Matlab 1 – vågformer, filter och talsyntes

1953 skapades Ove (Orator Verbis Electris) av Gunnar Fant, KTH.

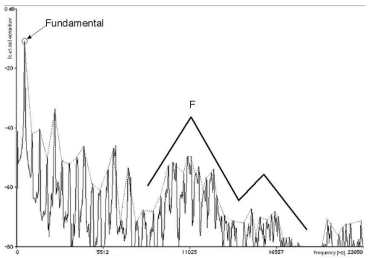
Ove var en talsyntesmaskin som kunde göra vokalljud. Ganska bra sådana dessutom, i alla fall med tanke på dåtidens teknik. Här finns en video där Jonas Beskow (också KTH) använder motion capture för att kontrollera Ove. I slutet av videon får ni dock en kort glimt av Gunnar Fant, och det interface han utvecklade i början av 50-talet:
https://youtu.be/xv4PlpUTf_c

Denna övning går ut på att framställa talljud med hjälp av Matlab, på liknande sätt som Ove.

Inför laborationen

Förberedelse – 1. Koll upp Ove och formantljud

För att skapa ett vokalljud krävs ett bredbandigt ljud, och sedan ett antal filter som formar detta ljud för att skapa formanterna, dvs de karaktäristiska övertoner som gör att olika språkljud skiljer sig åt.



Först behövs en fundamentalfrekvens (F_0). Det är själva grundtonen i rösten. För att härma en mansröst ligger F_0 runt 130 Hz, för att härma en kvinnoröst ligger F_0 runt 195 Hz. F_0 gör ni enklast genom att skapa en sinuston i rätt frekvens och längd som ni senare mixar in. Ni kan också testa att göra ett bandpassfilter med rätt brytfrekvens (F_0) och som ni senare applicerar på det bredbandiga ljudet.

Läs mer här:
https://www.soundonsound.com/techniques/formant-synthesis
http://person2.sol.lu.se/SidneyWood/praate/whatform.html
https://en.wikipedia.org/wiki/Formant

Förberedelse – 2. Skapa grundljudet

Ett sätt att sedan skapa ett bredbandigt ljud är med brus. Vitt brus (white Gaussian noise) i Matlab skapas med `y = wgn(m,n,p);`, där y är en m*n vektor med signalstyrkan p (som definieras i decibel relativt till en watt). Eftersom ett ljud inte får överstiga/understiga +/-1 när det spelas upp eller sparas ut från Matlab, måste ni kolla skalan så att inte distorsion uppstår. Tänk också på längden på vektorn, samt samplingsfrekvensen, och åt vilket håll ni bygger vektorn (allt blir enklare om ni jobbar med vektorer åt samma håll). Sätt samplingsfrekvensen till 44100 och ljudlängden till 3 sekunder.

Förberedelse – 3. Välj formant och kolla upp om bandpassfilter

Bestäm er för ett formantljud ni vill göra, och kolla upp om bandpassfilter och rätt brytfrekvenser för formantljudet.

Redogör för vilket vokalljud ni använt:_____

Formanter

För att forma formanterna behövs tre bandpassfilter (F_1, F_2, F_3). Dessa filter skapar övertonsserien som gör talljudet. I huvudsak är det de första två formantfrekvenserna som definierar vokalljudet, vilket syns på att de varierar markant mer mellan vokalljuden än vad F_3 gör. Formantfiltrerna filtrerar det ursprungliga ljudet parallellt och inte seriellt. Skulle filtren läggas parallellt skulle det inte finnas mycket information för F_3 om F_0 användes först, och tvärt om, eftersom de frekvenserna redan skulle vara dämpade av det/de tidigare filtren. Välj ett vokalljud (engelskt sådant) utifrån följande tabell (M = man, W = kvinna, Ch = barn):

Formant frequencies (Hz)		/i/ (ee)	/ɪ/ (i)	/ɛ/ (e)	/æ/ (ae)	/ɑ/ (ah)	/ɔ/ (aw)	/ʊ/ (ù)	/u/ (oo)	/ʌ/ (u)	/ɝ/ (er)
F_1	M	270	390	530	660	730	570	440	300	640	490
	W	310	430	610	860	850	590	470	370	760	500
	Ch	370	530	690	1010	1030	680	560	430	850	560
F_2	M	2290	1990	1840	1720	1090	840	1020	870	1190	1350
	W	2790	2480	2330	2050	1220	920	1160	950	1400	1640
	Ch	3200	2730	2610	2320	1370	1060	1410	1170	1590	1820
F_3	M	3010	2550	2480	2410	2440	2410	2240	2240	2390	1690
	W	3310	3070	2990	2850	2810	2710	2680	2670	2780	1960
	Ch	3730	3600	3570	3320	3170	3180	3310	3260	3360	2160
Formant amplitudes (dB)		−4	−3	−2	−1	−1	0	−1	−3	−1	−5
		−24	−23	−17	−12	−5	−7	−12	−19	−10	−15
		−28	−27	−24	−22	−28	−34	−34	−43	−27	−20

Source: Peterson and Barney (1952).

