Name:	
ID number:	LiU-ID:
Passed:	Date:

TSBB16 Datorövning B
Prediktion
Transmission av digitala signaler

Utvecklad av Klas Nordberg

Computer Vision Laboratory, Linköping University, Sweden

16 september 2015

Introduktion

I den här övningen kommer du att jobba med två grundläggande problem inom signalbehandling. Del I handlar om prediktion: hur värdet av en tidsdiskret signal vid tiden k kan predikteras från kända värden av samma signal vid tidigare tidpunkter $k-1$, $k-2$, osv. Den enklaste typen av prediktion är linjär prediktion, där prediktionskoefficienterna kan bestämmas från kända signalsekvenser. Del II handlar om hur en digital signal, en sekvens av ettor och nollor, kan skickas över en kanal och återskapas till samma digitala signal. Kanalen består av en högtalare och en mikrofon som sitter i ändarna av ett platsrör.

1 Förberedelseuppgifter

Innan du kommer till datorövningen är det viktigt att du har en klar uppfattning om det tillhörande kursmaterialet: kapitel C och D i kompendiet (motsvarande föreläsningarna 4 och 5, lektion 3 och 4).

Nedan finns ett antal förberedelseuppgifter som ska besvaras innan du deltar i datorövningen. Förutom föreläsningar och lektioner behöver du läsa och sätta dig in i de olika övningarna i denna guide för att kunna besvara en del av uppgifterna.

1.1 Del I: Prediktion

1. Hur bestäms det predikterade sigalvärdet $s_{pred}[k]$ från den tidsdiskreta signalen $s[k]$ vid linjär prediktion med n prediktionskoefficienter? *Ledning:* se inledningen av kapitel D.
SVAR:

2. Prediktion av en tidsdiskret signal $s[k]$ kan beskrivas som ett minstakvadratproblem. Hur ser felet ut som ska minimeras? *Ledning:* se inledningen av kapitel D.
SVAR:

3. Vad är det för variabler som kan varieras i detta minimeringsproblem? *Ledning:* se inledningen av kapitel D.
SVAR:

4. Hur beräknas det optimala värdet av dessa variabler? *Ledning:* se inledningen av kapitel D.

SVAR:

5. I allmänhet, vad borde hända med prediktionsfelet när n ökar? Varför?

SVAR:

6. Rita ett blockschema för en sändare respektive mottagare som använder förlustfri prediktion. Rita in var du hittar prediktionsfelet och var den predikterade signalen finns, i båda fallen både för sändaren och mottagaren. *Ledning:* se tillämpningsdelen av kapitel D.

SVAR:

7. Varför har även sändaren en prediktor? Den har ju tillgång till signalen och skulle inte behöva prediktera den.

SVAR:

8. Rita ett blockschema för en sändare respektive mottagare som använder förstörande prediktion. *Ledning:* se tillämpningsdelen av kapitel D.

SVAR:

9. Hur många bitar behövdes för att representera samplen i en ljudsignal innan det låter illa? *Ledning:* Vad fick du för resultat vid laboration A?)

SVAR:

1.2 Del II: Transmission av binära signaler över en kanal

10. I avsnitt avsnitt 5.2 kommer du att mäta bruset i en överföringskanal. Hur måste den signal som ska skickas genom kanalen vara beskaffad i förhållande till bruset för att transmissionen ska fungera bra?

SVAR:

11. I detta avsnitt kommer du att mäta upp kanalens amplitudkarakteristik. Vad är den förväntade karakteristiken under antagandet att röret har längd $L = 60$ cm, ljudets hastighet är 340 m/s, mikrofonen och högtalaren sitter i ändarna och ljudet reflekteras förlustfritt mot ändarna?

SVAR:

12. Vid frekvensmodulering representeras de två symbolerna "0" och "1" med cosinussignaler multiplicerade med en gaussisk envelopp. Hur skiljer sig pulserna år? *Ledning:* se avsnitt C.2.

SVAR:

13. De två olika pulserna detekteras med hjälp av matchande filter. Hur definieras impulssvaret för ett filter som matchar en specik puls? *Ledning:* se avsnitt 5.3

SVAR:

14. Hur säkerställs att en puls som representerar "0" inte detekteras efter filtret som är matchat för "1"?

SVAR:

2 Matlab-funktioner

En lista med Matlab-funktioner som används i övningen:

- `zeros`: Skapar en vektor eller matris med alla element $= 0$.
- `ones`: Skapar en vektor eller matris med alla element $= 1$.
- `figure`: Skapa ett nytt fönster att rita figurer i.
- `plot`: Rita upp en signal eller funktion. Använd zoom-funktionen (+) för att titta närmare på detaljer i signalen.
- `subplot`: Delar upp ett fönster i flera underfönster som var och en kan användas i `plot`-funktionen.
- `hold`: Lås uppritad figur i ett fönster så att ny ritning kan ske "ovanpå", utan att radera.
- `reshape`: Omforma en matris eller vektor till en matris med specifik storlek.
- `title`: Skriv ut en rubrik på en figur.
- `flipud`: Flippa ordningen på raderna i en matris, så att första radan hamnar sist och tvärtom.
- `liu_NPrevSamples`: Extraherar n tidigare sampel ur en signal $s[k]$, som en kolumn $\mathbf{b}[k]$.
- `liu_quantize`: Kvantiserar elementen i en vektor/matris så att de kan representeras med visst antal bitar inom ett visst intervall.

