen återskapas, det räcker inte med att frekvensinnehållet är rätt.

Vi har ingen störd EKG-signal att testa på och det blir lite krångligt med uppkopplingen om vi skulle vilja simulera något liknande.

Insignal

Vi väljer i stället att arbeta med en fyrkantsignal som vår signalgenerator kan alstra. Alla periodiska signaler kan ju skrivas som en summa av sinusformade signaler, en fourierserie.

En fyrkantsignal med periodtiden T och grundfrekvensen $f = \frac{1}{T}$ kan skrivas som

$$k \cdot \left(\cos(2\pi f \cdot t) - \frac{1}{3} \cos(2\pi \cdot 3f \cdot t) + \frac{1}{5} \cos(2\pi \cdot 5f \cdot t) - \frac{1}{7} \cos(2\pi \cdot 7f \cdot t) \dots \right) \text{ där } k \text{ beror av}$$

signalens amplitud.

För en fyrkantsignal med nivå ± 1 V är $k = \frac{4}{\pi}$.

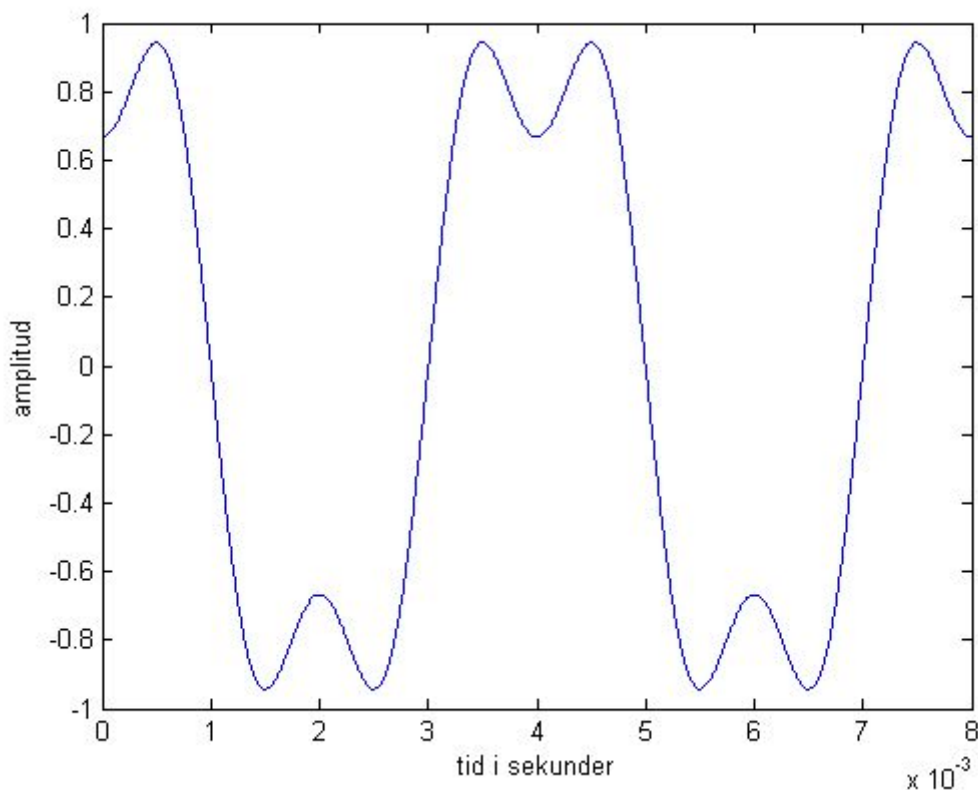
Låt oss nu låtsas att de två första termerna i summan dvs $k \cdot \left(\cos(2\pi f \cdot t) - \frac{1}{3} \cos(2\pi \cdot 3f \cdot t) \right)$ är den ursprungliga signalen och att resten $k \cdot \left(\frac{1}{5} \cos(2\pi \cdot 5f \cdot t) - \frac{1}{7} \cos(2\pi \cdot 7f \cdot t) + \dots \right)$ är en störning som vi vill filtera bort.

Med $f=250$ Hz svarar kravspecifikationen för våra filter ovan just mot detta!

Hur ser då den ”ursprungliga” signalen ut?

Använd Matlab för att plotta signalen $1 \cdot \left(\cos(2\pi f \cdot t) - \frac{1}{3} \cos(2\pi \cdot 3f \cdot t) \right)$ så du ser ett par perioder.

