



KTH Datavetenskap
och kommunikation

Spektrala transformer

Laboration: DTMF-detektor

1 Introduktion

I denna laboration kommer du att bygga en detektor för att identifiera DTMF-signaler, dvs de tonsignaler som används för signallering i telenätet (tonvalstelefoner). Ditt program ska kunna ta in en ljudfil med DTMF-pulser (med godtycklig samplingshastighet), analysera den, och returnera korrekt knapp-sekvens som ASCII-tecken. Slutligen ska du även se till att detektorn blir robust mot måttliga störningar i signalen i form av brus.

2 DTMF-signallering

En DTMF-signal består av två överlagrade sinustoner. På telefonens knappsats motsvarar den lägre frekvensen knappens rad och den högre frekvensen knappens kolumn, se tabell 1. Frekvenserna är ordnade i två icke-överlappande grupper, den låga och den höga gruppen, där den förra motsvarar raden och den senare kolumnen. Signalerna för A, B, C och D används sällan.

Vidare finns specifikationer för timingen hos DTMF-signaler: längden på varje ton måste vara minst 40 millisekunder, och mellan varje ton måste det vara minst 40 ms tystnad.

3 Utförande

Du ska utföra ett antal uppgifter med hjälp av Matlab, och svara på de frågor som ställs i peket. Laborationen utföres självständigt, antingen enskilt eller i grupp om två. Börja med att hämta `lab-dtmf.zip` från kurshemsidan. Denna fil innehåller några hjälpfunktioner och andra filer som du kommer att behöva. Packa upp arkivet i din hemkatalog och använd detta som “working directory” i matlab. Under `prototypes/` finner du även platshållare för de funktioner du själv ska skriva. Det är filer som endast innehåller funktionshuvuden, där du själv ska fylla på med kod.

I matlabkoden i peket betecknas alltid arrayer och matriser med versaler, medan skalärer betecknas med gemener. Vidare används genomgående konventionen att skriva all kod med

Tabell 1. Frekvenser för DTMF-signallering

	1209 Hz	1336 Hz	1477 Hz	1633 Hz
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

skrivmaskinsstil. Den kod som efterfrågas (och ska redovisas) ska oftast vara i form av matlabfunktioner. Ofta beskrivs även hur du kan testa att funktionen gör vad den ska. Du gör klokt i att skriva dessa tester i m-filer (behöver *ej* vara funktioner) för att göra själva testandet rationellt och konsekvent.

Var noga med att dokumentera vad du gör och spara kod, plottar, bilder, ljudfiler och vad det kan vara till redovisningstillfället, då du ska redogöra för hur du kommit fram till dina resultat.

Uppgift 1: Manuell analys och segmentering

Börja med att titta på en DTMF-sekvens. Öppna filen `dtmf_all.wav` (den innehåller samtliga 16 signaler) i *WaveSurfer* och studera spektrogram och spektrum (*Spectrum section*). Försök uppskatta vilken analysfönsterlängd (FFT-längd) som krävs för att kunna separera frekvenserna ordentligt. Vad händer om du väljer för långt analysfönster?

Passa även på att dela upp (segmentera) signalen i segment som motsvarar DTMF-tonpulser och spara i en egen wav-fil som du sedan ska läsa in och analysera i matlab. (I slutet av labben finns en extra uppgift där du ska göra detta moment automatiskt, baserat på signalens energiinnehåll)

Att redovisa: kommentarer

WaveSurfer-tips

I *Spectrum Section*-fönstret kan du jämföra två spektra genom att ta ett "snapshot" (med snapshot-menyn) och sedan flytta tidsmarkören.

