KTH Datavetenskap
och kommunikation

Spektrala transformer
Laboration: Vokalsyntes

Introduktion

I denna laboration är målsättningen att du ska få handgripliga erfartenterer av digital filtrering. Du ska implementera en enkel men användbar typ av återkopplat filter - tvåpolsresonator, och använda den för att simulera mänskligt tal.

En modell av vokalproduktion

Produktion av vokalljud i mänskligt tal kan grovt sett beskrivas så här: luft pressas ur lungorna mellan stämbanden, vilket ger upphov till ljudpulser, som fortplantas genom talröret (strupen, munhålan, näshålan, etc). Talröret formar ljudet olika beroende på form och läge hos tunga, käke, läppar mm, och producerar den slutliga ljudsignalen. Stämbanden kan därför ses som en *ljudkälla*, och talröret som ett *filter*. Denna populära beskrivningsmodell kallas *källa-filter-modellen* och den ska du tillämpa i denna laboration. Det filter som talröret utgör har en komplicerad överföringsfunktion, som innehåller ett antal resonansfrekvenser som kallas *formanter*. För vokaler kan filtret effektivt appoximras med ett antal tvåpolsresonatorer som parallellkopplas, där varje formant motsvaras av en resonator. Vanligtvis räcker det med att modellera de fyra första formanterna, vars centrumfrekvenser brukar betecknas F_1, F_2, F_3 och F_4, med tillhörande bandbredder B_1, B_2, B_3 och B_4

Utförande

Du ska utföra ett antal uppgifter med hjälp av Python, och svara på de frågor som ställs i peket. Laborationen utföres självständigt, antingen enskilt eller i grupp om två. Börja med att hämta `lab-reson.zip` från kurshemsidan. Denna fil innehåller några hjälpfunktioner och andra filer som du kommer att behöva.

I koden i peket betecknas alltid arrayer och matriser med versaler, medan skalärer betecknas med gemener. Vidare används genomgående konventionen att skriva all kod med `skrivmaskinsstil`. Den kod som efterfrågas (och ska redovisas) ska oftast vara i form av funktioner. Ofta beskrivs även hur du kan testa att funktionen gör vad den ska. Du gör klokt i att skriva dessa tester i en .py-fil (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.

Tvåpolsresonatorn

En tvåplolsresonator är en enkel typ av återkopplat filter som beskrivs av överföringsfunktionen

$$H(z) = \frac{b_0}{(1 - Re^{j\theta}z^{-1})(1 - Re^{-j\theta}z^{-1})}$$

som har två poler i det komplexa planet på avståndet R från origo, och med vinklarna $\pm\theta$. (Dessutom har den två nollställen i origo, men dessa påverkar inte frekvenssvaret.)

Tvåpolsresonatorn har ett frekvenssvar som består av en topp, vars läge i frekvensled bestäms av polvinkeln θ, och toppens "spetsighet" bestäms av polradien R (ju närmare R är 1, desto spetsigare topp).

Impulssvaret hos en tvåpolsresonator kan beskrivas som en dämpad svängning, där graden av dämpning bestäms av R - ju mindre R desto mera dämpning får man. Specialfallet när $R = 1$, dvs polerna ligger på enhetscirkeln, ger en helt odämpad svängning dvs en oscillator. $R > 1$ ger ett instabilt filter.

I denna laboration vill vi styra resonatorn i termer av resonansfrekvens f och bandbredd ψ. Polernas parametrar R och θ kan approximativt bestämmas ur frekvensen f och bandbredden ψ enligt följande[1]:

$$R \approx 1 - \psi/2$$

$$\theta \approx 2\pi f$$

där f och ψ är resonansfrekvens resp. bandbredd som fraktion av samplingsfrekvensen, dvs $f = 0.5$ motsvarar nykvistfrekvensen.

Förberedelseuppgift (för hand)

Skriv om uttrycket för överförinsfunktionen $H(z)$ ovan på formen

$$H(z) = \frac{b_0}{a_0 + a_1 z^{-1} + a_2 z^{-2}}$$

och bestäm koefficienterna a_k uttryckt i frekvens f och bandbredd ψ.

