Diskret-tids systemer

Miniprojekt

G4-101a
Trine Nyholm Jensen
Christian Hilligsøe-Toft
Frederik Appel Vardinghus-Nielsen
Martin Kamp Dalgaard

Aalborg Universitet
Institut for Matematiske Fag

Indhold

1 Indledning

Denne rapport dokumenterer gruppens arbejde med miniprojektet i kurset Diskret-tids systemer, som omhandler design og simulering af et diskret-tids signalbehandlingssystem. Mere specifikt arbejdes der med

- **Signal-generering**
  Simulering af sampling af et kontinuert signal $s(n)$.
- **Signal-analyse**
  Frekvensanalyse af det samplede signal.
- **Signal-modifikation**
  Filtrering af det samplede signal.

Hvert punkt har et dedikeret kapitel med redegørelse for overvejelser og beslutninger samt dokumentation af implementeringen i Python. Det vedlagte Python-script udfører databehandlingen og printer figurerne anvendt i denne rapport. Derudover gemmes figurerne i en separat mappe, så læseren kan zoome ind og kigge nærmere på dem.

2 Generering af signal

I dette kapitel genereres et signal $s(n)$, som skal samples til $s[n]$. I projektbeskrivelsen er følgende krav til signalet opgivet:

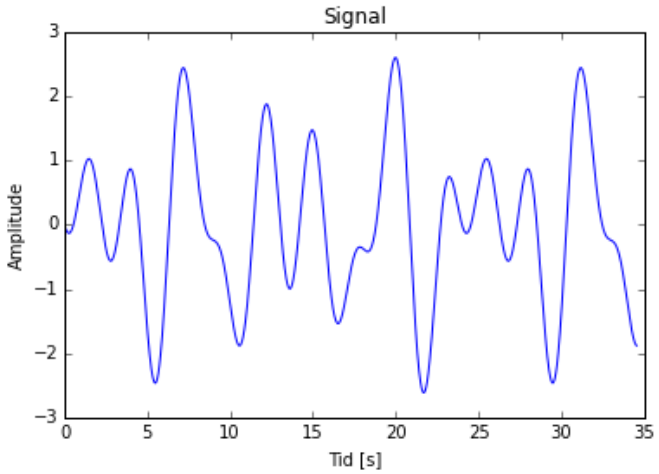
- Signalet skal bestå af en sum af tre individuelle sinus-komponenter.
- De tre frekvenser af komponenterne skal findes ved $\omega_1 = \pi/3$, $\omega_2 = \pi/2$ og $\omega_3 = 3\pi/4$, alle i rad/s.
- Komponenterne skal have indbyrdes fasedrej på $\varphi = 2\pi/3$ rad.
- Komponenterne skal have ens amplitude.

På baggrund af ovenstående krav er følgende signal opstillet:

$$s(n) = \sin(\omega_1 n) + \sin(\varphi + \omega_2 n) + \sin(2\varphi + \omega_3 n) \tag{2.1}$$

$$= \sin\left(\frac{\pi}{3}n\right) + \sin\left(\frac{2\pi}{3} + \frac{\pi}{2}n\right) + \sin\left(\frac{4\pi}{3} + \frac{3\pi}{4}n\right) \tag{2.2}$$

I figur 2.1 ses signalet.



Figur 2.1: Plot af signalet i lidt over en periode.

2.1 Sampling af signal

For at kunne simulere signalet (2.1) i Python og efterfølgende lave frekvensanalyse og filtrering af det, kræves det, at signalet samples. For at undgå spejling som følge af for lav samplingsfrekvens f_s bruges Nyquist-Shannons samplingsteorem, som siger,

at et signal kan fuldstændigt bestemmes ud fra dets samples, hvis det samples med en samplingsfrekvens f_s således, at

$$f_s > 2B, \tag{2.3}$$

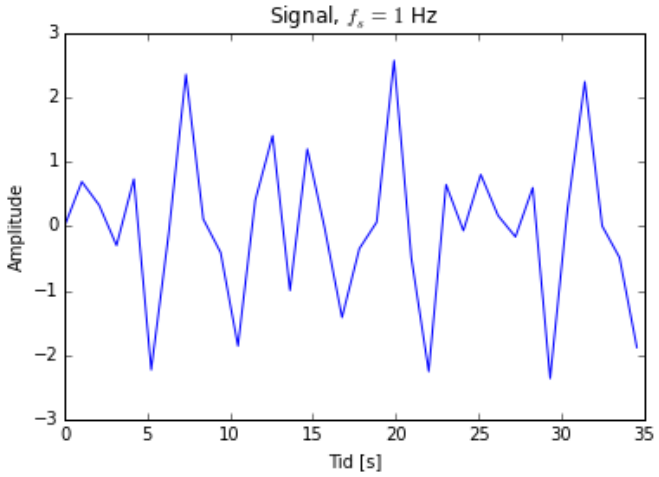
hvor B er den højeste frekvens optrædende i signalet. Derfor bestemmes B nu ved at bestemme den højeste frekvens optrædende i signalet ved at bestemme frekvenserne i de enkelte komponenter[1]:

$$\begin{aligned} f_1 &= \frac{\omega_1}{2\pi} = \frac{1}{6} \text{ Hz}, \\ f_2 &= \frac{\omega_2}{2\pi} = \frac{1}{4} \text{ Hz}, \\ f_3 &= \frac{\omega_3}{2\pi} = \frac{3}{8} \text{ Hz}. \end{aligned}$$

Det ses, at $f_3 = \frac{3}{8}$ Hz er den højeste frekvens i signalet. Altså haves $B = \frac{3}{8}$ Hz og Nyquistraten $2B = \frac{3}{4}$ Hz. Fra dette konkluderes, at en samplingsfrekvens, der opfylder

$$f_s > \frac{3}{4} \text{ Hz} \tag{2.4}$$

vil medføre, at der ikke sker spejling, når $s(n)$ samples. I figur 2.2 ses $s(n)$ samplet ved $f_s = 1$ Hz.



Figur 2.2: Signal samplet ved 1 Hz.

Med $s[n]$, som fuldstændigt bestemmer $s(n)$, fortsættes nu til frekvensanalysen.

[1] Det antages, at $s(n)$ er båndbegrænset.

3 Frekvensanalyse

I dette kapitel foretages der frekvensanalyse af det samplede signal $s[n]$. Dette gøres for at undersøge signalets frekvensmæssige indhold og fortsætte det videre arbejde med modifikation af signalets indhold.

3.1 Dokumentation af algoritmer

Frekvensanalysen gøres mulig med den diskrete Fouriertransformation (DFT), som beregnes ved hjælp af en *fast Fourier transform*-algoritme (FFT) baseret på Cooley-Tukey algoritmen. Algoritmen er implementeret i Python og udnytter DFT-summens symmetri til at lave en rekursiv opdeling af summen, således den beregningsmæssige kompleksitet nedbringes fra $O(N^2)$ (naiv implementering af DFT) til $O(N \log N)$ for store N, hvor N er datalængden. Herunder ses implementeringen af DFT og FFT i et Pythonscript.

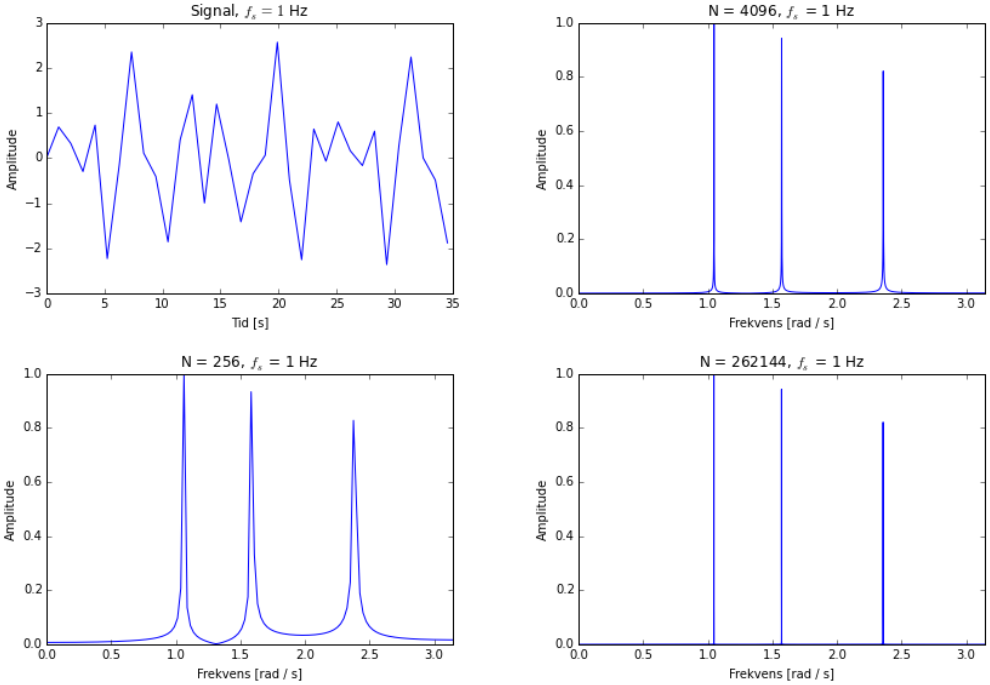
```
def DFT(x,c):
    X = np.zeros(c,dtype=complex)
    for k in range(len(x)):
        a = 0+0*1j
        for n in range(c):
            a += x[n]*np.exp(-2*np.pi*1j*k*n/float(c))
            X[k] = a
    return X

def is_power2(num): # Checks if a number is a power of 2
   return num != 0 and ((num & (num - 1)) == 0)

def FFT(x):
    N_new = len(x)
    if is_power2(N_new) == False:
        raise ValueError("N skal vaere en potens af 2.")
    elif N_new == 2:
        return DFT(x,N_new) # Returnerer DFT naar data ikke kan deles mere op
    else:
        X_even = FFT(x[::2]) # Deler rekursivt input op - lige dele
        X_odd = FFT(x[1::2]) # Deler rekursivt input op - ulige dele
        factor = np.exp(-2j * np.pi * np.arange(N_new) / N_new) #
            Twiddlefaktor
        return np.concatenate([X_even + factor[:int(N_new / 2)] * X_odd,
                               X_even + factor[int(N_new / 2):] * X_odd])
```