Ovanstående tabell kommer från: Peterson, G.E., and H.L. Barney, "Control Methods Used in a Study of the Vowels," Journal of the Acoustical Society of America, vol. 24, 1952.

Bandpassfilter skapas enkelt med matlabkod. Bandpassfiltren ska filtrera det bredbandiga ljudet parallellt, och de tre filtrerade ljuden ska därefter summeras/mixas samman i olika amplitudnivåer. Ett filter skapas med `[b,a] = butter(n,Wn,ftype);`, där b och a är koefficienterna för filtrets transferfunktion, n ställer ordningen på filtret, Wn är brytfrekvensen(erna) och ftype sätter filtertypen. Ni filtrerar därefter originalljudet genom `dataOut = filter(b,a,dataIn);`.

Lämpligt är 2a-ordningens bandpassfilter. Brytfrekvenserna sätts till korrekt Hz enligt ovanstående tabell, och bandbredden för filtren ska vara bredast för F_1, smalare för F_2 och smalast för F_3. Sätt bandbredden till 0.1 för F_1 (dvs lägre brytfrekvens för $F_1 = F_1-F_1*0.1$ och övre brytfrekvens för $F_1 = F_1+F_1*0.1$), 0.07 för F_2 och 0.05 för F_3. Liksom i laboration 1 ska brytfrekvensen `Wn` vara ett värde normaliserat mellan 0 och 1, där 1 motsvaras av halva samplingsfrekvensen (enligt Nyquistteoremet).

Vilken typ av filter användes? Dvs brytfrekvenser, ordning på filtret, osv? Varför? Testa gärna olika ordning på filtret och experimentera med brytfrekvenser för att försöka förbättra ljudet. Hur påverkas filtret och därmed ljud i och med olika ordning på filtret?

_____

_____

_____

Amplitud på formanterna och mix

Sedan ska de fyra ljuden mixas, summeras ihop till ett ljud. I tabellen ovan visas amplituden för F_1, F_2, respektive F_3 dämpat gentemot F_0. Värdet är angivet i dB och behöver konverteras från dB till magnitud. Detta fixas enkelt med `db2mag()` eller med `(10^(amplitudeSetting/20))`. Multiplicera varje formantljud med motsvarande formantnivå, och summera slutligen de fyra ljuden.

Hur låter det? Låter det naturligt? Vad är det/vilka egenskaper i det bredbandiga ljudet är det som ger detta intryck?

_____

_____

_____

Infade och utfade

Ljudet borde fadeas in och ut lite för att snygga till starten och slutet. Detta går enkelt att göra linjärt med en fadeVektor som går från 0 till 1 i ett visst antal steg, till exempel 100ms, och se till att ta hänsyn till samplingsfrekvensen. Exempelvis: `fadeVektor = [0:1/4410:1];` men, det är snyggare att göra detta så att det enkelt går att förändra fadetiden i ms. Denna vektor vänder vi sedan på för att göra utfade, se `fliplr`.

Skapa sedan en vektor med ettor i som är lika lång som längden av F_0 minus två fadeVektor. Använd ones(n), se till fadeVektorn och vektorn har samma proportioner, annars måste den ena transponeras med `'`. Därefter skapar ni en ny vektor där ni först lägger till fadeVektor, sedan vektorn med ettor och slutligen fadeVektor men som är vänd. Slutligen applicerar ni fadeVektorn på det färdiga talljudet med `.*`.

Spela upp ert vokalljud.

```
p = audioplayer(y, Fs);
playblocking(p);
```

Vilken tid valde ni för fade? Och, hur passade det?

_____

_____

_____

Förbättringar

Nu är det dags för att förbättra, eller i alla fall förändra, vokalljudet. Bruset skapar egentligen inte den bästa grunden för talljud. Kanske vore rosa brus, dvs brus som har jämn fördelning av energi i alla oktaver, även kallat 1/f-brus, ge ett bättre bredbandigt ljud. Men, i stället ska ni använda additiv syntesmetod bestående av fyra operatorer. Skapa därför en

frekvensmodulerad signal som byggs upp av en sinusoscillator i F_0, som frekvensmoduleras av en sinusoscillator i F_1, som moduleras av en sinus i F_2, som moduleras av en sinus i F_3. Liknande detta, fast med fyra operatorer:

```
outputSine = sin(2*pi*t.*freq+fmDepth*sin(2*pi*t.*freq));
```

Därefter ska formantfiltren appliceras som tidigare.