Studerar man $H(z)$ ser man att täljarens enda koefficient b_0 kommer att tjäna som någon sorts global förstärkningsfaktor. Detta är speciellt viktigt om resonansfrekvensen ska vara variabel, är det ofta önskvärt att filtrets totala förstärkning är oberoende av resonansfrekvensen. Vi kan låta b_0 ta hand om detta åt oss, genom att välja den så att filtrets förstärkning vid resonansfrekvensen alltid är ett. Detta fås då

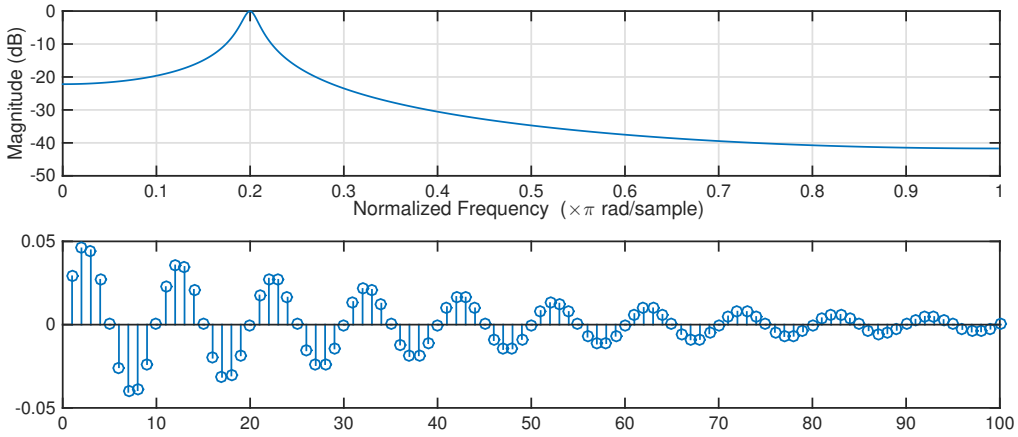
$$b_0 = (1 - R^2) \sin\theta$$

Verifiera gärna att det stämmer med egna uträkningar!

Innan du sätter igång med uppgifterna bör du ha koll på

1. Vad R och θ är samt hur de beror av f och ψ.
2. Vad formanter är och vad *källa-filter-modellen* är.
3. Vad som avses med parallellkopplade tvåpolsresonatorer, särskilt i kontext med ovanstående båda punkter.
4. Vad kopplingen är mellan normaliserad frekvens och samplingshastighet

[1] dessa approximationer gäller när $R \approx 1$, dvs för relativt skarpa resonanser, vilket kan antas i denna laboration



Figur 1. Frekvenssvar och impulssvar för tvåpolsresonatorn då $f = 0.1$ och $\psi = 0.05$.

Uppgift 1: statiska vokaler

I den första uppgiften ska du skapa ett statiskt tvåpolsfilter enligt ovanstående överföringsfunktion. Utgå från den bifogade mallen. Kolla på funktionen `scipy.signal.lfilter`. Som inparametrar anger man två vektorer B och A som innehåller koefficienterna $b_0, b_1, ...$ resp. $a_0, a_1, ...$, samt en vektor för indatasekvensen som ska filtreras. Kolla dokumentationen online!

Innan du ger dig på att göra vokalljud är det bra att verifiera att filtret gör vad det ska. Ett bra sätt att göra detta är att plotta frekvenssvar och impulssvar för filtret. Kolla t.ex. `scipy.signal.freqz`. Impulssvaret fås genom att mata in en impuls (se tipsruta nedan). I figur 1 kan du se hur dessa plottar bör se ut för $f = 0.1$ och $\psi = 0.05$. Pröva även med andra värden på f och ψ tills du förstår hur saker hänger ihop.