3.2 Tids- og frekvensplot

I dette afsnit undersøges signalets frekvensspektrum og hvordan dette afhænger af antallet af samples N og samplingsfrekvensen f_s. Det vides, at opløsningen i tid afhænger af f_s og at opløsningen i frekvens afhænger af f_s og N, således

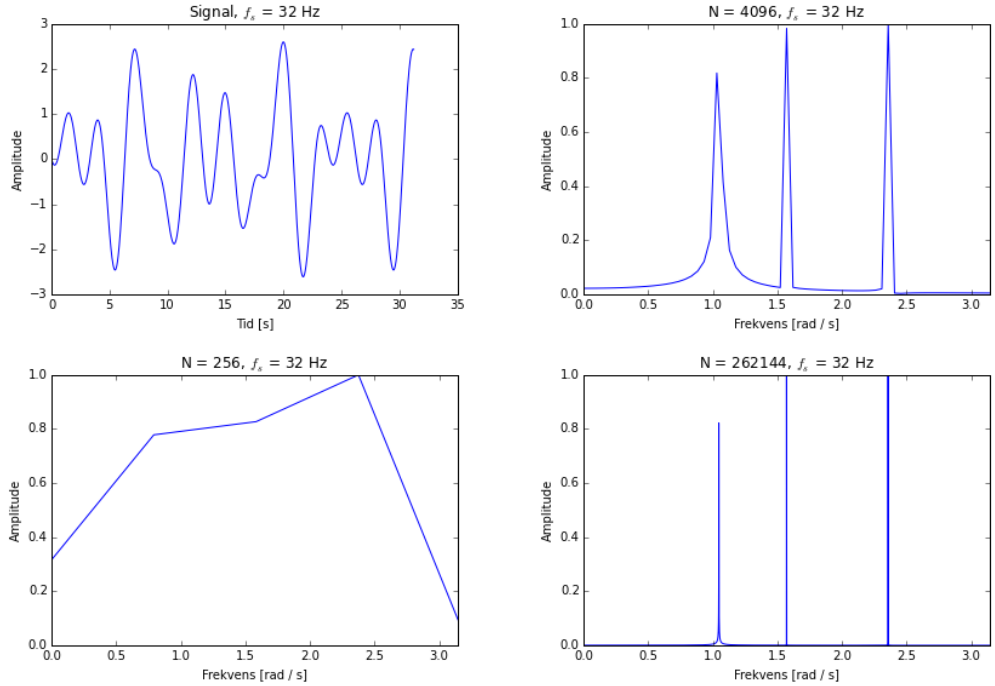
$$T = \frac{1}{f_s} \quad \text{og} \quad bin = \frac{f_s}{N}, \tag{3.1}$$

hvor T er samplingsperioden og bin er intervallet mellem to diskrete frekvenser i frekvensdomænet. Dette afsnit undersøger disse relationer.
På figur 3.1 ses signalet i tids- og frekvensdomænet samplet med $f_s = 1$ Hz og med $N = 2^8, 2^{12}, 2^{18}$ samples.