Testa att spela upp vokalljudet igen.

Hur låter det? Låter det naturligt? Hur skiljer sig detta från det första bredbandiga ljudet? Vad är det/vilka egenskaper i det bredbandiga ljudet är det som ger detta intryck?

_____

_____

_____

Ett annat sätt att få ett bättre talljud är genom summering av många sinus med olika frekvenser, en så kallad Band-Limited Pulse Generator. Detta görs genom att skapa ett antal sinustoner från grundfrekvensen, F_0, och uppåt (så långt som det behövs övertoner, dvs till och med F_3 och lagom långt uppåt i frekvenserna så att F_3s alla övertoner ryms) fördelade över frekvenserna. Detta kan ni räkna ut, tillräckligt noga genom att dela F_3s frekvens med F_0s frekvens, avrunda och lägga till 10. Detta ger 29 sinusljud för *a* enligt tabellen ovan (2440/130+10). Skapa den första övertonen genom att ta F_0s frekvens*(i+1), om i räknas från 1 och upp till antalet sinusljud.

Lämpligt används en for- (eller while-) loop till detta. Sinustonerna skapar ett mjukare och mer naturligt vokalljud jämfört mot det brusiga bruset och det metalliska FM-ljudet. Tänk på att skala ljudet i efterhand så att inte distorsion uppstår, och att alla toner summeras lika starka.

Därefter ska formantfiltren appliceras som tidigare.

Testa att spela upp vokalljudet igen.

Hur låter det? Låter det naturligt? Hur skiljer sig detta från det första bredbandiga ljudet? Vad är det/vilka egenskaper i det bredbandiga ljudet är det som ger detta intryck?

_____

_____

_____

Vidare utvecklingar

Ett sätt att göra vokalljudet mer levande och intressant är att skapa en enkel envelop som förändrar tonhöjden, antingen i en uppåtgående eller nedåtgående rörelse. Detta görs tämligen enkelt genom att skapa en vektor för frekvensförändringen som går från 1 (originalfrekvens) till det läge ni vill komma till, 1.05 för 5% uppåt eller 0.90 för 10% nedåt osv. Stegen i denna vektor måste dela förändringen i längden av tidsvektorn för F_0. Om t är tidsvektorn för F_0, startFrekvens är startförändring av F_0s grundfrekvens, slutFrekvens är slutförändring av F_0s grundfrekvens så:

```
frekvensVektor = [startFrekvens:skillnad mellan startFrekvens och
slutFrekvens/(length(t)-1):slutFrekvens];
```

Detta läggs sedan till F0 genom:

```
fundamentalSinus = sin(2*pi*fundamentalFrekvens.* frekvensVektor.*t);
```

Enklast är att enbart applicera detta på F_0 (om ni använder sinustoner), eftersom F_0 mixas in utan att filtreras.

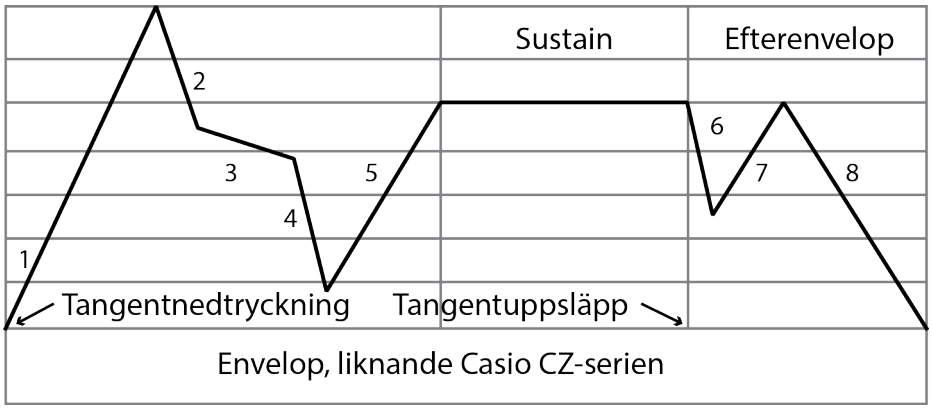
Vad innebär pitchenvelop det för slutljudet? Hur upplever man uppåtgående respektive nedåtgående enveloper?