Kopiera och klistra in här:

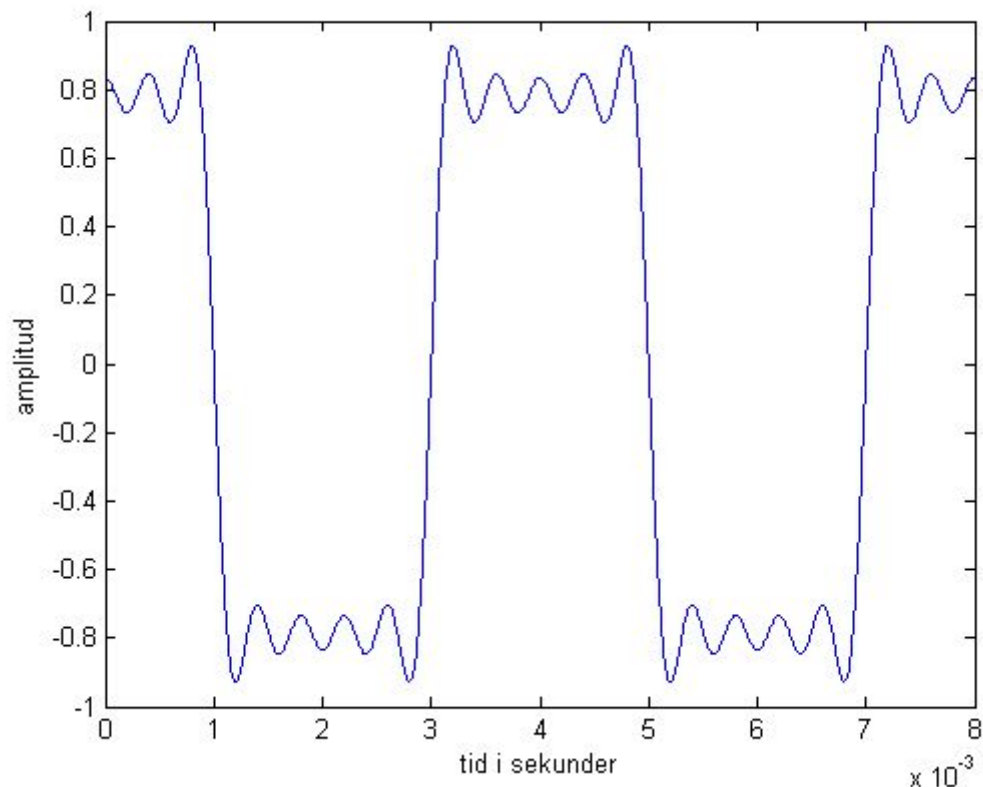


När vi har filtrerat bort frekvenserna över ca 1000 Hz är det alltså denna kurvform vi hoppas få fram.

För att övertyga dig om att en fyrkantsignal verkligen har fourierserien ovan så gör även en

plot med fler övertoner medtagna. Lägg först till $\frac{1}{5}\cos(2\pi \cdot 5f \cdot t)$, sen $-\frac{1}{7}\cos(2\pi \cdot 7f \cdot t)$, sen $\frac{1}{9}\cos(2\pi \cdot 9f \cdot t)$. Som du ser börjar signalen allt mer att likna en fyrkantsignal.