Om du känner dig osäker på hur dessa funktioner används, titta i Matlabs dokumentation av dessa funktioner innan du gör övningen.

Använd möjligheten att placera den Matlab-kod som som du själv producerar i denna övning i en skriptfil (.m fil) som du kan editera och köra valda delar av från Matlab.

Del I

Prediktion

I denna del av laborationen ska du undersöka hur en tidsdiskret signal kan predikteras från föregående samplen med en linjär metod. Först undersöks en signal i form av en växelkurs och sedan en ljudsignal. I det senare fallet ska du även undersöka hur prediktion kan användas för att minska antalet bitar som behöver överföras när en ljudsignal skickas från en sändare till en mottagare.

3 Prediktion av en växelkurs

Matlab-koden som används i denna del av övningen finns i en skript-fil som heter `prediktion.m`. Ladda in den i Matlab-editorn och titta igenom koden. Följande är en genomgång av de olika delarna i skriptet.

Den tidsdiskreta signalen `bitcoin.mat` laddas in och och ritas upp

```
load bitcoin
signal = bitcoin;
figure(1);plot(signal);
title('Signal som ska predikteras');
```

Signalen har 1240 sampel och består av växelkursen för den digitala valutan bitcoin gentemot dollarn (dollar per bitcoin) under en längre tidsperiod. Varje sampel visar medelvärdet av växelkursen under ett dygn. I denna del av laborationen ska du undersöka möjligheten att prediktera växelkursen för ett specifikt dygn baserat på samma kurs under de n föregående dygnen.

3.1 Prediktion baserat på ett föregående sampel

Som en första övning ska du prediktera växelkursen baserar på endast ett tidigare sampel, från dygnet innan.

```
n = 1;
```

Skapa först en radvektor `a` med $m = 1239$ element, där varje element är en växelkurs som ska predikteras. Eftersom prediktionen är baserad på *ett* tidigare känt värde måste du hoppa över det första värdet i `signal`, som ju inte har något föregående värde att predikteras från.

```
interval = (n+1):length(signal);
```

```
a = signal(interval);
```

På motsvarande sätt skapar du en sekvens $m = 1239$ element, där varje element är det värde från vilket prediktionen görs. Här måste du hoppa över sista elementet i `signal` som ju inte har något efterföljande värde att prediktera.

```
b = liu_NPrevSamples(signal,1);
```

Här används funktionen `liu_NPrevSamples` för att konstruera `b` som en $n \times m$ matris, där varje kolumn innehåller n på varandra följande sampel i sekvensen (senaste först) som ska användas för att prediktera efterföljande sampel, som ligger i motsvarande element i `a`. Trixet är inte så viktigt nu, när $n = 1$, men blir viktigare i nästa övning när $n > 1$.

Bestäm prediktionskoefficienten `c` och den predikterade signalen

```
c = inv(b*b')*b*a';
a_pred = c'*b;
```

Rita upp signalen tillsamman med den predikterade signalen, rita upp prediktionsfelet och även prediktionkoefficienterna (bara en i detta fall).

```
figure(2);plot(interval,a,'b',interval,a_pred,'r');
title('Signalen (blå), predikterad signal (röd)');
figure(3);plot(interval,a-a_pred);
title('Differens mellan signal och predikterad signal');
figure(4);plot(c,'-o');
title('Prediktionskoefficienter');
```

Slutligen, beräkna standardavvikelsen av felet

```
fprintf('Standardavvikelse för felet %f\n',std(a-a_pred));
```

Kör skript-filen `prediktion` och observera resultatet.

FRÅGA: Vilket värde har prediktionskoefficienten `c`? Hur tolkar du det?
SVAR:

Titta i figur 3 och 4 samt notera värdet av prediktionsfelets standardavvikelse.

FRÅGA: Hur bra är prediktionen? Generellt, när blir den bra? Dålig? Varför?

SVAR:

3.2 Prediktion baserat på två föregående sampel

Du ska nu göra samma sak som ovan, men prediktera ett sampel från de två föregående i sekvensen. Ända i skript-filen så att det nu står:

```
n = 2;
```

I detta fall blir `b` en $2 \times m$ matris, där varje kolumn innehåller två på varandra följande sampel i sekvensen som ska användas för att prediktera efterföljande sampel, som ligger i motsvarande element i `a`. Eftersom det finns $n = 2$ finns det två element i den ursprungliga signalen som inte kan predikteras, dvs. $m = 1238$.