Med hjälp av ditt tvåpolsfilter ska du nu syntetisera några statiska vokaler. Använd två parallellkopplade filter för att simulera de två första formanterna F_1 och F_2. I tabell 1 finner du typvärden på formantfrekvenserna för de Svenska vokalerna. Som källfunktion (dvs "stämband") kan du använda ljudfilen `source.wav`. Samplingsfrekvensen är 16 000 Hz. Läs in filen med kommandot `fs,X = scipy.io.wavfile.read('source.wav')` och filtrera med frekvenser ur tabellen, för några olika vokaler som du väljer. Bandbredderna kan du sätta enligt följande: $B_1 = 50Hz$, $B_2 = 75Hz$, ($B_3 = 100Hz$, $B4 = 150Hz$). Lyssna, lägg ihop formanterna och spara som en fil, till exempel med `fs,X = scipy.io.wavfile.write('vocal.wav')`. Titta gärna på spektrum i exempelvis *WaveSurfer*[2]. *Notera att frekvenser och bandbredder naturligtvis måste räknas om till normerad frekvens, dvs delas med samplingsfrekvensen.*

För att vara klar med den här uppgiften bör du ha

1. En tydlig graf över frekvenssvar och impulssvar, antingen sida vid sida eller i samma subplot.

2. Ett par vokaler sparade i filer som du kan spela upp. Varje vokal bör bestå av två formanter.

[2]Laddas hem från `www.speech.kth.se/wavesurfer`

Tabell 1. Formantfrekvenser i Hz för Svenska vokaler för en mansröst (efter Fant). För en kvinnoröst ligger formanterna ca 20% högre.

Vokal	exempel	F_1	F_2	F_3	F_4
O:	rot	300	600	2350	3250
O	rott	350	700	2600	3200
Å:	rå	400	700	2450	3250
Å	rått	500	850	2550	3250
A:	bar	600	950	2550	3300
A	barr	750	1250	2500	3350
I:	bil	250	2200	3150	3750
I	Bill	350	2150	2750	3500
E:	deg	350	2250	2850	3550
E/Ä	vägg	500	1900	2550	3350
Ä3	herr	650	1700	2500	3450
Y:	ny	250	2050	2700	3300
Y	nytt	300	2000	2400	3250
Ö:	föd	400	1750	2300	3350
Ö	född	550	1550	2450	3300
Ö3	för	550	1150	2450	3250
U:	duk	300	1650	2250	2250
U	puck	450	1050	2300	3300

Python-tips
En insignal med en impuls kan man göra med t.ex.
`X = np.append([1], np.zeros(n-1))`
där `n` är önskad sekvenslängd. För plottningen kan det vara bra att även skapa en vektor med tidsvärden. Till sånt är `np.linspace` din bästa vän.
Plotta t.ex. med `plt.stem` eller `plt.plot`

Uppgift 2: en tidsvariabel resonator

I talsignalen (och i alla andra någorlunda intressanta ljud) så varierar frekvesninehållet kontinuerligt över tiden. Detta innebär att om man ska modellera tal med ett filter, måste filterparametrarna kunna variera. Problemet i detta fall är att matlabs `filter` inte tillåter variabla filterkoefficienter, så vill man göra detta måste man antingen filtrera små bitar (typ 10 millisekunder) åt gången, eller så får man göra sin egen filterfunktion. Det senare är mer lärorikt och inte alls så svårt som det kanske låter.

Skriv en funktion som implementerar en tidsvariabel tvåpolsresonator - ett formantfilter. Funktionen ska följa formatet för funktionen formantfilter i den bifogade mallen.

Funktionen ska alltså ta en insekvens $x(n)$ och filtrera den enligt filterekvationen

$$y(n) = b_0 x(n) - a_1 y(n-1) - a_2 y(n-2)$$

där b_0 och a_n är desamma som de du fick fram i förberedelseuppgiften. Den viktiga skillnaden nu är att de ändras hela tiden, eftersom de beror av F som är en vektor av samma längd som X. Bandbredden kan vara konstant i denna laboration.

tips: denna uppgift löses bäst med en for-slinga (se `help for`*) - något som man annars ofta kan klara sig utan med numpy om man använder det på bra sätt.*

Om du vill testa funktionen kan du filtrera en brussignal med ett filtersvep – låt f svepa från 0 till 0.5 i 8000 steg (tips linspace!). En brussignal av samma storlek som skapar du enkelt med `N=np.random.rand(size(8000));`. Sätt bandbredden till 0.05. Lyssna på ljudet - det ska låta som någon sorts "swooosch"... Ett spektrogram över ljudet bör visa ett tydligt diagonalt stigande band.