Klistra in den sista plotten här



Mätningar

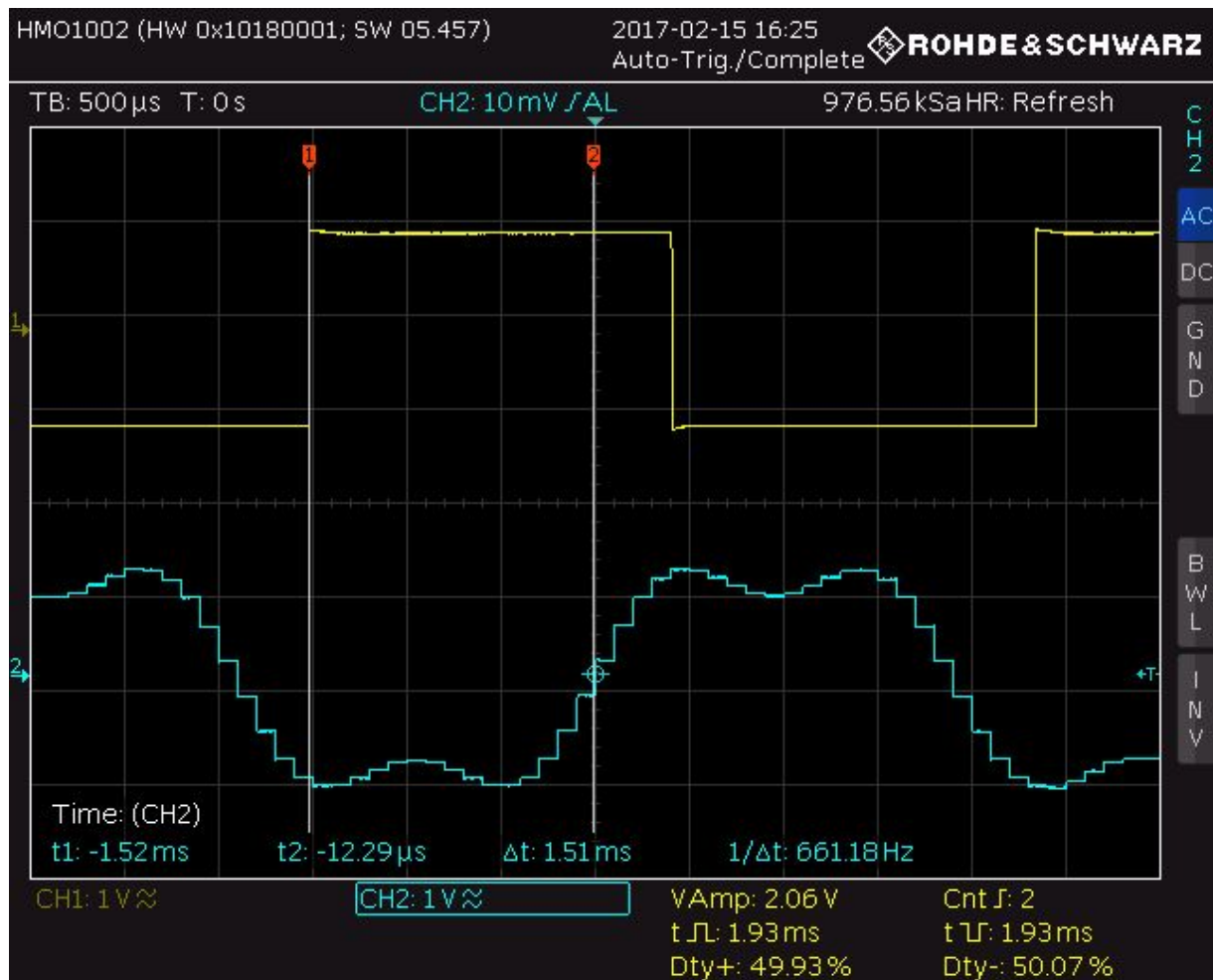
Ställ nu in signalgeneratoren på fyrkantsignal och 250 Hz.

Ställ amplituden så att signalen säkert hamnar inom området för AD-omvandlaren, låt det vara lite marginal.

FIR-filter

Kör FIR-filterprogrammet och studera insignalen och den resulterande utsignalen från DA-omvandlaren. Om det inte ser ut som du förväntat prova att minska amplituden lite på insignalen.

Fota med mobilen och klistra in bilden men de två signalerna (insignalen överst) Se till man kan se oscilloskopinställningen (dvs tid/ruta och spänning/ruta) på fotot.