Kör skriptet på nytt och titta på de två prediktionskoefficienterna.

FRÅGA: Vad får koefficienterna för värde? Hur tolkar du detta resultat?

SVAR:

FRÅGA: Hur skiljer sig prediktionsfelets standardavvikelse mot undersökningen i avsnitt 3.1? Verkar det rimligt?

SVAR:

3.3 Prediktion baserat på n föregående sampel

Prova flera olika värden på $n > 2$ upp till $n = 30$. Det senare fallet motsvaras alltså av att prediktera växelkursen baseras på kursen under den senaste månaden.

FRÅGA: Finns det något intressant mönster i prediktionskoefficienterna?

SVAR:

FRÅGA: Är detta ett praktiskt tillämpart sätt att prediktera växelkursen?

SVAR:

FRÅGA: Kan du komma på någon annan typ av tidsdiskret signal inom ekonom/logistik där prediktion av denna typ är användbar?

SVAR:

3.4 Icke-linjär prediktion

Extra

Vid linjär prediktion bestäms det predikterade värdet i `a_pred` som en linjärkombination av de n tidigare samplen. Vid icke-linjär prediktion bestäms det predikterade värdet som en olinjär funktion av samma sampel. Generellt kan den olinjära funktionen definieras på många olika sätt, men ett enkelt sätt är att lägga till polynom av gradtal > 1 som element i kolumnerna i `b`, från vilken prediktionen bestäms. Det predikterade värdet blir då fortfarande en linjärkombination av data från signalen, men dessa data är inte linjära funktioner av signalens sampelvärden.

Prova att komplettera de linjära termerna i `b` med andragradstermer, innan prediktionskoefficienterna i `c` beräknas:

```
b = [b;b.^2];
```

och kör skriptet igen

FRÅGA: Vad blir det för skillnad på prediktionsfelet jämfört med linjär prediktion? Varför?
SVAR:

Prova att även lägga till termer av ordning tre och högre i `b`.

FRÅGA: Vad händer? Varför?
SVAR:

Prova att lägga till andra typer av olinjära termer i `b` så att prediktionsfelet blir så litet som möjligt.

FRÅGA: Vilka valde du? Vad hände? Varför?
SVAR:

4 Prediktion av ljudsignal

Prediktion kan appliceras på i stort sett vilken typ av signal som helst, även om det inte alltid leder till användbara prediktioner. Ett andra exempel på signal som kan predikteras är ljud. En tillämpning i detta sammanhäng är att reducera antalet bitar som används för att representera ljud som skickas från en sändare till en mottagare. Det kan göras genom att både sändare och mottagare predikterar nästa sampel i ljudsekvensen och att det enda som behöver överföras från sändaren till mottagaren är felet i prediktionen, snarare är själva signalen.

De signaler du ska undersöka är ljudsignaler som ligger i en cellarray `ljud1` (samma som du använt i tidigare laboration).

```
load ljud1
```

4.1 Förlustfri prediktiv kodning

Modifiera ditt prediktionsskript så att det använder någon av ljudsignalerna (istället för `bitcoin`), exempelvis det sista:

```
signal = ljud1{10};
```

Undersök prediktion av denna signal för olika värden på $n = 1, \ldots, 10$. Anteckna standardavvikelsen av prediktionsfelet för varje n.

FRÅGA: Hur beter sig prediktionsfelets standdardavvikelse när n ökar? Är det som förutsett?
SVAR:

FRÅGA: Hur stort verkar det rimligt att välja n? Varför?
SVAR:

I resten av denna övning ska du använda $n = 3$. Sätt denna parameter i ditt skript och kör om beräkningen av prediktionskoefficienterna. Ändra sedan inte värdet på `n` och `c` i resten av detta avsnitt.

4.1.1 Sändaren

Du ska nu undersöka en sändare av ljudsignalen, som inte sänder själva ljudsignalen utan istället skickar prediktionsfelet till mottagaren. Baserat på prediktionsfelet kan sedan mottagaren rekonstruera ljudsignalen, även det med hjälp av prediktion. Matlab-koden för sändaren finns i skript-filen `transpred.m`. Vi går nu igenom

vad som händer när sändaren körs.

För att prediktionen ska fungera måste all prediktion initialiseras med kända värden, exempelvis noll. Därför läggs n sampel med värdet noll i början av signalen.

```
signal = [zeros(1,n) ljud1{10}];
```

I övrigt sker prediktionen på samma sätt som tidigare:

```
interval = (n+1):length(signal);
a = signal(interval);
b = liu_NPrevSamples(signal,n);
a_pred = c'*b;
d = a - a_pred;
```

Vektorn `d` kommer att innehålla prediktionsfelet, det sparas på en fil som mottagaren kan läsa. Felet kan även spelas upp som ett ljud.

```
save('pred_diff','d');
sound(d,44100);
```

FRÅGA: Hur låter prediktionsfelet?
SVAR:

Observera att prediktionsfelet innehållet n färre sampel än signalen, eftersom den enbart bestämmer felet efter de n första samplen (som är = 0) i signalen.

