TSKS24 Signaler och bilder

Lab 2: Digitalisering

Mikael Olofsson

12 februari 2024

Fyll i detta med bläckpenna (icke suddbart)

Laborant 1
Person-nummer
Laborant 2
Person-nummer
Datum
Godkänd

1 Allmänt

Denna laboration syftar till att ge dig vissa insikter i problem och begränsningar i samband med digitalisering av signaler. För den som inte har använt Matlab tidigare, så avser laborationen också att ge en viss vana vid Matlab.

2 Teori

Läs igenom kursbokens avsnitt om DFT, fönstring, sampling och kvantisering. Där beskrivs den teori som berör det du ska göra i denna laboration. Läs också igenom "Short Matlab Manual" av Mikael Olofsson, som du laddar ner från kursrummet, speciellt om du inte har använt Matlab tidigare. Det dokumentet utgör en kortfattad manual för Matlab, och innehåller dessutom lite instruktioner om hur man skriver skript och funktioner i Matlab.

Det är en god idé att skriva skript för alla deluppgifter i labben. Då kan du med ett kommando köra all programkod för uppgiften. Det är då också en god idé att börja alla skript med följande:

```
clear; close all
```

Det första kommandot rensar arbetsminnet och det andra stänger alla figurfönster. På det viset riskerar du inte att använda gamla siffror via gamla variabler och du blir av med gamla figurfönster. Då vet du att allt du har genererat kommer från just det skriptet.

3 Förberedelseuppgifter

Läs igenom lab-PM. Det ska man alltid göra inför en laboration, oavsett om någon sagt att man ska göra det eller inte.

En stationär sinussignal med frekvens 4 kHz samplas och rekonstrueras sedan idealt. Vilken frekvens får resultatet om samplingsfrekvensen är **a**. 10 kHz, **b**. 5 kHz och **c**. 3 kHz?

4 Labuppgifter

4.1 Inledande övning - skapa och presentera signaler

Denna första övning avser dels att ge lite Matlab-vana, dels att introducera några kommandon som är användbara i denna lab, och slutligen att definiera några signaler som du ska använda dig av. För att starta Matlab, skriv följande i ett terminalfönster:

```
module add prog/matlab/2023b
matlab
```

Denna övning resulterar i ett flertal grafer. Se till att du kan redovisa dem alla för assistenten då detta avsnitt är avklarat.

Följande Matlab-kommandon används i detta avsnitt:

```
sin, plot, stem, axis, figure, title, xlabel, ylabel, hold on/off,
histogram, subplot
```

Det kan vara en bra ide att läsa på om dem i Matlabs hjälp allt eftersom du använder dem. Du når Matlabs hjälp med frågetecken-knappen upptill nära mitten i fönstret.

Matlab har en interaktiv pront som ser ut så här:

```
>>
```

Den används här i detta lab-PM för att indikera något som kan eller bör skrivas in i Matlab.

Kommentarer i Matlab inleds med ett `%`, och det använder vi även här för att kommentera angiven Matlab-kod. Exempelvis betydelsen av ett semikolon (`;`):

```
>> b=3+4   % skapar variabeln b med värdet 7 och skriver ut det.
>> a=1+2;  % skapar variabeln a med värdet 3, men skriver inte ut det.
```

Vi behöver några signaler att använda som utgångspunkt i denna laboration. Alla dessa signaler ska ha följande parametrar:

- Amplitud 1
- tidsutbredning 2.5 sekunder
- Samplingsfrekvens 40 kHz

Skapa nu två sådana signaler enligt följande:

- Signalen `x1` ska ha frekvensen 8000 Hz.
- Signalen `x2` ska ha frekvensen 8017 Hz.