3.1 En DTMF-transform

Det viktigaste elementet i DTMF-detektorn är givetvis att kunna detektera förekomsten av de ingående frekvenserna i en DTMF-signal. Ett sätt att göra detta är göra en spektral analys med DFT (FFT), och sedan identifiera topparna i spektrum. Detta blir dock ett ganska ineffektivt sätt att lösa problemet på, för att DTMF-frekvenserna ligger relativt tätt, vilket ställer krav på hög upplösning i DFT'n – man måste räkna fram ett stort antal frekvenspunkter för att sedan slänga bort alla utom åtta av dessa. DFT'n ges som bekant av

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-jnk2\pi/N}$$

där

$$e^{jnk2\pi/N} \quad (n = 0, 1, 2 \dots N - 1)$$

är den k :te basvektorn, som motsvarar frekvensen $f_k = kf_s/N$.¹

När vi vet exakt vilka frekvenser vi letar efter, kan vi göra ett mycket bättre jobb än den generella DFT'n genom att skraddarsy vår egen transform. Vi vill begränsa vår analys till exakt de åtta frekvenser som ingår i DTMF-specifikationen, genom att skapa egna basvektorer. Vinsten med detta blir inte bara att vi endast behöver beräkna 8 istället för N frekvenspunkter, vi kan dessutom beräkna *exakt* de frekvenser vi söker. Hade vi använt DFT'n fick vi nöja oss med den frekvens som låg närmast, vilket ger sämre precision i detektionen.

¹Notera att DFT ekvationen beskriver en skalärprodukt av två komplexa vektorer – insignal och basvektor – i vilken basvektorns komponenter skall vara komplexkonjugerade. Därav minustecknet i exponenten.

Utifrån uttrycket för DFT'ns basvektorer inses att en basvektor motsvarande frekvensen ψf_s beskrivs med

$$e^{jn2\pi\psi}$$

Givet basvektorerna, erhålls vår DTMF-transform sedan som den inre produkten (skalärprodukten) mellan basvektorer och insekvensen $x(n)$ – helt i analogi med definitionen av DFT'n ovan. För DTMF-transformen kan vi välja N helt fritt - ett högre värde ger bättre precision i detektionen av toner, till priset av sämre tidsupplösning och ökad beräkningstid.

Notera en viktig skillnad mellan DTMF-basen och DFTn: I DTMF-transformen är basvektorerna inte ortogonala, eftersom antalet perioder inte går jämnt upp (DTMF-frekvenserna är inte jämna multiplar av någon gemensam grundfrekvens). Detta är dock inte något problem eftersom vi inte kommer att behöva beräkna transformens invers (vi ska bara göra analys - inte syntes).

Uppgift 2

Skriv en Matlab-funktion som beräknar de komplexa basvektorerna för DTMF-transformen, enligt nedanstående prototyp. Vektorerna ordnas som kolumner i en matris.

Notera: en konvention som används genomgående i dessa labbar är att skriva matriser och vektorer med versaler och skalärer med gemena. N i formlerna ovan motsvaras därför av n i funktionen.

```
function B = dtmf_basis(n,fs)
%
% dtmf_basis - calculate basis vectors for a DTMF-transform
%
% input:
%   n (real number) - Basis vector length
%   fs (real number) - Sampling frequency
% output:
%   B = (nx8 matrix) - 8-column complex matrix where each column
%                       is a basis vector
```

Studera basvektorerna i den resulterande matrisen genom att plotta real- och imaginärdel för varje kolumn. Alternativt kan du studera realdelen av hela matrisen på en gång som en bild med hjälp av `imagesc()`. Välj ett värde på N så att flera hela perioder av den lägsta frekvensen kommer med.

Att redovisa: m-fil

Algebra-tips

Notera sambandet mellan vanlig matrismultiplikation och skalärprodukt: om \mathbf{x} är en radvektor ($1 \times N$) och \mathbf{A} är en matris ($N \times M$) så är $\mathbf{x}\mathbf{A}$ en radvektor ($1 \times M$) innehållande skalärprodukterna mellan \mathbf{x} och kolumnerna i \mathbf{A} .

3.2 Avkodning

Sista steget i DTMF-detektorn är avkodaren, där segmenten ska analyseras och tolkas till en sträng av knappsymboler ($0 - 9, *, \#, A - D$) med hjälp av DTMF-transformen. Avkodaren ska

göra en sammanvägning av energierna hos de olika frekvenserna för att kunna välja den symbol som är mest trolig för varje segment.