4.1.2 Mottagaren

Matlab-koden för mottagaren finns i en skript-fil `receiverpred.m`. Vi går nu igenom vad som händer när mottagaren körs.

På mottagarsidan behövs först och främst prediktionsfelet som "skickats" från sändaren:

```
load 'pred_diff'
```

Dessutom behövs samma initialisering av den rekonstruerade signalen, samma som på sändarsidan:

```
s_rek = zeros(1,length(signal));
```

Det är egentligen bara de n första elementen i `s_rek` som ska sättas till noll, de övriga kommer att rekonstrueras genom att först göra en preliminär prediktion, `s_pred`, som sedan justeras med motsvarande prediktionsfel som kommer från sändaren.

```
for k = 1:length(d),
  s_pred = fliplr(s_rek((k):(k+n-1)))*c;
  s_rek(k+n) = s_pred + d(k);
end
```

Kör skriptet för mottagaren och kontrollera att den signal som rekonstrueras av mottagare är identisk med den ursprungliga signalen som sändaren matas med

```
norm(signal-s_rek)
```

FRÅGA: Stämmer det?
SVAR:

Om det inte stämmer finns det ett fel någonstans i din kod som behöver åtgärdas innan du går vidare!

I den typ av överföring är det prediktionsfelet som skickas från sändaren till mottagaren.

FRÅGA: Varför tror du att den kallas förlustfri kodning?
SVAR:

4.2 MÄSTARPROV: Förstörande prediktiv kodning

Att sända prediktionsfelet istället för själva signalen innebär i sig ingen större vinst, om felet representeras med full numerisk upplösning (många bitar). En reduktion

av bitar som transmitteras kan ske genom kvantisering av prediktionsfelet.

Prova först med att kvantisera rekonstruktionsfelet till ett begränsat antal bitar, börja exempelvis med $b = 5$ bitar.

4.2.1 På sändarsidan (första försöket)

Titta på storleken av prediktionsfelet för att bestämma `r`, det område som felet kommer att kvantiseras med `b` bitar. Modifiera sändaren så att den kvantiserar prediktionsfelet till b bitar innan det sparas på fil.

```
d = liu_quantize(d,r,b);
```

Kör sedan sändaren.

4.2.2 På mottagarsidan

Kör mottagaren på samma sätt som tidigare. Lyssna på den rekonstruerade signalen

```
sound(s_rek,44100)
```

FRÅGA: Låter det bra? Varför?
SVAR:

4.2.3 På sändarsidan (andra försöket)

För att åstadkomma en bättre rekonstruktion på mottagarsidan kan sändaren skicka ett prediktionsfel baserat på en jämförelse mellan den verkliga signalen och den rekonstruktion som mottagaren kan åstadkomma med hjälp av det kvantiserade prediktionsfelet. Det betyder att sändaren behöver kunna utföra samma rekonstruktion som görs på mottagarsidan, inklusive kompenseringen med det kvantiserade prediktionsfelet, och jämföra det med den inkommande signalen.

För att få till allt detta behövs en helt ny version av sändaren, som finns i skript-filen `transpred2`. Den innehåller tre tidsdiskreta signaler:

- `s_pred`: prediktionen av insignalen,

- `d`: prediktionsfelet som kommer att vara kvantiserat.
- `s_rek` en korrektion av prediktionen med prediktionsfelet i `d`.

Prediktionsfelet `d` är n sampel kortare än `s_pred` och `s_rek` eftersom de senare måste initialiseras med n nollor.

```
s_pred = zeros(1,length(signal));
s_rek = zeros(1,length(signal));
d = zeros(1,length(signal)-n);
```

För varje sampel beräknas först prediktionen `s_pred` från den rekonstruerade signalen `s_rek`, på samma sätt som tidigare, därefter bestäms prediktionsfelet `d`, som slutligen adderas till prediktionen för att ge den rekonstruerade signalen `s_rek`.

```
for k = 1:length(d),
  s_pred(k+n) = fliplr(s_rek(k:(k+n-1)))*c;
  d(k) = signal(k+n) - s_pred(k+n);
  s_rek(k+n) = s_pred(k+n) + d(k);
end
```

Notera att det som händer inne i for-loopen är i det närmaste identiskt med vad som görs i mottagaren, med skillnaden att här i sändaren beräknas prediktionsfelet `d`, medan mottagaren endast använder felet för att justera den predikterade signalen. Vidare bestäms prediktionsfelet med full numerisk noggrannhet, det är ännu inte kvantiserat. Slutligen plottas och sparas prediktionsfelet (skickas till mottagaren).

```
figure(5); plot(d,'-o')
save('pred_diff','d');
```

Kör denna kod. Om du har gjort rätt kommer prediktionsfelet att bli detsamma som tidigare. Det betyder att om du även kör mottagaren på samma sätt som tidigare så rekonstrueras samma signal som skickades in.