_____

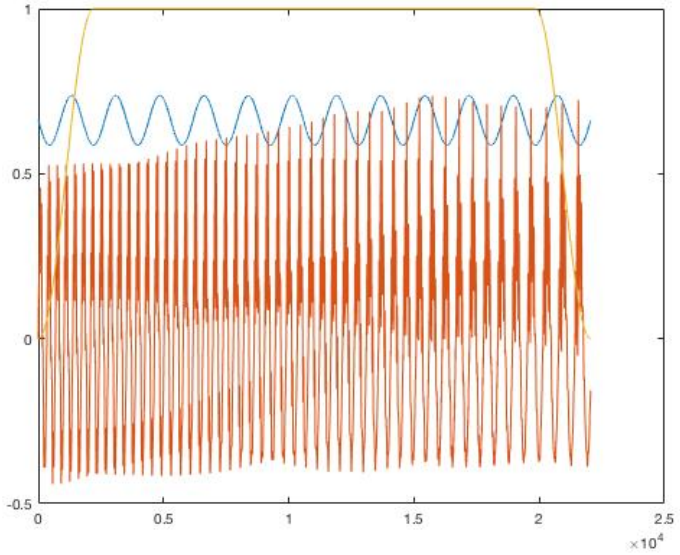
_____

_____

Vidare kan ni skapa en mer intressant amplitudenvelop än fade in och fade ut, som en ADSR (Attack, Decay, Sustain, och Release), och applicera den på ljudet. I analoga syntar användes oftast en ADSR (Attack, Decay, Sustain, och Release). Skapa en vektor som går från 0 till 1 under Attack tid, sedan från 1 till Sustainläget under Decay tid, och som avslutningsvis går från Sustainläget till 0 under Release tid. Totalt ska envelopvektorn vara lika lång som ljudvektorn. Envelopen appliceras sedan på ljudet med `.*`. Vill ni skapa en mer intressant envelop kan ni göra det liknande som gjordes i Yamaha DX7 och Casio CZ1000 där envelopen bestod av flera steg och varje steg bestod av en tidsinställning och ett värde. Det varje steg gjorde var att gå till värdet på den tid som ställdes in.



Ni kan också amplitudmodulera talljudet exempelvis med en sinusvåg, genom att skapa en sinuston som modulerar talljudet i ställbar frekvens och styrka. Kom ihåg att en sinus normalt svänger mellan -1 och +1, men amplituden nog bäst moduleras någonstans mellan 0.75 och 1 liknande den blå vågformen nedan (den röda vågformen är ett vokalljud och den gula är en enkel amplitudenvelop).



Ljudet lär säkerligen bli mer levande med sådana amplitudmodulationer, om än kanske inte nödvändigtvis mer röstlikt eller verkligt.

Hur och vilken AM och/eller amplitudenvelop testade ni? Vad innebar det för slutljudet?

_____

_____

_____

Finns tid kan ni prova att förändra ordningen/brantheten på bandpassfiltren och se vad som händer, om det blir bättre eller sämre. Prova gärna andra sorters filter än butterworth också.

Hann ni testa andra typer än Butterworth-filter? Hur skiljde sig filtren åt? Vilken typ av filter passade bäst?

_____

_____

_____

Att diskutera och reflektera kring

Vilken metod att skapa det bredbandiga ljudet var bäst? Och, varför?

_____

_____

_____

Hur skiljde sig dessa bredbandiga ljud åt? Och, vad innebar dessa skillnader för slutljudet?

_____

_____

_____

Hur skulle man kunna skapa konsonanter som S, T, K, eller L, M, N, …?

_____

Vidare läsning

Peterson, G.E., and H.L. Barney, "Control Methods Used in a Study of the Vowels," Journal of the Acoustical Society of America, vol. 24, 1952.
http://www.ling.upenn.edu/courses/Fall_2013/ling520/PetersonBarney52.pdf

Mer info om formanter:
https://en.wikipedia.org/wiki/Formant

Och ännu mer info om rösten:
http://newt.phys.unsw.edu.au/jw/voice.html

En kort text om modellering av talljud, Measuring and Modeling Speech Production:
http://www.haskins.yale.edu/featured/heads/mmsp/acoustic.html

En mer djupgående kurs om talsyntes:
http://www.phon.ucl.ac.uk/courses/spsci/iss/week5.php

Och slutligen kan det kanske roa er att lyssna på gamla inspelningar från historiska talsyntesmaskiner:
http://www.cs.indiana.edu/rhythmsp/ASA/partA.html