Figur 3.1: Plot af signal samplet ved $f_s = 1$ Hz i tidsdomænet og plot af de tilhørende frekvensdomæner for antal af samples $N = 2^8, 2^{12}, 2^{18}$.

På figur 3.2 ses signalet ligeledes i tids- og frekvensdomænet samplet med $f_s = 32$ Hz og med $N = 2^8, 2^{12}, 2^{18}$ samples.



Figur 3.2: Plot af signal samplet ved $f_s = 32$ Hz i tidsdomænet og plot af udsnit af de tilhørende frekvensdomæner for antal af samples $N = 2^8, 2^{12}, 2^{18}$.

3.3 Diskussion

Det ses tydeligt på tidsdomænerne i figur 3.1 og 3.2, at opløsningen af signalet i tid afhænger af f_s og bliver bedre som f_s stiger. Dette stemmer overens med (3.1).

Det ses ligeledes tydeligt, at opløsningen i frekvensdomænet afhænger af N og bliver bedre som N stiger. Det er ydermere bemærkelsesværdigt, at øgning af f_s giver ringere opløsning i frekvensdomænet som set i andet plot i figur 3.2. Disse observationer stemmer ligeledes overens med (3.1).

Det ses i figur 3.1 og 3.2, at amplituderne for de tre frekvenser, som signalet indeholder, ikke er lige store, selvom signalet tydeligvis er konstrueret således. Det ses også på figur 3.1 og 3.2, at amplituderne varierer som f_s varierer, men ikke som N varierer. Dette skyldes spektral lækage, og er ikke et problem for rekonstruktion af signalet, da energien blot er distribueret rundt om den frekvens, som den bør forefindes ved, således amplituden ved denne frekvens bliver 1.

4 Filtrering af signal

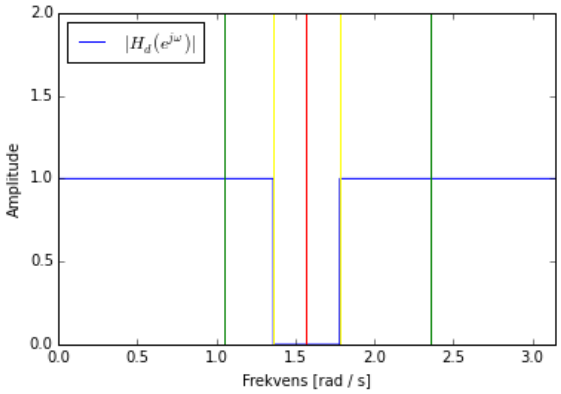
I dette kapitel designes et båndstopfilter med endelig impulsrespons (altså et FIR-filter), der har til formål at eliminere den midterste frekvens ved $\omega_2 = \frac{\pi}{2}$ mest muligt og samtidig reducere de øvrige frekvenser ved $\omega_1 = \frac{\pi}{3}$ og $\omega_3 = \frac{3\pi}{4}$ mindst muligt. I kapitlet beskrives først specifikationerne og design af filteret, hvorefter filterets effekt på signalet vurderes. Afslutningsvist forsøges filteret optimeret.

4.1 Specifikationer

FIR-filteret vælges til at være af type 1, altså er filterordenen M lige og impulsresponsen $h[n]$ er symmetrisk, så $h[n] = h[M - n]$. Filteret har 2 knækfrekvenser ω_{c_1} og ω_{c_2} i rad/s, som vælges til at ligge symmetrisk omkring ω_2, der skal elimineres ved hjælp af filteret:

$$\omega_{c_1} = \omega_2 - \delta,$$
$$\omega_{c_2} = \omega_2 + \delta,$$

hvor δ vælges til at være $\frac{\pi}{15}$. Filteret designes dermed som et båndstopfilter med ovenstående knækfrekvenser. Den ideelle amplituderespons $|H_d(\mathrm{e}^{j\omega})|$ er skitseret på figur 4.1 ud fra de opstillede specifikationer.



Figur 4.1: Den ideelle amplituderespons for filteret. De to grønne streger markerer ω_1 og ω_3, som skal beholdes, og de to gule streger markerer ω_{c_1} og ω_{c_2}, som ligger symmetrisk omkring den røde streg, der markerer ω_2, som skal elimineres.