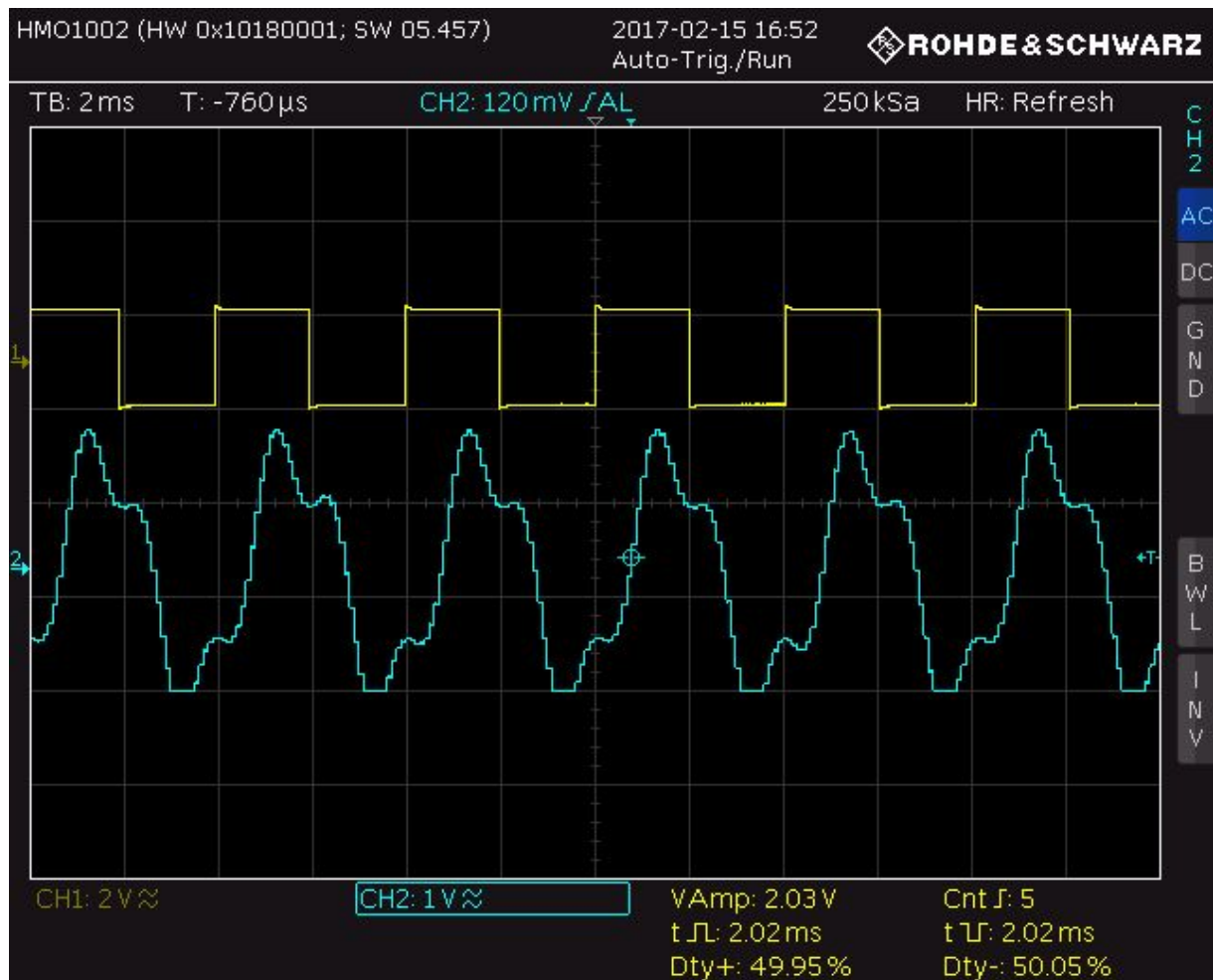


Hur stämmer utsignalen med den "ursprungliga" signalen som du plottat med Matlab ovan? Hur är det med fördröjningen? (mät tiden mellan uppåtgående flank på fyrkantsignalen och mittpunkten på den uppåtgående flanken på utsignalen) Hur många sampel svarar fördröjningen mot? (Ett sampel är ju $1/10000 \text{ s} = 100 \text{ μs}$). Jämför med Matlab, dvs. diagrammet för fördröjning som du klistrat in ovan:

Fördröjningen stämmer någorlunda överens med diagrammet från matlab som vi skapade innan. Med en fördröjning på $14 \cdot 100 \text{ μs}$ ger det 1.4 ms dvs ungefär vad vi fick på 1.51 ms , det är en liten skillnad, men i närheten. Det svarar mot 15 samplingar istället för de 14 vi i teorin skulle få. Varför den skiljer sig, är för att i Matlab får vi en fördröjning vilket syftar på ett system som sker omedelbart medan i vårt fall har vi en extra sampling pga att vi har en extra fördröjning på 77.6 μs vilket ger 1 sample extra.

IIR-filter

Kör IIR-filterprogrammet och studera insignalen och utsignalen från kopplingen. Fota med mobilen och klistra in bilden men de två signalerna (insignalen överst) Se till man kan se oscilloskopinställningen (dvs tid/ruta och spänning/ruta).



Hur stämmer utsignalen med den ”ursprungliga” signalen som du plottat med Matlab ovan?

Kan man säga något om fördröjningen?:

Singalen stämmer egentligen inte överens med det vi plottat i matlab och detta är eftersom fördröjningen sker olika vid olika signaler vilket inte kan få exakt samma signal som vi plottat ut. IIR filtret har en fördröjning som är olika och därför skiljer sig från FIR filtret.

Ett IIR-filter av elliptisk typ har väldigt olinjär fasgång (jämför diagrammen ovan från fdatoool) och är alltså olämpligt i denna situation. Andra IIR-filtertyper är mer linjära men inte lika branta vilket innebär att det blir fler koefficienter och längre beräkningstid.

Test med EKG-liknande signal

Ladda ner filen ”ekg.mp3” från kursens Its learningsida till din mobil. Den ligger i samma mapp som labhandledningen. Om du har en iphone kan du t.ex. använda appen ”MP3 Music Downloader Free”. Filen som är gjord i Matlab innehåller en EKG-liknande signal på ena kanalen (förutom att kurvformen är fel så är frekvensområdet helt fel men passar bra till vårt filter...). På andra kanalen finns samma signal med ett överlagrat brus som ligger i frekvensområdet 2,5 – 4 kHz.