Den första av dessa kan skapas så här:

```
>> T=2.5;              % Tidsutbredning
>> fs=4e4;             % Samplingsfrekvens
>> N=T*fs;             % Antal sampel
>> n=0:N-1;            % Vektor med sampelindex
>> t=1/fs*n;           % Vektor med sampeltidpunkter
>> f1=8000;            % signalens frekvens
>> x1=sin(2*pi*f1*t);  % Vektor med alla sampel
```

Den koden finns i kursrummet under `Kursdokument/laborationer`. Ladda ner skriptet `lab2part41params.m`. Låt gärna variabelnamnen ovan fortsatt ha de värden som de ges här. Det förutsätts här och där i lab-PM. Editera skriptet, så att det också skapar signalen `x2`. Lämpligen startar du sedan varje skript med att du anropar det skriptet. Det skriptet inkluderar även de rekommenderade kommandona `clear` och `close all`. Genom att anropa skriptet, så uppnår du alltså även det.

Plotta dessa signaler och zooma in på *fem perioder* av motsvarande analoga signaler, dels med kommandot `plot` och dels med kommandot `stem`. Notera skillnaden mellan dessa två sätt att använda `plot`:

```
>> plot(x1)
>> plot(t,x1)
```

Du kan använda kommandot `axis` efter att du har skapat en graf, för att visa en önskad del av den. I detta sammanhang kan du ha nytta av kommandot `figure` som gör en existerande graf aktiv, vilket betyder att nästa kommando som manipulerar en graf opererar på den grafen. Man kan också göra en existerande graf aktiv genom att klicka på den. Om en

graf inte existerar med angivet nummer, så skapar `figure` ett fönster för det. Fönstren med grafer har menyer. Titta specifikt på menyn Tools, och prova där menyvalen Zoom In, Zoom Out, Data Tips, Pan och Restore View.

Efter att ha valt någon av Zoom-funktionerna, så kan du högerklicka i grafen, så ser du några olika varianter att zooma. Med Data Tips kan du göra sifferavläsningar i grafen genom att klicka på en punkt. Pan låter dig flytta grafen, och även den har alternativ som du når genom att högerklicka på grafen. Slutligen, om du vill tillbaka till hur grafen såg ut då du skapade den, använd Restore View.

Horisontella axeln ska vara graderad med naturlig tid i sekunder. Använd gärna kommandot `title` för att ge en rubrik till varje graf. Kommandona `xlabel` och `ylabel` kan användas för att ange vad som finns på de två axlarna i en graf.

Kombinera kommandona `plot`, `hold on`, `stem` och `hold off` för att skapa en graf av *fem perioder* av signalerna både som en kontinuerlig signal och som en samplad signal. Det går också att skapa en kontinuerlig graf med markeringar för själva samplen på följande sätt:

```
>> plot(t,x2,'b-',t,x2,'rx')
```

Försäkra dig om att du har klart för dig vad varje detalj i uttrycket ovan gör. Matlabs hjälptext för kommandot `plot` kan vara användbar här.

Använd `histogram` för att skapa ett histogram av de två signalerna med 100 bingar.

☞ Besvara följande:

- Hur många sampel består signalerna av? .....
- Hur många perioder (av de tänkta analoga signalerna) består signalerna av?

  `x1`: ..... `x2`: .....
- Hur många sampel består en period (av de tänkta analoga signalerna) av?

  `x1`: ..... `x2`: .....
- Går perioden för vardera signal ens jämnt ut på ett antal sampel?

  `x1`: ..... `x2`: .....
- Vad visar histogrammen?

  .....
- Varför är de så olika?

  .....

Signatur:

Tips: Kommandot `subplot` kan användas för att placera flera grafer bredvid varandra i samma fönster. Det är en god vana att skriva `figure` innan varje plot. Det ger dig ett nytt figurfönster till den plotten, och då försvinner inte den du gjorde innan.

4.2 Spektrum - linjär skala och dB

Följande nya Matlab-kommandon används i detta avsnitt:

```
fft, abs, db
```

Använd kommandot `fft` för att fouriertransformera de två signalerna. Detta kommando beräknar DFTn av sitt argument med en snabb implementering som brukar kallas FFT (Fast Fourier Transform). Transformlängden är längden hos argumentet. DFTn är generellt komplex, precis som alla andra fouriertransformer. Plotta absolutbeloppet av dessa DFTer (kommandot `abs`) i linjär skala mot naturlig frekvens. Alltså något sådant här:

```
>> f=fs/N*n;              % Vektor med naturliga frekvensvärden
>> figure; plot(f,abs(fft(x1)))
>> figure; plot(f,abs(fft(x2)))
```

Det underlättar ofta om liknande grafer har samma skala. Det kan du åstadkomma med kommandot `axis` efter att du har skapat en graf.