4.2 Design af filter

I dette afsnit dokumenteres designprocessen samt væsentlige dele af Python-scriptet, der anvendes til at filtrere signalet.

Først defineres henholdsvis filterordenen M, som på grund af implementeringen af FFT-algoritmen skal være en potens af 2, længden af filteret $l = M - 1$, en vektor n med $l + 1$ heltal mellem 0 og l samt en vektor x med $l + 1$ værdier mellem $-\pi$ og π.

Den ideelle impulsrespons h_d defineres nu som en funktion på baggrund af den ideelle amplituderespons $|H_d(e^{j\omega})|$, jf. figur 4.1. Udledning af $h_d[n]$ findes i bilag A.1. Funktionen i Python bruger vektoren n, ordenen M samt frekvenserne ω_{c_1} og ω_{c_2} defineret ovenfor. Den ideelle impulsrespons er derfor:

$$h_d[n] = \begin{cases} \dfrac{\sin\left(\omega_{c_1}\left(n - \frac{M}{2}\right)\right)}{\pi\left(n - \frac{M}{2}\right)} - \dfrac{\sin\left(\omega_{c_2}\left(n - \frac{M}{2}\right)\right)}{\pi\left(n - \frac{M}{2}\right)} & \text{for} \quad n \neq \frac{M}{2} \\ 1 - \frac{\omega_{c_2} - \omega_{c_1}}{\pi} & \text{for} \quad n = \frac{M}{2} \end{cases}$$

Dette er implementeret i Python-scriptet således:

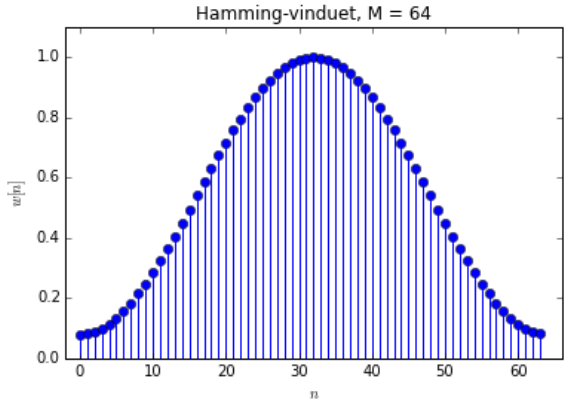
```
def hd(n,M,f1,f2):
    hd = np.zeros(len(n))
    for i in range(len(n)):
        if n[i] == M/2:
            hd[i] = 1 - (f2 - f1)/np.pi
        else:
            hd[i] = (np.sin(f1*(n[i] - M/2.)) / (np.pi*(n[i] - M/2.))) \
            - (np.sin(f2*(n[i] - M/2.)) / (np.pi*(n[i] - M/2.)))
    return hd
```

Fordi impulsresponsen er endelig skal den ideelle impulsrespons ganges med en vinduesfunktion, og dette Fourier-transformeres herefter for at opnå en realiserbar frekvens- og amplituderespons. Der er forskellige muligheder for denne vinduesfunktion såsom det rektangulære vindue, Hamming-vinduet, Hann-vinduet og Blackman-vinduet.[1] Implementeringen af vinduerne i scriptet er ganske simpel, og som eksempel ses implementeringen af Hann- og Hamming-vinduet herunder:

```
def ha(n,M,a): # Hann-vindue, hvis a = 0.5. Hamming-vindue, hvis a = 0.54.
    w = np.zeros(len(n))
    for i in range(len(n)):
        if n[i] >= 0 and n[i] <= M:
            w[i] = a - (1 - a)*np.cos((2*np.pi*n[i])/M)
        else:
            w[i] = 0
    return w
```

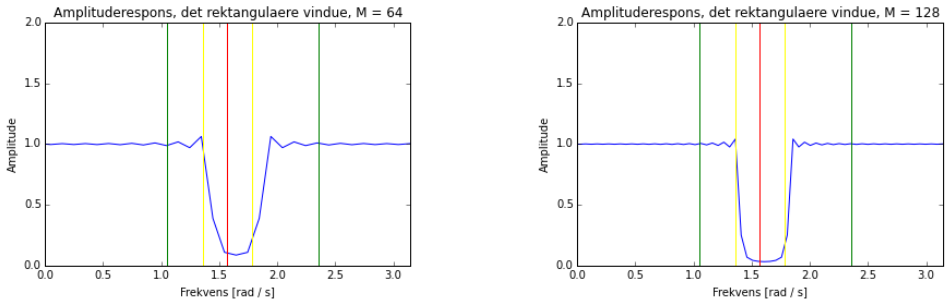
Grafen for et Hamming-vindue af orden 64 er vist på figur 4.2.