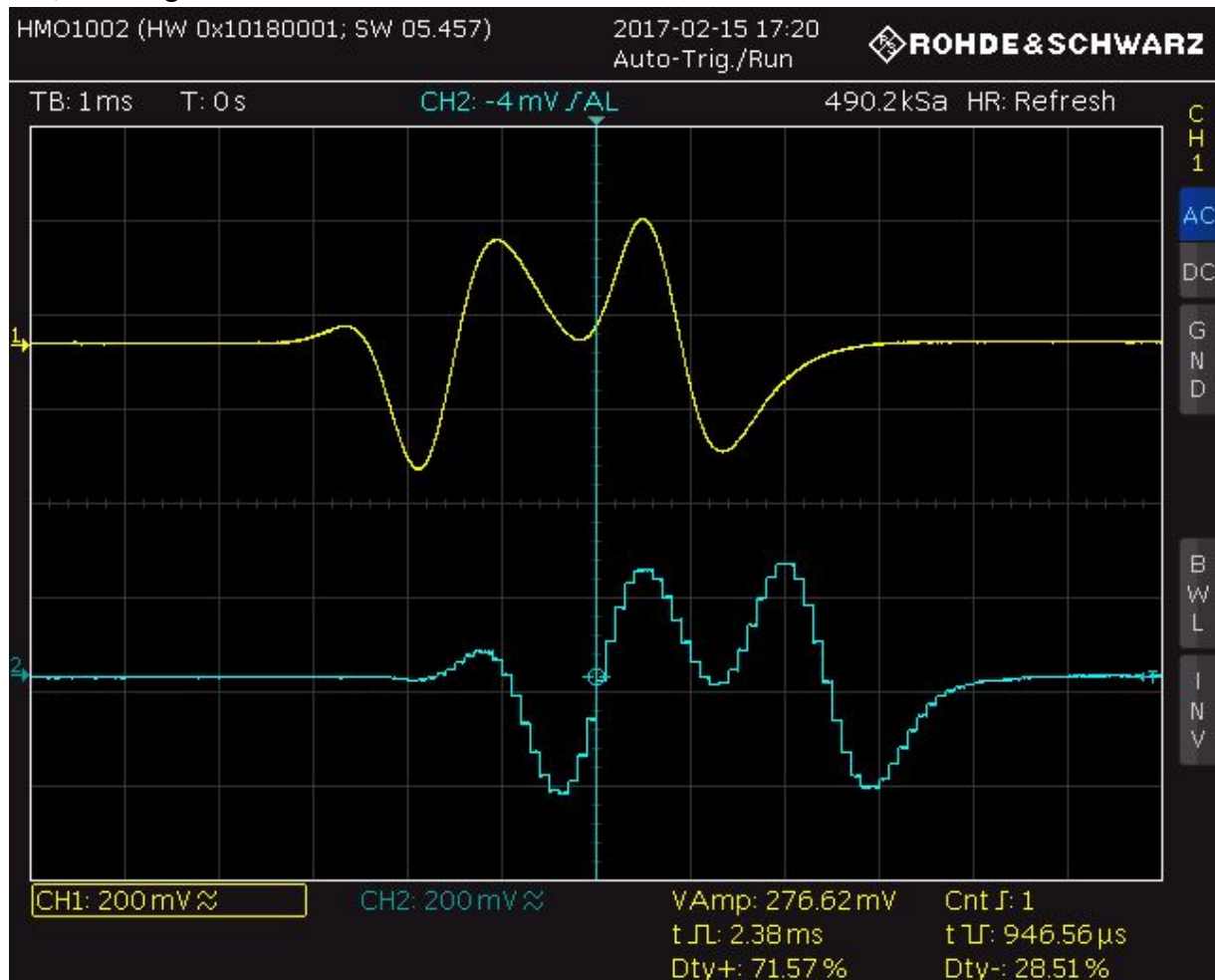
Anslut mobilen till ingången i stället för signalgeneratoren. Ställ in lämplig volym så du

utnyttjar filtret maximalt.