Kontrollera detta. **Om det inte stämmer finns det ett fel någonstans i din Matlab-kod som behöver åtgärdas innan du går vidare!**

Nu kommer det kluriga: Hittills har prediktionsfelet bestämts med full numerisk precision, men nu ska du kvantisera felet så att det åter representeras med b bitar. Lägg till

```
d(k) = liu_quantize(d(k),r,b);
```

på lämpligt ställe i for-loopen och kör sändaren så att den lagrar det kvantiserade prediktionsfelet. Kör även mottagaren.

FRÅGA: Hur låter det nu? Blev det någon skillnad mot senast du kvantiserade prediktionsfelet? Varför?

SVAR:

Prova att ändra på b i sändaren, dsv. antalet bitar som representerar prediktionsfelet. Kör sändar-skriptet.

FRÅGA: Hur litet kan b bli utan att det låter illa?

SVAR:

FRÅGA: Hur många bitar behövde du använda för att representera motsvarande ljudsignal i föregående laborationen för att det skulla låta bra?

SVAR:

I denna typ av överföring skickas ett kvantiserad prediktionsfel från sändaren till mottagaren.

FRÅGA: Varför tror du att den kallas förstörande kodning?

SVAR:

4.3 Sammanfattnig av del I

FRÅGA: Vad har du lärt dig av detta avsnitt i laborationen? Vad finns det för praktiska tillämpningar av det du har sett hittills?
SVAR:

FRÅGA: Är det något som du tycker är oklart?
SVAR:

Del II

Transmission av binära signaler

5 Transmission av digital information

I denna del av laborationen ska du undersöka hur digital information kan skickas över en kanal. Informationen kan initialt ses som ett meddelande, en sekvens av binära symboler (bitar), där varje symbol kan anta värdet *noll* eller *ett*. Den enhet som ska skicka meddelandet, sändaren, består av en modulator som överför meddelandet till en tidskontinuerlig signal $s_1(t)$, där varje bit i meddelandet motsvaras av en puls. Pulsen har en position i tiden, styrd av var i meddelandet motsvarande bit förekommer, och en kurvform styrd av vilken symbol som ska skickas. Signalen s_1 skickas genom en kanal som med god approximation kan beskrivas som ett LTI-system. Den signal som är insignal till mottagaren kommer dessutom att innehålla en viss mängd brus som adderats till signalen från kanalen.

Mottagaren demodulerar sin insignal och rekonstruerar meddelandet som skickades. På grund av bruset kommer inte varje bit i det rekonstruerade meddelandet att bli identiskt med motsvarande bit i det ursprungliga meddelandet, men målet är att så få bitar som möjligt ska bli fel.

5.1 Utrustningen

Den kanal som ska undersökas för transmission av digital information består av en högtalare och en mikrofon placerade i ett plaströr.

Koppla högtalaren och mikrofonen till datorns utgång respektive ingång för ljud och låt dom sitta kvar där under resten av laborationen. Starta om Matlab. Om sladdarna av någon anledning lossas från sina kontakter på datorn är det sannolikt nödvändigt att starta om Matlab för att allt ska fungera enligt anvisningarna.

Låt röret ligga så att det inte utsätts för stötar eller vibrationer, eller nära ett annat rör under laborationen. Allt externt ljud kommer att tas upp av mikrofonen vilket stör mätningarna.

De Matlab-funktioner som används i denna laboration fungerar endast för version 2014b och senare.

Kontrollera att kablarna som hör till rörets högtalare och mikrofon är kopplade till rätt uttag på datorn. Dra upp högtalarens volym till max i operativ-

systemets högtalarkontroll.

Testa utrustningen genom att köra Matlab-skriptet

```
testscript
```

Den sänder ut en sinussignal till högtalaren och läser av den mottagna signalen vid mikrofonen och ritar upp den mottagna signalen. Den ska tydligt visa en sinussignal plus en viss mängd brus.

Om det inte stämmer finns det ett fel någonstans i kopplingar eller komponenter behöver åtgärdas innan du går vidare!

5.2 Brusmätning

Börja med att undersöka hur mycket brus som kanalen har. Det görs genom att skicka in en signal som är konstant lika med 0 till högtalaren och mäta signalen från mikrofonen. Om kanalen beter sig som ett LTI-system är den mottagna signalen enbart brus.