[1] Disse vinduer er beskrevet på side 559-560 i *Discrete-Time Signal Processing*, Oppenheim & Schafer (2014).



Figur 4.2: Eksempel på Hamming-vinduet med $M = 64$.

Efter implementeringen af vinduerne kaldes funktionerne for den ideelle impulsrespons og det ønskede vindue, som herefter ganges sammen. Dette resultat Fouriertransformeres ved hjælp af den implementerede FFT beskrevet under frekvensanalysen. Hermed opnås frekvensresponsen $H(\mathrm{e}^{j\omega})$ og amplituderesponsen $|H(\mathrm{e}^{j\omega})|$, hvoraf sidstnævnte (ideelt set) er 0 mellem ω_{c_1} og ω_{c_2} og 1 ellers. Dette afhænger dog især af valget af vinduet og filterordenen M. Figur 4.3 viser eksempler på anvendelsen af det rektangulære vindue for $M = 64$ og $M = 128$.

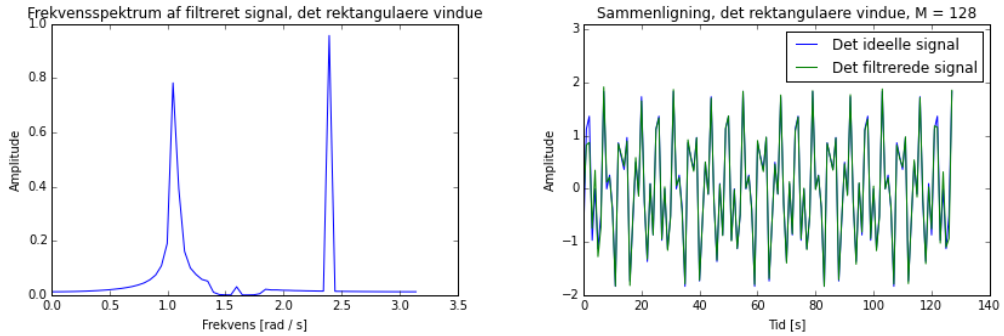


Figur 4.3: Eksempler på båndstopfilteret under anvendelse af det rektangulære vindue og med forskellige filterordener M.

Ud fra figur 4.3 ses det, at filteret til en vis grad overholder specifikationerne for begge værdier af M, at transitionsbåndet bliver smallere for højere M, at stopbåndet kommer tættere mod 0 for højere M samt at der dog er såkaldte "ripples" i pasbåndet. Disse "ripples" skyldes diskontinuiteterne ved det rektangulære vindue, der pludselig skifter fra 0 til 1 og tilbage igen. For at undgå dette kan man bruge et glattere vindue som for eksempel Hamming-vinduet illustreret på figur 4.2.

4.3 Filtrering

Filtreringen af signalet foregår ved at folde signalet med impulsresponsen. Ved Fourier-transformationen svarer foldning i tidsdomænet til multiplikation i frekvensdomænet, og frekvensspektret for det filtrerede signal kan derfor findes ved at Fourier-tranformere signalet og filteret hver for sig og herefter gange disse to sammen. Figur 4.4 viser henholdsvis frekvensspektret for det filtrerede signal og en sammenligning mellem det filtrerede signal og det ideelle signal ved anvendelse af det rektangulære vindue.



Figur 4.4: Frekvensspektrum for det filtrerede signal under anvendelse af filter med det rektangulære vindue af orden $M = 128$ (venstre) og sammenligning af det filtrerede signal og det ønskede signal (højre).

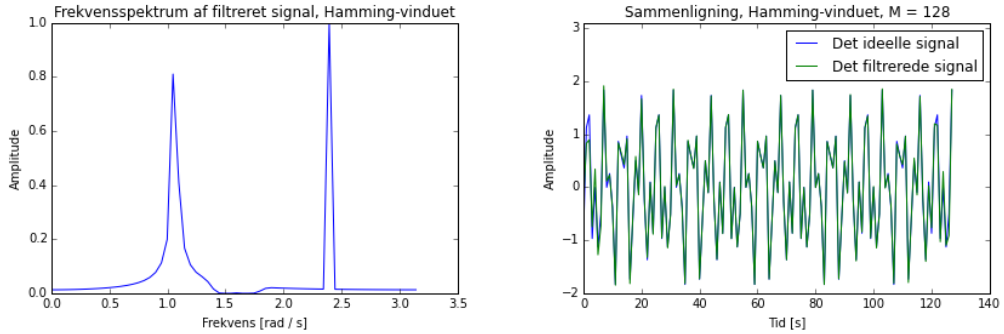
Af frekvensspektret på figur 4.4 ses det, at ω_2 fremgår, men dog med væsentlig minimeret amplitude. Dette skyldes, at stopbåndet i det anvendte filter ikke er identisk 0.