☞ Besvara följande:

- Förklara vad du ser i dessa plottar.

  .....

  .....

  .....

  .....

- Varför är det två toppar?

  .....

  .....

  .....

  .....

- Hur stor del av dessa spektra är relevant att visa?

  .....

  .....

  .....

  .....

Ofta brukar man presentera spektra i dB. Det är ett logaritmiskt mått som gör det möjligt att jämföra värden som skiljer sig åt med flera tiopotenser i en och samma skala. Ett amplitudspektrum $|X(f)|$ uttrycks i dB som $20\log_{10}(|X(f)|)$. Matlab har ett kommando `db` som gör den avbildningen. Plotta detta för de två signalerna `x1` och `x2`, alltså

```
>> figure; plot(f,db(abs(fft(x1))))
>> figure; plot(f,db(abs(fft(x2))))
```

☞ Besvara följande:

- Förklara vad du ser i dessa plottar.

  .....

  .....

  .....

- Det är en tydlig skillnad mellan de två signalernas spektra. Vad kan det bero på?

  .....

  .....

  .....

Signatur:

Fortsättningsvis ska alla spektra i denna laboration plottas i dB-skala på detta vis.

4.3 Fönsterfunktioner

Följande nya Matlab-kommandon används i detta avsnitt:

```
floor, ceil, rectwin, nuttallwin
```

En multiplikation med en fönsterfunktion innebär att en del av signalen är kvar, medan resten av signalen sätts till noll. Detta gör man i praktiken vanligen för att begränsa komplexiteten hos en beräkning. I nästa avsnitt ska vi se vad det har för effekter i frekvensled. *Här ska vi först studera själva fönstren.*

Följande definierar ett rektangulärfönster av längd `M` som är paddat med nollor på båda sidor så att det totalt blir lika många sampel som i de två signalerna `x1` och `x2`, dvs `N`, given att `M` är definierat och inte är större än `N`.

```
>> w=[zeros(1,floor((N-M)/2)),rectwin(M)',zeros(1,ceil((N-M)/2))];
```

Kommandot `rectwin` returnerar ett rektangulärfönster av längd `M`, som en kolumnvektor. Våra signaler är radvektorer, varför vi vill ha vårt fönster som en radvektor. Den fnutt (`'`) som finns i raden ovan transponerar fönstret, så att det blir en radvektor. Plotta

amplitudspektrum för detta fönster i dB-skala för `M=50` och `M=200`. Prova också detta för fönsterfunktionen `nuttallwin` istället för `rectwin`.

Vi vill alltså plotta `db(abs(fft(w)))` för respektive fönster `w`, och vi vill göra det mot naturliga frekvenser `f`.

Fönstret `rectwin` har värdet 1 i alla sina sampel. Det finns flera olika andra fönsterfunktioner som har olika värden i sina sampel. Detta påverkar hur huvudloben och sidoloberna ser ut.

☞ Vi intresserar oss nu för hovudlobens och sidolobernas förhållande till `M`. Besvara därför följande för de olika fönstren:

- Hur bred är huvudloben för följande värden på `M`?

  `rectwin:` 50:..... 200:.....

  `nuttallwin:` 50:..... 200:.....

- Hur breda är sidolobernas bredd för följande värden på `M`? Strunta här i de sidolober som är närmast huvudloben.

  `rectwin:` 50:..... 200:.....

  `nuttallwin:` 50:..... 200:.....

- Skillnaden mellan huvudlobens och högsta sidolobens höjder (i dB), hur stor är den för följande värden på `M`? Tips: Detta ses enklast om du normerar varje spektrum (i dB) genom att subtrahera dess max-värde från hela spektrat.