För att mäta bruset behöver du spela in en kort sekvens av det ljud som mikrofonen tar upp, utan att något sänds ut på högtalaren. De Matlab-funktioner som beskrivs i detta avsnitt används för att spela in ljud. Börja med att sätta samplingsfrekvens och bitar per sampel för inspelningen från mikrofonen:

```
fs = 44100; %Sätt samplingsfrekvens
b = 16; %Sätt antalet bitar
```

Skapa ett "objekt" i matlab för inspelning av en ljudkanal:

```
recObject = audiorecorder(fs, b, 1);
```

Spela in 5 sek av ljud från mikrofonen med "inspelningsobjektet":

```
recordblocking(recObject, 5);
```

Överför det inspelade ljudet till en "vanlig" Matlab-vektor, med ett element per sampel:

```
signal=getaudiodata(recObject);
```

Rita upp den inspelade signalen och beräkna dess standardavvikelse

```
figure(1);plot(signal);
std(signal)
```

FRÅGA: Vad fick du för värde? Varierar det om du provar att spela in bruset flera gånger? Varför?

SVAR:

5.3 Kanalen som ett LTI-system

Du ska nu mäta upp kanalens amplitudkarakteristik[1] $|H(f)|$. Det görs genom att skicka en sinussignal $s(t)$ till högtalaren, med en bestämd amplitud och frekvens f. Om kanalen uppträder som ett LTI-system kommer mikrofonen att ta emot en sinussignal $y(t)$, med samma frekvens som $x(t)$ men en amplitud som förstärkts med $|H(f)|$. Genom att skicka en sekvens av sinussignaler med olika frekvenser, kan $|H(f)|$ bestämmas, för alla de frekvenser som undersöks. Om de ligger tillräckligt tätt kan en approximation av $|H(f)|$ erhållas, som en *frekvensdiskret funktion*.

Börja med att skapa en lista med de frekvenser som ska mätas. Gör mätningen runt den första resonansfrekvens som bestämdes i förberedelseuppgift 11, så att den ligger ungefär i mitten av ett intervall som är 200 Hz långt, med 10 Hz upplösning:

```
frekvenser = ...
```

Mätningen görs automatiskt av en funktion `meassureAmplitudes`:

```
H = measureAmplitudes(frekvenser);
figure(2);plot(frekvenser, H);
```

Här borde du få en topp för en bestämd frekvens. Om inte finns det ett fel i ditt system som måste åtgärdas innan du går vidare.

Mät noggrannare runt den resonansfrekvens du hittar, för att bestämma den med 1 Hz noggrannhet.

FRÅGA: Vid vilken frekvens ligger den första resonansfrekvensen?

SVAR:

[1] Här väljer vi att bestämma frekvens i Hertz, perioder per sekund, snarare än som vinkelfrekvens i radianer per sekund.

Gör motsvarande mätning kring den andra resonans frekvensen och bestäm var den ligger.

FRÅGA: Vid vilken frekvens ligger den andra resonansfrekvensen?
SVAR:

FRÅGA: Stämmer det med teorin?
SVAR:

Upprepa mätningen, fast nu över ett större frekvensintervall och med grövre frekvensupplösning (det tar någon minut...):

```
frekvenser = 100:25:3000
H = measureAmplitudes(frekvenser);
figure(2);plot(frekvenser, H);
```

FRÅGA: Är resultat det förväntade? Varför?
SVAR:

Undersök kanalens karakteristik för väldigt låga frekvenser:

```
frekvenser = 10:10:100
H = measureAmplitudes(frekvenser);
figure(2);plot(frekvenser, H);
```

FRÅGA: Vad blev resultat.

SVAR:

Gör även en avslutande mätning över ett mycket stort frekvensintervall, men med mycket låg frekvensupplösning (det tar någon minut...):

```
frekvenser = 10:100:2000
H = measureAmplitudes(frekvenser);
figure(2);plot(frekvenser, H);
```

FRÅGA: Hur höga frekvenser kan kanalen överföra?

SVAR:

Den kanal som används består dels av datorns ljudkort för upp- och inspelning av ljud, högtalaren, mikrofonen och själva röret med sina akustiska egenskaper.

FRÅGA: Vilka delar tror du bidrar till att H ser ut som den gör?

SVAR:

FRÅGA: Givet den grova bild av amplitidkarakteristiken som du nu har mätt upp, vilket frekvesområde bör de signaler som används för överföring av binära signaler ha för att effektivt transmitteras?

SVAR:

5.4 Unipolär modulation

Ett enkelt sätt att modulera binära meddelanden är att översätta symbolen *ett* till en fyrkantspuls, med en viss tidslängd T_p och amplitud 1, och symbolen *noll* översätts till en lika lång puls som har amplituden 0. Här kan T_p sättas till 0,25 s. Modulera ett kort meddelande med 8 bitar enligt denna metod:

```
sampleRate = 44100;
Tp = 0.25 * sampleRate;

data = [1 0 1 1 0 0 1 0 1 0];
pulser = {zeros(1,Tp), ones(1,Tp)};
signalx = modulateData(data,pulser);
```

`signalx` är den modulerade ljudsignal som ska skickas till högtalaren. Kontrollera att den överensstämmer med bitarna i `data`.

```
figure(8);plot(signalx,'LineWidth',2);
```

Notera att funktionen `modulateData` lägger till en sekvens med nollsatta sampelvärden i början och slutet av den modulerade signalen.