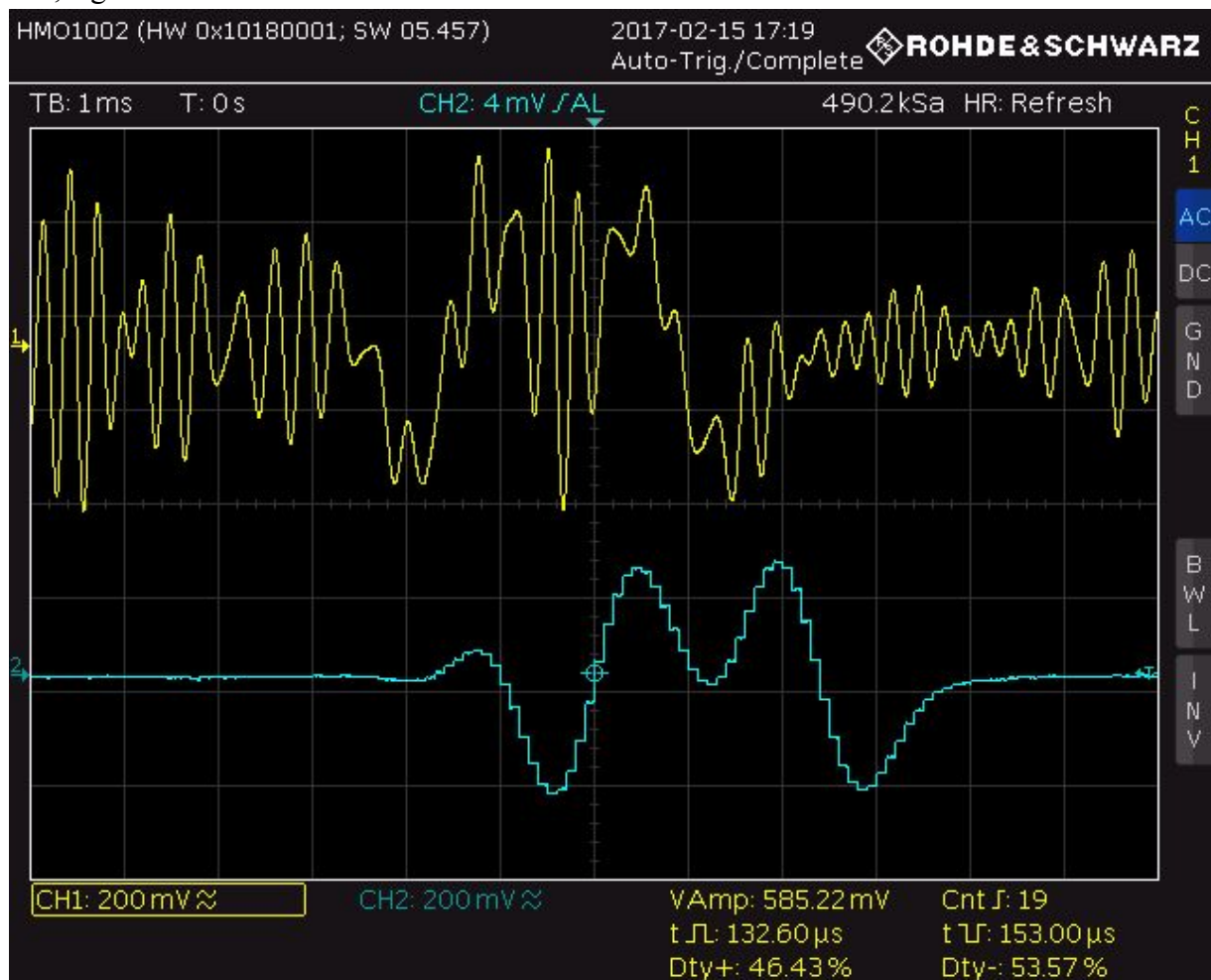
Fota några representativa bilder av oscilloskopskärmen där man jämföra hur signalen ser ut utan brus, med brus och FIR- respektive IIR-filternas påverkan på signalen.

Klistra in:

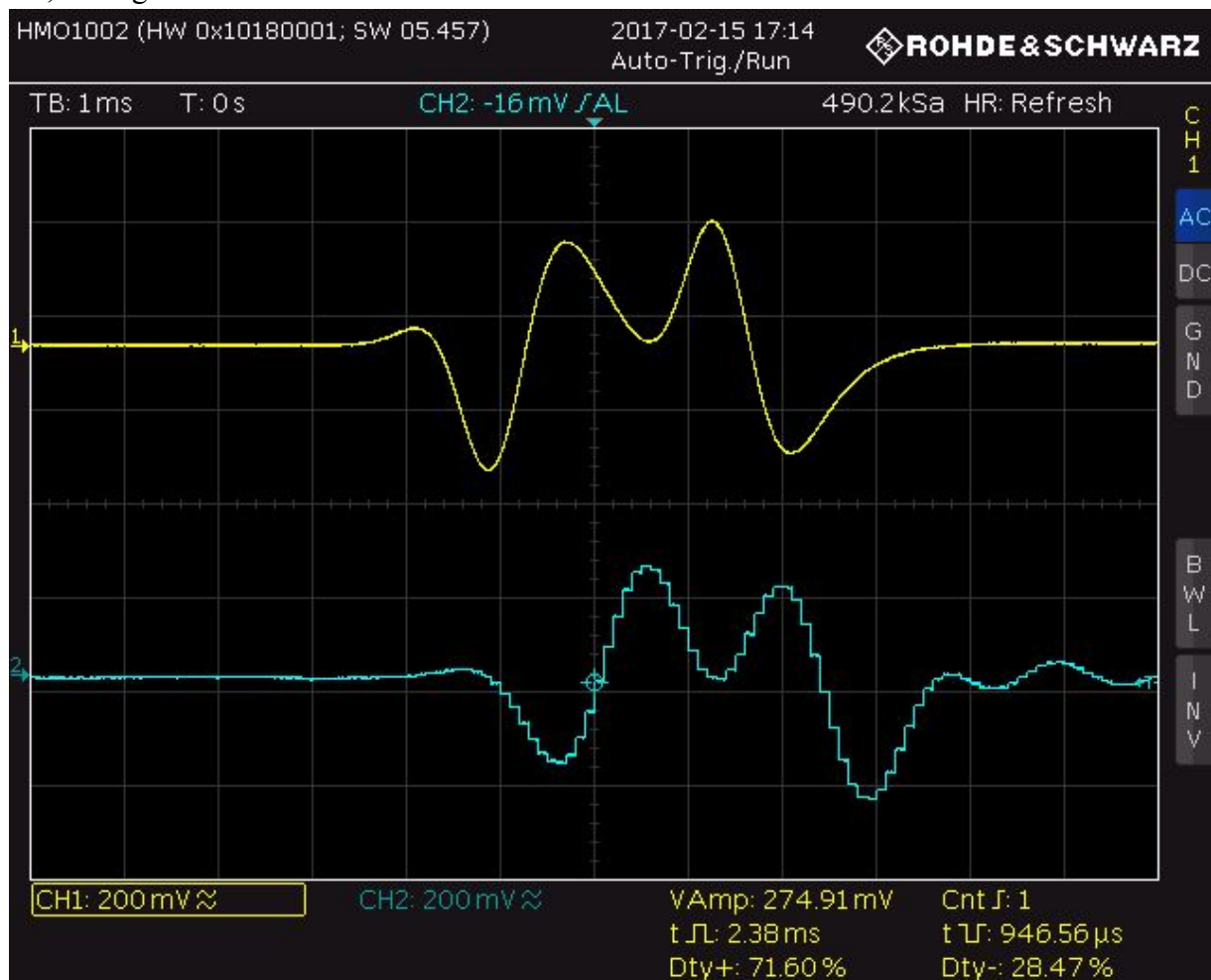
FIR, ren insignal



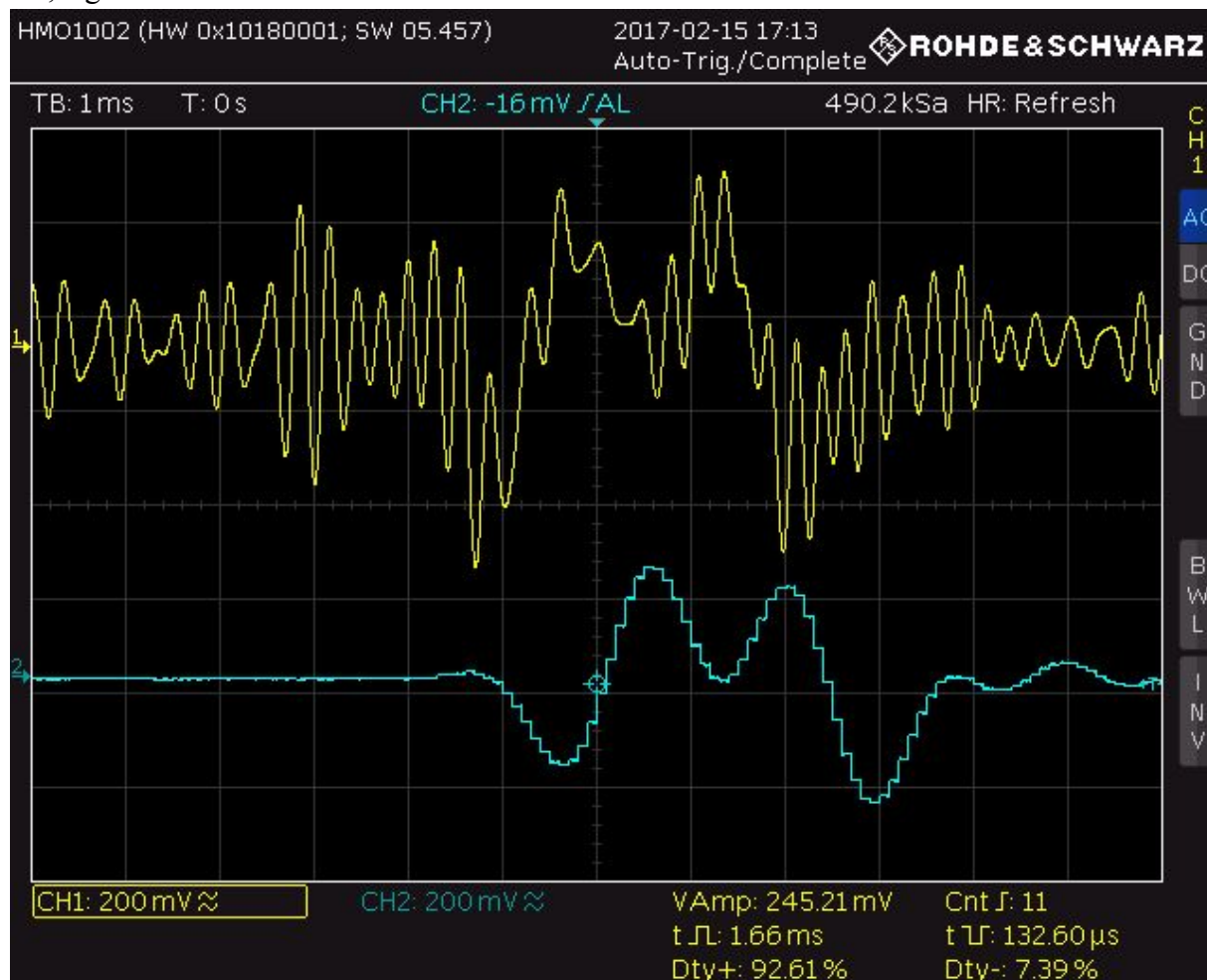
FIR, signal med brus



IIR, ren signal



IIR, signal med brus



Kommentar:

Slutsatser

Jämför FIR och IIR-filter när det gäller beräkningstider och påverkan på kurvformen:

FIR-filter har en kortare beräkningstid jämfört med IIR-filter. Fördelen med FIR är att man kan uppnå en linjär fas vilket filtrerar bort störning och ger den likadana kurvformen, då måste de ingående deltonerna fördröjas lika lång tid i filtret och därför tar kortare tid. För IIR filtret så kan den inte uppnå linjär fas och därför långsammare samt därför får sämre kurvform än FIR filtret.

Avslutande kommentar

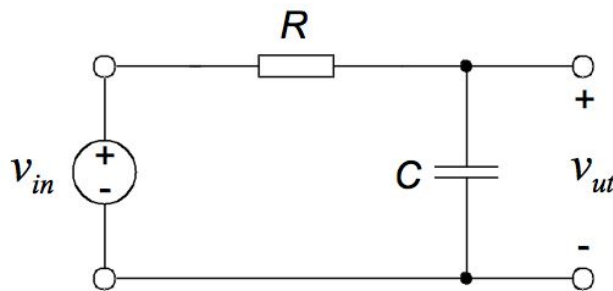
Antivikningsfilter (Antialiasing filter)

När man samplar en signal skall man se till att frekvenser över halva sampelfrekvensen är försumbara för att undvika aliasing (vikning). Detta gör man med ett analogt lågpasfilter. Eftersom fyrkantssignalen är rik på höga frekvenser (dvs övertonernas amplitud avtar långsamt med frekvensen) borde detta vara extra viktigt i vårt fall. Problemet är dock att analog filter inte har linjär fas. Det betyder att om vi ansluter ett sådant filter blir inte faslinjäriteten hos FIR-filtret så tydlig. I de fall faslinjäriteten är viktig kan man använda analog filter av sk Besseltyp. De är relativt faslinjära men behöver vara av rätt högt ordningstal.

Det finns tyvärr ingen "inbyggd" toolbox i Matlab för att designa analog filter. Den som är intresserad och har tid kan dock ladda ner en sådan freeware toolbox här:

<http://www.mathworks.es/matlabcentral/fileexchange/9458-analog-filter-design-toolbox>.

I enklare tillämpningar kan det ofta duga med ett enkelt RC-filter av första ordningen.



Det finns också i stort sett kompletta analog filterkretsar att köpa.

Utfilter

Signalen från DA-omvandlaren är ju trappformad och innehåller rester av aliasfrekvenserna som vi såg i andra labben.

Om man vill "snygga" till utsignalen kan man koppla in ett analogt lågpasfilter efter DA-omvandlaren, t.ex. ett likadant filter som man använder som antivikningsfilter. Filtret "jämnar" ut utsignalen. Sett till frekvensinnehållet så dämpar filtret resterna av aliasfrekvenserna. Problemet med faslinjäritet är detsamma som för antivikningsfiltret.

Lämna som vanligt in rapporten i pdf-format på Its learning. Var beredd på att kunna beskriva mätningarna och demonstrera programmet och förklara hur det fungerar.