För att vara klar med den här uppgiften bör du ha

1. En funktion som implementerar en tidsvariabel tvåpolsresonator.
2. Ett par vokaler sparade i filer som du kan spela upp.

Uppgift 3: sammansättning och icke-statiska vokaler

I naturligt tal så ändras talröret - och därmed formanterna - kontinuerligt, dvs de glider mellan olika vokaler. Frekvenserna i tabell 1 är en sorts riktvärden för normaltal, men i praktiken handlar det om ett kontinuum (ofta refereat till som *vokalrymden*).

Nu ska du syntetisera icke-statiska vokaler, sk diftonger med hjälp av den nya resonatorn. Resonatorerna parallellkopplas, precis som i första uppgiften, men använd här fyra istället för två. Det ger lite mer definierad diskant (ljudet kan bli lite för vasst - i så fall kan man minska den relativa nivån på resonator 3 och 4 genom att helt enkelt applicera en skalfaktor på resonatorns utsignal. 0.7 för formant 3 och 0.3 för formant 4 kan vara bra, prova dig gärna fram)

Skriv ett program som genererar diftonger, dvs glidningar från en vokal till en annan. Man ska kunna ange startvokal och slutvokal, samt längden för övergången. För att underlätta, så finns en tabell med vokalerna i de medföljande filerna i form av en variabel för varje vokal (titta i filen så förstår du).

Använd linspace för att skapa styrvektorerna till formantfiltren. Bandbredden kan du sätta till samma som i uppgift 1 (den kan hållas konstant). Glöm inte att frekvenser & bandbredd måste normeras (delas med fs) innan de går in till filtret!

Du behöver även en röstkälla - använd den bifogade funktionen `simplesource()` - det är en enkel implementation av en tidsvariabel röstkälla där varje röstpuls modelleras som en "spik". Denna källa måste också ha en styrparameter, $F0$ dvs grundtonsfrekvensen. Denna kan modelleras enligt samma princip som formatfrekvenserna, dvs ange önskad start och slutfrekvens. F_0 för en mansröst varierar oftast inom intervallet $85 - 180Hz$, och för en kvinnoröst mellan $165 - 255Hz$.

För att vara klar med den här uppgiften bör du ha

1. Ett program som lever upp till kraven i uppgiften.
2. Ett par diftonger sparade i filer som du kan spela upp.

Uppgift 4 (frivillig): kvinnor, barn och längre sekvenser

Utveckla diftong-synten så att du kan syntetisera vokalsekvenser av valfri längd, genom att foga samman diftonger. OBS, gör sammanfogningen direkt på styrvektorerna (och inte på ljudsignalen) för att undvika diskontinuiteter. Utnyttja definitionerna i vokalmatrisen - det kan vara

smidigt att beskriva vokalsekvenser på formen `[vA vI vA vO]` - detta ger en matris med *målvärden* för vokalsekvensen där varje kolumn motsvarar en tidpunkt och varje rad en formant. Motsvarande sekvenser kan göras för grundtonen F_0 och för durationen. Gör sedan en loop som sätter ihop kolumnerna parvis till diftonger, som i sin tur läggs efter varandra till sekvenser.

Försök få din vokalsyntesmaskin att säga olika saker (fundera ut ord och meningar som bara innehåller vokaler - det finns fler än man tror) och laborera med olika $F0$-konturer - stigande, fallande, topp i mitten, etc. Kan du få det att låta som en fråga?

Försök även att få rösten att låta som en kvinna eller ett barn genom att applicera skalfaktorer på formantfrekvenserna. Riktvärden att testa kan vara +20% för kvinnoröst, +50% för barnröst. Du kan skala om grundtonskurvan (F_0) med större skalfaktorer. Vad händer om du sänker formanterna?

För att vara klar med den här uppgiften bör du ha

1. Vokalsekvenser som skulle kunna tillhöra människor av olika ålder, kön och storlek.
2. Vokalsekvenser av varierande längd.