4.4 Vurdering

På baggrund af graferne i sektion 4.3 kan det vurderes, at det designede FIR-filter kun til en hvis grad formår at filtrere frekvensen $\omega_2 = \pi/2$ fra det samplede signal $s[n]$ ved anvendelse af det rektangulære vindue og en filterorden på 128.
Figur 4.3 illustrerer hvorledes, at de opstillede specifikationer kun til en vis grad overholdes ved en filterorden på henholdsvis 64 og 128 og anvendelse af det rektangulære vindue. Det ses tydeligt, at transitionsbåndet bliver smallere og at stopbåndet flader ud jo højere orden der anvendes. De ripples, der forekommer i pasbåndet formindskes også, når ordenen øges. Dog forsvinder de ikke, da det er det rektangulære vindue, der anvendes, og amplituderesponsen er ikke 0 i stopbåndet, hvormed signalet muligvis ikke filteres optimalt. Herunder optimeres filteret ved at anvende Hamming-vinduet.

4.5 Optimering

Forsøg har vist, at der opnås en god filtrering ved hjælp af Hamming-vinduet med $M = 128$, hvormed frekvensen $\omega_2 = \pi/2$ i udstrakt grad filtreres fra uden at påvirke de øvrige frekvenser. Dette resultat ses på figur 4.5, der henholdsvis viser frekvensspektret, hvor frekvensen $\omega_2 \approx 1.57$ ikke fremgår, og en sammenligning mellem det ideelle filtrerede signal og det filtrerede signal. Af sidstnævnte fremgår det, at de to signaler ligger oveni hinanden med enkelte afvigelser. Sammenlignes dette med resultaterne i figur 4.4 bemærkes det dog, at ved repræsentation af signalet i tid ses ingen visuel forbedring af det filtrerede signal til trods for en synlig optimering i frekvensdomænet.



Figur 4.5: Frekvensspektrum for det filtrerede signal under anvendelse af filter af orden $M = 128$ (venstre) og sammenligning af det filtrerede signal og det ønskede signal (højre).

Amplituderesponser for Hamming-vinduet med $M = 64$ og $M = 128$ ses desuden på figur A.1 i bilag A.2. Af disse figurer fremgår det, at et filter med Hamming-vinduet med $M = 128$ er tæt på den ideelle amplituderespons.

4.6 Samlet konklusion

I dette miniprojekt er et signal indeholdende 3 sinus-komponenter blevet genereret og samplet, hvorefter det er blevet analyseret for dets frekvensmæssige indhold ved hjælp af en implementering af DFT'en i form af en FFT-algoritme. Afslutningsvis er signalet blevet filtreret ved hjælp af et båndstopfilter med endelig impulsrespons og et Hamming-vindue af orden 128, som i vid udstrækning har formået at eliminere frekvensen på $\pi/2$ uden at påvirke de øvrige frekvenser. Resultatet af denne filtrering ses på figur 4.5.

A Bilag

A.1 Udledning af den ideelle impulsrespons

For at udlede den ideelle impulsrespons defineres først den ideelle frekvensrespons som følger på baggrund af filterets specifikationer defineret i sektion 4.1.

$$H_d(\mathrm{e}^{j\omega}) = \begin{cases} \mathrm{e}^{-j\omega\frac{M}{2}}, & |\omega| \leq \omega_{c_1} \\ 0, & \omega_{c_1} \leq |\omega| \leq \omega_{c_2} \\ \mathrm{e}^{-j\omega\frac{M}{2}}, & \omega_{c_2} \leq |\omega| \end{cases}$$