För att Matlab ska kunna hantera att både spela upp ljud och spela in ljud samtidigt används dessa kommandon:

```
soundObject = audioplayer(signalx, sampleRate);
play(soundObject);

recObject = audiorecorder(sampleRate, 16, 1);
recordblocking(recObject, length(signalx) / sampleRate + 0.5);
signaly = getaudiodata(recObject);
```

Den inspelade signalen `signaly` blir 0,5 sekunder längre än den uppspelade. Rita upp den inspelade signalen

```
figure(9);plot(signaly);
```

Eftersom de två sekvenserna `signalx` och `signaly` har olika antal sampel kan det vara svårt att jämföra dom, men försök lägga figurfönstren för `siganlx` och

`signaly` den ena ovanför den andra och förläng dom så att första och sista pulsen i `siganlx` och i `signaly` ungefär sammanfaller.

Det som syns i dessa figurer är att signalen från mikrofonen innehåller en slags pulser, men inte samma sort som finns i den signal som skickas till högtalaren.

FRÅGA: Hur skiljer sig pulserna som sänds åt från de som tas emot?
SVAR:

FRÅGA: Vilken frekvens har oscillationerna i de pulser som tas emot?
SVAR:

FRÅGA: Denna frekvens har dykt upp tidigare i labben. Var? Vad finns det för samband?
SVAR:

De pulser som du ser i den mottagna signalen kallas kanalens *stegsvar*. Stegsvaret är en beskrivning av kanalens överföringsegenskaper som är lika specifikt som dess amplitudkarakteristik $|H(f)|$.

FRÅGA: Var i den mottagna signalen hittar du information om det skickade meddelandet?

SVAR:

I detta experiment tar varje bit $T_p = 0,25$ sekunder att överföra. Det motsvarar en överföringshastighet på 4 bitar per sekund. För att överföra 100 bitar per sekund skulle vi alltså behöva sätta $T_p = 0,01$ s.

```
Tp = 0.01 * sampleRate;
```

Gör den ändringen och kör skriptet igen

FRÅGA: Hur ser den mottagna signalen ut nu?

SVAR:

FRÅGA: Vilka observationer kan du göra av detta experiment? Skulle det fungera bra att använda unipolär modulation av en binär signal? Motivera svaret.

SVAR:

5.5 Frekvensmodulation

Du ska nu modifiera de pulsformer som används för att representera symbolerna *noll* respektive *ett*. Varje symbol representeras nu av en puls som består av en gauss-funktion multiplicerad med en sinusvåg. Gauss-funktionens bredd (standardavvikelse) bestämmer pulsens effektive längd, och sinusvågens frekvens bestämmer vilken symbol som sänds, symbolerna noll respektive ett motsvaras av två olika frekvenser.

```
k = -700:700;
sigma = 300;
frek0 = 1000;
frek1 = 2000;
puls0 = exp(-(k/sigma).^2) .* cos(2*pi*frek0*k/sampleRate);
puls1 = exp(-(k/sigma).^2) .* cos(2*pi*frek1*k/sampleRate);
```

I övrigt görs samma sak som innan. Kör skriptet.

FRÅGA: Vad blir det för skillnad mellan frekvensmodulation och unipolar modulation, med de givna parametrarna, specifikt vad gäller hur de mottagna pulserna liknar de utsända?
SVAR:

FRÅGA: Verkar det som att det finns bättre möjlighet att skärskilja de olika symbolerna från varandra i detta fall?
SVAR:

5.6 Startpuls

I nästa steg ska den mottagna signalen demoduleras och den ursprungliga sekvensen av binära symboler rekonstrueras. För att det ska fungera bra behöver mottagaren veta var starten på den utskickade sekvensen finns. Det görs genom att sändare lägger en startpuls, som tydligt skiljer sig från pulserna för symbolerna noll och ett, i början av signalen på ett bestämt avstånd från de övriga pulserna. Bredden på startpulsen är inte så viktig, den kan vara relativt lång, eftersom den endast sänds en gång per meddelande så påverkar den inte signifikant överföringshastigheten.