  `rectwin:` 50:..... 200:.....

  `nuttallwin:` 50:..... 200:.....

- Sammanfattningsvis, vad är skillnaden mellan de olika fönstrena?

  .....

  .....

  .....

  ..... Signatur:

  .....

4.4 Fönstring

Följande nya Matlab-kommando används i detta avsnitt:

```
cos
```

När man använder fönster så multiplicerar man en signal med dem, dvs. fönstret skalar signalens sampel. Nästa steg är därför att använda fönstren på en signal, och då ska det vara följande signal.

```
>> x=sin(2*pi*3999*(1+5e-7*cos(2*pi*100*t)).*t)+sin(2*pi*4099*t)...
+1e-3*sin(2*pi*4400*t);
```

Här används konstanten `pi` som förstås är ett närmevärde för π. De tre punkterna talar om för Matlab att uttrycket fortsätter på nästa rad. Man fönstrar signalen `x` med fönstret `w` genom att multiplicera dem komponentvis. Tecknet `*` betyder i Matlab multiplikation av skalärer, multiplikation av en skalär med en vektor/matris eller matrismultiplikation, beroende på operandernas storlekar. Här ska inte något av det användas. En punkt (`.`) före operatorn talar om för Matlab att den ska operera komponentvis. Alltså ger följande avsett resultat.

```
>> y=x.*w;
```

Kravet för att detta ska fungera är att `x` och `w` är lika stora, alltså lika många rader och lika många kolumner. Resultatet `y` är då lika stor som `x` och `w`.

Fönstra nu signalen `x` med fönsterlängd 500 och 2000. Gör detta med de två fönster som vi tittat på ovan. Och slutligen plotta amplitudspektrum för dessa fyra fall i dB-skala för naturliga frekvenser mellan 3000 och 5000 Hz. Plotta också som en jämförelse amplitudspektrum för `x` i dB-skala.

Vi vill alltså plotta `db(abs(fft(y)))` mot naturliga frekvenser `f`, med `y` enligt ovan.

☞ Besvara följande:

- Hur bra är de olika fallen på att återge ursprungssignalens spektrum?

  .....

  .....

  .....

- Varför blir det på det viset? Tips: Multiplikation i tidsled motsvarar faltning i frekvensled.

  .....

  .....

  .....

- Är det någon detalj ur ursprungssignalens spektrum som inte syns i något av de fönstrade fallen?

  .....

  .....

  .....

- Är det någon detalj som framkommer i något av de fönstrade fallen som inte ens syns i ursprungssignalens spektrum? Tips: Även ursprungssignalen kan ses som fönstrad.

  .....

  .....

  .....

Slutligen, i en praktisk tillämpning skulle vi bara använda just de fönstrade samplen och inte alla de som är noll. Och så bestämma FFTn i det fallet. Gör nu det för de fyra fallen.

Här vill vi ur den fönstrade signalen plocka ut just de `M` sampel som motsvarar de möjligen nollskilda samplen i mitten. Vi får loss de samplen genom att indexera från `floor((N-M)/2)+1` till `floor((N-M)/2)+M`. Sedan är vi intresserade av dess spektrum i dB-skala, och det ska plottas mot naturlig frekvens. Men! Vi har nu ett spektrum bestående av `M` sampel istället för `N` som tidigare. I uppgiften har vi valt dessa så att `M` delar `N`. Sampelfrekvensen är fortfarande 40 kHz. Vi behöver därför `M` frekvenser jämnt fördelade över 40 kHz. Det åstadkommer vi enklast med `downsample(f,N/M)`, vilket behåller vart `N/M`-te sampel av `f` och kastar alla andra. Med `y` som ovan, så vill vi alltså titta på följande för alla fallen:

```
>> z=y(floor((N-M)/2)+1:floor((N-M)/2)+M);
>> figure; plot(downsample(f,N/M),db(abs(fft(z)))
```

☞ Besvara följande:

- Försämrar detta situationen?

  .....

  ..... Signatur:

  .....