Herudfra bestemmes den ideelle impulsrespons ved hjælp af den inverse Fourier-transformation:

$$\begin{aligned} h_d[n] &= \frac{1}{2\pi}\left(\int_{-\pi}^{-\omega_{c_2}} \mathrm{e}^{-j\omega\frac{M}{2}} \cdot \mathrm{e}^{j\omega n} d\omega + \int_{-\omega_{c_1}}^{\omega_{c_1}} \mathrm{e}^{-j\omega\frac{M}{2}} \cdot \mathrm{e}^{j\omega n} d\omega + \int_{\omega_{c_2}}^{\pi} \mathrm{e}^{-j\omega\frac{M}{2}} \cdot \mathrm{e}^{j\omega n} d\omega\right) \\ &= \frac{1}{2\pi}\left(\int_{-\pi}^{-\omega_{c_2}} \mathrm{e}^{j\omega\left(n-\frac{M}{2}\right)} d\omega + \int_{-\omega_{c_1}}^{\omega_{c_1}} \mathrm{e}^{j\omega\left(n-\frac{M}{2}\right)} d\omega + \int_{\omega_{c_2}}^{\pi} \mathrm{e}^{j\omega\left(n-\frac{M}{2}\right)} d\omega\right). \end{aligned}$$

Her udnyttes det nu, at der gælder:

$$\begin{aligned} \int_{-\pi}^{\pi} \mathrm{e}^{j\omega\left(n-\frac{M}{2}\right)} d\omega &= \int_{-\pi}^{\pi} \cos\left(\omega\left(n-\frac{M}{2}\right)\right) + j\sin\left(\omega\left(n-\frac{M}{2}\right)\right) d\omega \\ &= \left[\sin\left(\omega\left(n-\frac{M}{2}\right)\right)\right]_{-\pi}^{\pi} \cdot \frac{1}{n-\frac{M}{2}}. \end{aligned}$$

Således fås:

$$h_d[n] = \frac{1}{\pi\left(n-\frac{M}{2}\right)}\left(\sin\left(\omega_{c_2}\left(n-\frac{M}{2}\right)\right) - \sin\left(\omega_{c_1}\left(n-\frac{M}{2}\right)\right) - \sin\left(\pi\left(n+\frac{M}{2}\right)\right)\right).$$

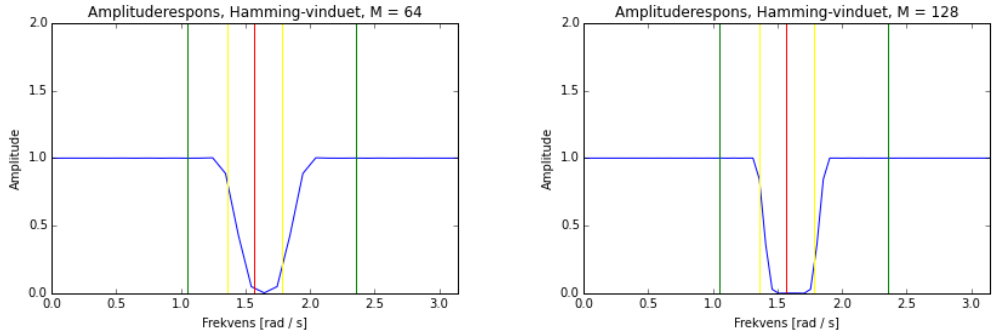
Da sidste led må være 0 defineres den ideelle impulsrespons som følger:

$$h_d[n] = \frac{1}{\pi\left(n-\frac{M}{2}\right)}\left(\sin\left(\omega_{c_2}\left(n-\frac{M}{2}\right)\right) - \sin\left(\omega_{c_1}\left(n-\frac{M}{2}\right)\right)\right)$$

Dette gælder dog kun for $n \neq 0$, og dermed defineres $h_d[0]$ seperat. Hertil anvendes l'Hôspitals regel, som siger, at $\lim \frac{f(x)}{g(x)} = \lim \frac{f'(x)}{g'(x)}$.

$$\begin{aligned} h_d[0] =& \frac{1}{\pi}\left(\cos\left(\omega_{c_1}\left(n-\frac{M}{2}\right)\right)\omega_{c_1} - \cos\left(\omega_{c_2}\left(n-\frac{M}{2}\right)\right)\omega_{c_2} + \cos\left(\pi\left(n-\frac{M}{2}\right)\right)\pi\right) \\ =& \frac{1}{\pi}\left(\omega_{c_1} - \omega_{c_2} + \pi\right) \\ =& 1 - \frac{\omega_{c_2} - \omega_{c_1}}{\pi} \end{aligned}$$

A.2 Amplituderespons for filteret



Figur A.1: Amplituderesponser for filteret af orden $M = 64$ og $M = 128$ ved anvendelse af Hamming-vinduet.