Uppgift 3

Skriv funktionen `dtmf_decode.m` som tar in en ljudvektor som innehåller en DTMF-puls och returnerar korrekt kod enl. tabell 1. Sekvenslängden N för beräkning av transformen bör motsvara kortaste tillåtna DTMF-pulslängd (se ovan) för optimal detektion.

```
function string=dtmf_decode(X,fs)
%
% dtmf_decode - decode DTMF sequence into string
%
% input:
%   X (Nx1 column vector) - Input signal
%   fs (real number)      - Sampling frequency
% output:
%   string (matlab string) - decoded keypad sequence
```

Att redovisa: m-fil

Display-tips

Om Q är absolutbeloppet av en DTMF-transform (radvektor, 8 värden) så ger

```
imagesc(Q(1:4) '*Q(5:8))
axis image
```

en intressant display. Hur kan den tolkas?

När du listat ut det kan du anropa funktionen

```
dtmf_plotkeys()
```

för att göra saken ännu tydligare!

4 Frivillig del

4.1 Automatisk segmentering

Nu när du har koden som kan koda av de enskilda pulserna är det dags att återvända till *segmenteringen* av signalen som du gjorde manuellt i första momentet genom att klippa ut pulserna. I följande uppgift ska du automatisera detta steg.

Segmentering av DTMF-signaler är ganska tacksamt tack vare att specifikationen kräver tystnad mellan varje tonpuls. Segmenteringen kan alltså göras helt baserat på signalens energi: varje gång signalenergin går från lågt till högt värde så börjar en ny puls, och när energin går ner igen så är pulsen slut (som ett mått på energi kan man ta signalen i kvadrat eller absolutbeloppet, det spelar ingen större roll här). För att kunna detektera pulser behöver vi ta ett *lokalt medelvärde* på energin, som sträcker sig över ett antal perioder av den lägsta DTMF-frekvensen. Detta lokala medelvärde får vi genom att lågpasfiltrera energisignalen, (kallas även för att "glätta ut" signalen). För utglättningen används exempelvis ett *rullande-medelvärde-filter* (moving average), vilket lättast görs med hjälp av matlabfunktionen `filter()`. För att välja

lämplig längd på medelvärdesfönstret kan du utnyttja vad vi vet från signalspecifikationen: alla giltiga DTMF-pulser är minst 40 ms långa och åtskiljda av minst 40 ms tystnad. Händelser kortare än denna tid är alltså ointressanta!

Pulserna kan sedan identifieras ur energikurvan med en enkel trösklings-operation, dvs när energin går över ett visst värde så har vi en puls, annars inte. För att detektionen inte ska vara beroende av insignalens absolutnivå, måste tröskelvärdet vara uttryckt relativt energikurvans toppvärde.

Uppgift 4

Du ska skriva en funktion `dtmf_calc_energy()` som beräknar en glättad energikurva lämplig att segmentera signalen med enligt ovanstående princip. Inparametrar är en ljudvektor och längden på glättningsfönstret som ska användas. Notera att ett medelvärdesfilter av längd N fördröjer signalen med $N/2$, något som blir uppenbart om in- och utsignal plottas tillsammans. Funktionen ska kompensera för denna fördröjning för att inte tappa "synk" i segmenteringen. Själva segmenteringen görs av den bifogade funktionen `dtmf_segment()`, som tar in energivektorn och returnerar en $k \times 2$ -matris där varje rad innehåller start- och sluttid för ett segment, där k är antalet detekterade segment i signalen.

Testa din funktion genom att beräkna energin för filen `dtmf_all.wav` som du läser in till en vektor med kommandot `[X,fs] = wavread('dtmf_all.wav')`. Välj lämpligt värde på fönsterlängden, och plotta energikurvan tillsammans med vågformen.

Anropa sedan `dtmf_segment()` och kontrollera att segmenteringen stämmer: det ska bli 16 segment, ca 100 ms/800 sampel långa - jämför några start- och sluttider (som alltså returneras i sampelnummer) med vad du kan avläsa i *WaveSurfer*.

Slutligen ska du sätta ihop denna kod med avkodaren för att få ett program som klarar av att ta in en fil med flera pulser och skriva ut/returnera avkodad knappsekvens i form av en Matlab-sträng (En sträng i matlab representeras som – du gissade det – en matris. Skriv `help strings` för att se hur det fungerar.) Se till att avkodningen (ev. tröskelvärden etc) blir oberoende av signalens absolutnivå.

Att redovisa: m-fil