Här används emellertid samma pulsbredd som tidigare. Istället för en specifik frekvens har startpulsen en varierande frekvens längs med pulsen, den går från frekvensen `chirpf0` till `chirpf1` under pulsens längd. En signal av denna typ brukar kallas för en *chirp*.

```
k = -700:700;
sigma2 = 300;
chirpf0=500;
chirpf1=2000;
chirpf = (chirpf0+chirpf1)/2+k*(chirpf0-chirpf1)/(k(end)-k(1));
chirp = exp(-(k/sigma2).^2) .* cos(2*pi*chirpf.*k/sampleRate);
startpuls = [chirp zeros(1,3*length(chirp))];
```

Generera sändarsiganlen som nu även ska innehålla startpulsen:

```
signalx=modulateData(data,pulser,length(puls0),startpuls,length(puls0));
```

Kör skriptet och titta på den mottagna signalen

FRÅGA: Ser det fortfarande ut som en sekvens av väl separerade och distinkta pulser?
SVAR:

FRÅGA: Pulserna som sickas till högtalaren på sändarsidan har alla samma amplitud. De har de inte på mottagarsidan. Varför?
SVAR:

FRÅGA: Vad är den ungefärliga genomsnittliga förstärkningen från sändare till mottagare för de pulser du ser?
SVAR:

5.7 Demodulering

För att demodulera eller avkoda signalen faltas den med de tre olika pulsformerna som används. Prova att göra denna faltning på den mottagna signalen;

```
figure(18);
subplot(3,1,1);plot(conv(signaly,chirp));
subplot(3,1,2);plot(conv(signaly,puls0));
subplot(3,1,3);plot(conv(signaly,puls1));
```

FRÅGA: Hur kan du beskriva de tre resulterande signalera?
SVAR:

Den praktiska demoduleringen görs i en funktion som faltar med pulserna, enligt ovan, och jämför vid varje puls vilken symbol som har störts utslag:

```
decodedMsg = demodulateSound(sampleRate, signaly, pulser, length(puls0),
```

Jämför det rekonstruerade meddelandet med det skickade

FRÅGA: Är de lika?
SVAR:

Om de inte är lika finns det ett fel någonstans i ditt system. Fixa det innan du går vidare!

FRÅGA: Vad har systemet för överföringshastiget nu?
SVAR:

FRÅGA: Vad har systemet för felfrekvens (felaktiga bitar/totala antet bitar)
SVAR:

5.8 Mästarprov

Denna del av laborationen avslutas med ett mästarprov. Mästarprov A är obligatoriskt, du kan sedan välja mellan de övriga.

5.8.1 A: ökad överföringshastighet

Med hjälp av de Matlab-skript som du redan har använt ska du nu försöka höja överföringshastigheten på det digitala kommunikationssystemet.

Det finns ett antal strategier som kan leda till ökad överföringshastighet

- Korta ned tiden T_p som varje puls tar. För att göra det måste även pulsformerna för de olika symbolerna ändras så att de "får plats" inom intervallet T_p. Det betyder att både pulserna bredd och frekvens behöver modifieras.
- Utöka alfabetet för de symboler som moduleras, exempelvis till de fyra symbolerna $\{00, 01, 10, 11\}$. Varje symbol motsvaras av en distinkt pulsform.

Givetvis är det även möjligt att kombinera dessa två strategier. Målet är att minst fördubbla överföringshastigheten i bitar per sekund relativt det som uppmättes i föregående övning utan att felfrekvensen överstiger 5%.

I ett verkligt system skulle någon typ av felrättande kodning användas för att ta hand om de felaktiga bitarna, men detta behandlas inte i denna övning.

5.8.2 B: Skicka ett SMS över kanalen

Skriv ett kort meddelande i form av en text, max 140 tecken. Översätt varje tecken till motsvarande ASCII-kod, en sekvens av 8 bitar:

```
ascii = unit8(string);
message = de2bi(ascii,8);
message = message(:)';
```

Skicka bitarna över kanalen och rekonstruera meddelandet.

```
rek_message = reshape(rek_message,length(message)/8,8);
rek_string = char(bi2de(rek_message)');
```

5.8.3 C: Skicka ljud över kanalen

Den första delen av laborationen avslutades med att en ljudsignal skickades genom en förstörande prediktiv kodare. Den genererar ett prediktionsfel som är kvantiserat till ett fåtal antal bitar, b. I detta mästarprov ska en ljudsignal först kodas med den prediktiva kodare, skickas över kanalen, och slutligen rekonstrueras till en ljudsignal.

5.8.4 D: Signalstyrkan

Undersök hur låg amplitud som signalen kan ha innan mottagaren börjar blanda ihop den med bruset.

Signalstyrkan kan ändras genom att den modulerade signalen förstärks med en faktor, som i detta fallet ska vara < 1. Undersök hur liten denna faktor kan bli innan du får en signifikant påverkan på felfrekvensen.

FRÅGA: Vid vilket värde på den utsända signalens amplitud får du signifikant mycket fel?
SVAR:

FRÅGA: Är det rimligt givet den ungefärliga förstärkning som kanalen har och den brusnivå du mätt upp tidigare?
SVAR:

5.9 Sammanfattning av del II

FRÅGA: Vad har du lärt dig av detta avsnitt i laborationen? Vad finns det för praktiska tillämpningar av det du har sett hittills?
SVAR:

FRÅGA: Är det något som du tycker är oklart?
SVAR: