Exercise #4:
Statistical parametric phone synthesis GMMs and LP

Exercise summary:

The fourth exercise focuses on synthesizing speech sounds using a statistical parametric model of speech (Fig. 1). A Gaussian Mixture Model (GMM) from Exercise #3 will be used to model distributional properties of different speech sounds in terms of MFCC features. Given a specification of sounds-to-be-synthesized, the GMM model is then sampled for feature vectors corresponding to the sounds. Then, the sampled MFCC vectors are converted into linear prediction (LP) coefficients (from Exercise #2). Finally, the LP coefficients are paired with an excitation signal (an impulse train or a glottis signal) and synthesized into audible speech.

More information and guidance for each step is provided in Appendices A–D.

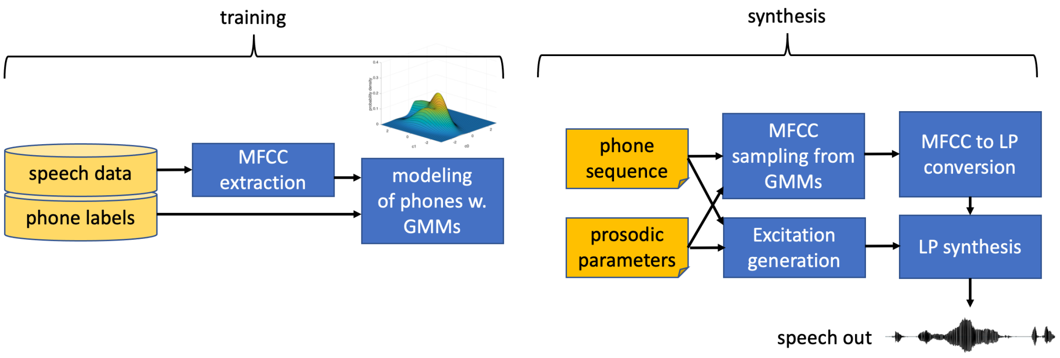

Figure 1: An overall schematic view of the synthesis pipeline implemented in this exercise. The training part is similar to MFCC and GMM -based phone classification in Exercise #3.

Please note that the exercise consists of questions related to different sub-tasks. However, given the nature of the synthesis problem, you may want to implement all steps before providing your final answers to each of the questions.

Software: MATLAB.

Data: Utterances and phone annotations of LibriSpeech corpus clean-dev section. Please download the data from: *https://www.openslr.org/resources/12/dev-clean.tar.gz* (audio) and https://zenodo.org/record/2619474#.YeULdCzkuL4 (annotations in .TextGrid format). Contact course assistants if you have trouble accessing the dataset.

Tasks:

0.1 Extract MFCC features for LibriSpeech clean-dev set (pre-provided)
0.2 Train a GMM model for each phone in the dataset (pre-provided)
0.3. Define parameters for the speech to synthesize (sounds, durations, F0, voicing)
1. Implement LP excitation signal generation according to speech parameters
2. Implement LP vocal tract filter parameter generation according to speech parameters
3. Implement synthesis of the speech sounds using LP.
4. Experiment with the speech synthesis pipeline.

Deliverables: `E4_main.m`, `computeExcitation.m`, `computeSynthesisOutput.m` and `computeTractParams.m`, five synthesized speech waveforms (`custom_synthesis.wav`, `test_synthesis_impulse.wav`, `test_synthesis_glottal.wav`, `excitation_impulse.wav`, `excitation_glottal.wav`), a written report (*`E4_surname_report`*`.pdf`).

Learning goals: statistical sampling, generative modeling, speech parametrization, speech synthesis

Note: All exercises of the course, including this one, will consist of a written report and additional files (e.g., code, sounds, annotation files) that are to be created and submitted for evaluation. In the report, always add your name and student ID to the beginning of the report. Use complete English sentences or paragraphs of text to answer the exercise questions, and number each response according to the question numbers in the exercise instructions. Some of the questions have strictly correct answers while others can have multiple valid responses.

Note 2: The exercise will be semi-automatically scored based on the variables and functions defined in the MATLAB code template. Do not change the naming of the key variables or specifications of function templates and calls defined. If you fail to complete the full assignment, in order to get partial points of correct parts, make sure your code runs up to the point that you managed to complete.

Task 0.1: Pre-process data for synthesis experiments

Task 0.1: Load LibriSpeech dataset and extract MFCC features and their phone labels for all speech frames using the pre-provided code in `E4_main.m`.

Note that, unlike the phone classification experiments of Exercise #3 (E3), now MFCC features are extracted *without* mean and variance normalization, as we want to model the features as they appear in speech without losing information to allow accurate feature generation.

Guidance: All code is pre-provided. Please follow the comments for explanation.

Task 0.2: Train a GMM model for all phone categories in the dataset.

Task 0.2: Use the pre-provided scripts to train a generative GMM model for each of the phone classes in the dataset.

This pre-provided processing step uses LibriSpeech dev-clean section to train a separate GMM model for each of the phone types in the dataset similarly to E3. The model is also tested on phone classification to ensure that the training has succeeded, with the exception that all speech data is now used for model training and testing (no separate test split) in contrast to E3.

Guidance: All code is pre-provided and follows the example solution to Exercise #3. Please follow the comments in E4_main.m. You can change the parameters of the GMM if you want (e.g., for computational speed or for experimenting at the end once you have a functioning synthesis pipeline), but please make sure your phone classification accuracy stays above chance level.
If GMM training takes a lot of time on your computer, consider saving/loading the trained model for later use.

Task 0.3: Define an utterance to-be-synthesized

Task 0.3: Use the pre-provided utterance specification template for sentence "*She is my friend*" as a starting point for later steps. Familiarize yourself with the basic specifications, so that you understand how the provided parameters are utilized in the next steps.

Guidance: In our simple synthesis setup, we will define four properties for the signal: a sequence of phones to synthesize, duration of each phone, whether each phone is synthesized as voiced or voiceless, and fundamental frequency (F0) of each (voiced) phone.

Phones are defined in terms of their ARPABET codes used for LibriSpeech annotation, listed in cell-array `phones_to_synthesize`. For reference, cell array `unique_phones` contains the list of all available phones for LibriSpeech (see Appendix D for a mapping table between IPA and ARPABET). A binary voicing flag for each phone is given as a vector entry in `voicing`. Phone durations are provided in vector `phone_durations` (in milliseconds). F0 values are provided for each sound in vector `f0` (in Hz). Dimensionality of each vector should correspond to the number of phones.

Example specification for an utterance "*She is my friend*" (also used in Task 4) is given as an example to get started with the following steps. In addition, an example of a simple four-vowel sequence with its parameter specifications is provided in the code (commented away).

- In E4_main.m, p3., the text should read "For white noise, you can use randn()to produce white noise with samples randomly drawn from a Gaussian distribution of zero mean and unit variance.", not about random uniform numbers in[-1, 1].

Task 1: Implement excitation signal generation

In order to synthesize speech with LP, an excitation signal is needed. Excitation for voiced sounds can be modeled as a periodic pulse train or using a real glottal source signal. Excitation for voiceless sounds can be modeled using white noise.

In the absence of real-time processing requirements, the easiest approach is to first create the excitation signal for the entire utterance before modeling the vocal tract. In this exercise, we will create excitation in a piece-wise manner by defining the excitation of each phone separately, followed by concatenation of the sound-specific excitation signals.

Task 1.1: Implement 16 kHz excitation signal generation for i) silent segments, ii) voiceless segments (phones), and iii) voiced segments (phones) using impulse train excitation.

Task 1.2: Implement voiced excitation generation using the pre-provided glottis_long.wav by adjusting its duration and F0 (by resampling) to match with the segment specifications. This is used when `excitation_style = 'glottis'` condition it satisfied.

Task 1.3: Use the default utterance from Step 0.3 to produce excitations with impulses and with glottal waveform and save them as `excitation_impulse.wav` and `excitation_glottal.wav` (16 kHz mono).

The final excitation waveform should have the same total duration as is the sum of your sound durations in synthesis specifications,

Guidance:

See Appendix A for a conceptual basis.

Duration of your excitation signal should match your specified phone durations, so `sum(phone_durations)/1000==length(excitation)/fs` should return `True`. Impulse train or glottal signal of each voiced sound should have the F0 value as specified in Task 0.3. Use white noise during unvoiced sounds, and silence (or inaudible noise levels) for 'sil' segments.

An *impulse train* is a signal that consists of zeros and ones (impulses), where the ones are spaced every *T* samples, where *T* is the period of the desired excitation F0 (in samples).

For glottal excitation, the original F0 of the recorded signal is approximately 114 Hz. You can adjust F0 of the glottal excitation simply by resampling the signal by suitable up/downsampling factors and using `resample()` function.

For white noise, you can use `randn()` to produce white noise with samples randomly drawn from a Gaussian distribution of zero mean and unit variance.

For unvoiced excitation, amplitude of the noise probably requires some manual tuning in order to balance it with the voiced segments.

Question 1.1: Can you identify vowel identities already based on the excitation signal and F0 alone? not really spectral content ağız shape falan

Question 1.2: Why the current type of piece-wise (phone-wise) excitation signal generation can be suboptimal for synthesis of continuous speech? not discontinuitiy

Question 1.3: Is a strict division into mutually exclusive voiced and unvoiced excitation signals an ideal solution? Can the excitation be a mixture of the two types in some cases?

fricatives and affricates? girerken çıkarken vocal fold azcık açık
böyle transition olan boundarylerde bu durum olabilir, triphone,diphone

Task 2: Implement vocal tract filter parameter generation

According to the source-filter model of speech, we will need filter parameters for the vocal tract (=LP coefficients) in addition to the generated excitation signal. In Task 0, each phone type was modeled by a generative GMM using MFCC features. In this step, you will sample these models in order to generate sound-specific vocal tract filters as a function of time.

We will create tract filter parameters with the same 16 kHz sampling rate as we have for the excitation.

Task 2.1: Implement a sampling procedure for obtaining a phone-specific MFCC vector **x** from the trained GMMs, given a pre-specified sound identity in `phones_to_synthesize`. There are many alternative ways for this. For instance, you can just use the mean vector of one random component or mean of the component means. Alternatively, you can generate a unique MFCC vector by probabilistic sampling from the distribution specified by the GMM. This will produce a different variant of the same sound every time you run the code. burayı doğru mu anladın

Task 2.2: Create vocal tract parameters for the example utterance defined in Step 0.3, and plot them using `plotTractParameters()`. Add the figure to your report.

Once you have an MFCC vector **x** for the current sound, it is mapped to LP coefficients **a** = *f*(**x**) using `mfcc2lpc()` function (pre-provided; essentially inverts the MFCC vector back to a time-domain waveform and then estimates the LP from the waveform). The created LP parameters are then repeated for the duration of the sound so that the sound-specific filters match timing of the excitation (code also pre-provided).

The loop repeats the whole generation process for each sound in your synthesis specifications. As a result, you should have a matrix of filter parameters, where the number of rows corresponds to the number of samples in your excitation, and number of columns should be `lp_order+1` (because of the included trivial 0th LP coefficient).

Guidance:

See Appendices B and C for conceptual basics.

Function `mvnrnd(mean,sigma)` allows you to draw random samples from multivariate Gaussian distributions. Remember that your Gaussian mean vectors for each GMM component are stored in `GMM.means{phone_index}(GMM_component,:)`, and covariance matrices sigma are stored in `GMM.sigmas{phone_index}(:,:,GMM_component)`.

the same MFCC vector will be assigned to each instance of the same phone to be synthesized.

allowing for greater variability

Question 2.1: What are the potential advantages and disadvantages of stochastically sampling vocal tract values from the GMM instead of using the fixed mean values of the model?

Question 2.2: The current approach to vocal tract modeling does not take coarticulation into account. Consider and describe how coarticulation could be modelled in the synthesis system.

Hidden Markov Models (HMMs), neural networks **biphpne triphone bence yine ağız yapısı,**

mesela duration hesaba katılmamış phone middleda nerde vs

Task 3: Implement synthesis stage

In the final generation step, we will combine the excitation signal and the vocal tract parameters (LP coefficients) to synthesize the desired utterance.

Task 3.1: Implement a sample-by-sample filtering loop where the excitation signal is filtered by the corresponding vocal tract parameters (LP coefficients).

Guidance:

Recall from Exercise #2 that LP synthesis filter is an IIR filter (the model is *autoregressive*). Output y[*n*] of an LP filter with prediction coefficients *a*[*k*] is therefore obtained with:

$$y[n] = x[n] - \sum_{k=1}^{p} a[k] \times y[n-k] \qquad (4.1)$$

where *x*[*n*] is the excitation signal.

Note that `mfcc2lpc()` returns the analysis filter coefficients a[k], but note that the vector contains the trivial coefficient $a_0 = 1$ which in MATLAB indexing corresponds to `a[1]`. Make sure your indexing is correct when implementing Eq. 4.1.

Task 4: Experiment with the synthesis pipeline

In the final step, experiment with your synthesis pipeline with different sound sequences, and perhaps with different details of your implementation. Once you are satisfied with your system, create two .wav files of synthesized signals:

Task 4.1: Synthesize the utterance "*She is my friend*" according to following specifications (same as in Step 0.3):

```
phones_to_synthesize = {'SH','IY','sil','IH','S','sil','M','AY','sil','F','R','EY','N','D'};
voiced =               [0    1    0    1    0   0    1   1    0    0   1   1    1   0];
phone_durations =      [200  150  80   100  80  30   40  220  50   50  120 200  80  160];
f0 =                   [100  130  100  100  100 100  100 130  100  100 100 120  120 100];
```

using glottal excitation and save it as `test_synthesis_glottal.wav` (16 kHz mono). Also create the same sound with impulse excitation and save it as `test_synthesis_impulse.wav`.

Task 4.2: Synthesize a custom utterance created by yourself (minimum duration 1 second) and name it as `custom_synthesis.wav` (16 kHz mono). The utterance can be a sequence of arbitrary sounds or an attempt to produce a comprehensible English utterance. Write down the specifications and text transcript of your utterance in your written report.

Guidance: You can use Appendix C or, e.g., https://isip.piconepress.com/projects/switchboard/doc/education/phone_comparisons/ for interpreting LibriSpeech phone codes in terms of English phones/phonemes. You can also use provided `babbleGenerator()` function to create random sequences of vowels to test and tweak your system.

Question 4.1: Describe the sound quality of the produced speech. Are the sounds understandable? Does the quality depend on sound in question? duyabildiğin phoneların ortak özelliği

Question 4.2: How would you compare the impulse train and glottal excitation in terms of speech naturalness? glottal more natural and human-like

Question 4.3: Besides accounting for coarticulation, how would you improve the system in order to produce more natural speech? dnn, başka ne var tek bir speaker biphone falan duration ağız yapısı

Question 4.4: How the hyperparameters (e.g., GMM components, LP order) in the current system impact the produced speech quality? Would there be a way to automatically optimize them for best sound quality? evaluate performance of the system on a validation set by trying diff hyperparameters

---

Returning the exercise

Return a written report (*`E4_surname_report`*`.pdf`) with a title page including your name and student number + content pages with written answers to Questions 1.1–4.4. Combine the report with the created synthesized waveforms (`custom_synthesis.wav`, `test_synthesis_impulse.wav`, `test_synthesis_glottal.wav`, `excitation_impulse.wav`, `excitation_glottal.wav`) and your `E4_main.m` `mcomputeExcitation.m`, `computeSynthesisOutput.m` and `computeTractParams.m`, scripts into a .zip file `E4_firstname_surname_studentID.zip` and upload it to Moodle.

Appendix A: Conceptual background, excitation generation

In normal speech, excitation of voiced speech is generated by sustained lung pressure, which, due to Bernoulli effect, causes vibration of the vocal folds in the larynx. This results in a periodic signal consisting of pulse-like "puffs of air", technically characterized by its *glottal volume velocity waveform*. Inverse of the period *T* of this pulsation is known as the fundamental frequency (F0) of speech. This excitation signal transmits energy to the vocal tract, which then filters the excitation according to its geometric and physiological properties.

In speech processing, the simplest approximation for a glottal excitation is simply an impulse train. An impulse train consists of zeros and ones, where the ones are spaced at every *T* in time in order to obtain the desired F0 for the signal.

For more natural sounding speech, glottal excitation could also be modeled as *pulses* instead of impulses, especially in terms of temporally *asymmetric pulses* similar to real glottal volume velocity waveforms (see, e.g., Liljencrantz-Fant model of excitation generation: https://ccrma.stanford.edu/~jos/SMAC03S/Liljencrantz_Fant_Derivative_Glottal_Wave.html or http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.643.5278&rep=rep1&type=pdf). Besides added naturalness, the use of such models provides control in terms of F0 but also *phonation style*.

In case of linear prediction, the excitation signal can also be a residual signal obtained by filtering the original speech with an LP-based vocal tract filter estimated from the same signal window. However, this only applies to analysis-synthesis cases where the original signal is known, and is not applicable to, e.g., speech synthesis where the signal is generated from text. In case of speech coding, the residual is not actually transmitted and used in detail, but its key characteristics are quantified, quantized, and transmitted to the synthesis side.

A fourth option is to use actual pre-recorded glottal excitation signals as the waveform. These signals are obtained from the process of *glottal inverse filtering* (GIF), which is a technique for removing (or separating) the effects of vocal tract and lip radiation from a recorded speech signal. This is then an approximate reconstruction on the actual phonation style of a real speaker. Since the excitation in the recording has a natural F0 value specific to that speaker and recording situation, adjustment of the F0 can be achieved by up-/down-sampling the excitation signal. Since the original GIF-based excitation signal has a finite length (as it is difficult to sustain phonation at a fixed F0 for long periods of time), length of the excitation signal can be extended by concatenating additional periods of the excitation to the end of the actual excitation. However, a naïve approach to concatenation will make the excitation sound unnatural for the extended sections, as the concatenation will introduce artificial exact repetitions and potential discontinuities/jitter to the periods that human hearing will pick up. In this exercise, the glottal waveform has been artificially extended in time to support much higher F0s for voiced segments than what the original one would enable.

Appendix B: Conceptual background, generative modeling

Statistical parametric speech synthesis, as its name implies, is based on statistical models of speech parameters. Unlike concatenative synthesis, where segments of real pre-recorded speech are combined to form the desired utterances, statistical synthesis relies on *generative machine learning models* that are capable of producing speech features corresponding to different speech sounds. Formally, a speech feature vector **x** is sampled from a model M_S of a speech sound *S* with pre-learned parameters $\boldsymbol{\theta}$ as:

$$\mathbf{x} \sim M_S(\boldsymbol{\theta}) \qquad (0.1)$$

This is then repeated for all time-steps in the signal-to-be-synthesized using the appropriate *S* for each time-step, and the resulting sequence of **x**[*n*], *n* = 1, 2, …, *N*, is finally converted into audible speech using digital signal processing means.

In a simple case, **x** could be magnitude spectrum vectors (from Fast Fourier Transform) and $M_S(\boldsymbol{\theta})$ could be a multivariate normal distribution fitted to a large number of **x** corresponding to sound *S***,** as extracted from annotated training data (cf. Gaussian phone model in Exercise #3). Given a sequence of **x** sampled from the model, the corresponding speech signal could be then obtained by simply performing Inverse Fast Fourier Transform (IFFT) on the sequence of **x**, converting spectra to time-domain, and somehow ensuring waveform continuity across the neighboring feature frames.

In classical statistical synthesis systems, the utilized generative models are Gaussian Mixture Model-Hidden-Markov Models (GMM-HMMs). In a GMM-HMM, GMMs are used to model distributions of acoustic features during specific sounds while the HMM is responsible for modeling the transitions from sounds to others, each HMM state corresponding to one sound or part of a sound (e.g., onset, middle, or offset part).

Since implementation of a complete GMM-HMM is beyond the scope of an introductory course to speech processing, we will focus purely on the generation of speech from a sequence of speech features sampled from a GMM.

Recall from Exercise #3 that the probability density function (pdf) of a multivariate Gaussian is defined as

$$p(x|\theta) = p(x|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{\exp\left(-0.5(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^2|\boldsymbol{\Sigma}|}} \quad . \quad (0.2)$$

In addition, the pdf of a mixture of *K* Gaussians (GMM) with weights w_k was defined as:

$$p(\mathbf{x}|\theta) = \sum_{k=1}^{K} w_k p(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \sum_{k=1}^{K} w_k \frac{\exp\left(-0.5(\mathbf{x}-\boldsymbol{\mu}_k)^{\mathrm{T}}\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right)}{\sqrt{(2\pi)^2|\boldsymbol{\Sigma}_k|}} \qquad (0.3)$$

Following the EM-algorithm in Exercise #3, the parameters of the GMM model can be estimated for each of the sounds (here: phones) we want to model. In the phone recognition case, we were interested in the relative likelihoods of the feature vectors across models of different phone GMMs, as we wanted to determine which GMM (which sound *S*) explains the observed feature values the best. Now, in the case of synthesis, we know the sound we want to produce, but we need to infer a likely **x**, given the GMM ("*how would a likely feature vector **x** look like for this chosen sound?*"). The way to obtain an **x** compatible with the GMM model is to sample from the model:

$$\mathbf{x} \sim \mathrm{GMM}_S(\boldsymbol{\mu}, \mathbf{w}, \boldsymbol{\Sigma}) \qquad (0.4)$$

This is done in two steps:

1. Given the GMM component weights w_k, $\sum_k w_k = 1$, for the sound of interest, sample a Gaussian component *k* proportional to the relative weights, i.e., by sampling from a discrete cumulative density function (cdf) of **w**.

$$k \sim cdf(\mathbf{w}) \qquad (0.5)$$

2. Given the sampled component *k*, sample from the corresponding multivariate Gaussian distribution with parameters $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$. While cdf of a multivariate Gaussian distribution does not have a closed analytic form, there are numerical and other means (see https://en.wikipedia.org/wiki/Multivariate_normal_distribution#Cumulative_distribution_function) to sample from the distribution, and standard mathematical libraries should have routines for this:

$$\mathbf{x} \sim N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \qquad (0.4)$$

In other words, the process goes as follows: 1) train a GMM for each phone of interest using labeled training data, 2) during synthesis time, sample from the phone-specific GMMs to obtain feature vectors for the desired sounds.

Appendix C: Conceptual background, features for synthesis

In Exercise #2, we saw that the use of linear prediction (LP) provides a compact way to represent vocal tract characteristics of speech with a low-dimensional feature vector **a** (the linear prediction coefficients). In addition, speech can be easily synthesized from the LP coefficients if a suitable excitation signal is available or can be created. Since the excitation is modeled separately from the vocal tract in LP, this allows independent modification of source and articulatory characteristics of speech, such as adjusting the fundamental frequency (F0) of speech while maintaining the linguistic content unaltered. All this makes LP an attractive signal representation for statistical synthesis.

However, a problem with standard LP coefficients **a** is that they are not “nicely” distributed across several tokens of the same sound (or even for different frames extracted from the same token), but the distributions tend to be non-normal and skewed, which makes them unideal for parametric modeling with Gaussian models (i.e., requiring several Gaussians even for a unimodal distribution). Perhaps more importantly, the relationship between magnitude of changes in the prediction coefficients and the corresponding changes in the audible speech is not consistent. For some filter configurations, a small change in **a** can lead to large audible changes or even to an instable synthesis filter, while for others, large changes in **a** can have only a small impact on the resulting sound. Since parametric probabilistic modeling of the features is deemed to have inaccuracies, ideal synthesis features would be more consistent in the face of distortions and errors in the modeling.

While there are ways to transform the canonical LP coefficients into a more robust form, statistical synthesis often takes another route: ***1) modeling speech with MFCCs at training time***, as in done in speech recognition, ***2) sampling MFCCs from the parametric model during the synthesis time***, and then 3) ***converting the sampled MFCC vectors to LP filter coefficients*** for the final waveform synthesis. This is possible because all the steps in MFCC calculation are invertible: time-domain speech frame **x** can be recovered from MFCC vector **y** by simply following the MFCC calculation steps backwards using the inverse cosine transform (IDCT), exponentiation (to return to linear domain from log-domain), filtering with a matrix inverse of Mel-filterbank matrix, and finally applying inverse Fast Fourier Transform (IFFT) to the obtained spectrum. Given that MFCCs are good at capturing phonemic information from speech in a compact and robust manner, they also make acceptable features for basic statistical speech synthesis.

Note that MFCC inversion process is not *lossless*. In MFCC calculation from a speech frame, signal phase information is lost when calculating magnitude spectrum in Fourier domain. In addition, spectral fine details (including F0-related excitation structure) are lost during Mel-filtering. While recovery of this information is not possible, MFCCs still encode the general resonance characteristics of the vocal tract, which is also what LP analysis models.

Appendix D: IPA to ARPABET conversion table

		IPA Symbol	ARPAbet (SV)	ARPAbet (UV)	Examples
Vowels	Front	i	i	IY	beat
		I	I	IH	bit
		e	e	EY	bait
		ɛ	E	EH	bet
		æ	@	AE	bat
	Back	ɑ	a	AA	Bob
		ɔ	c	AO	bought
		o	o	OW	boat
		U	U	UH	book
		u	u	UW	boot
	Mid	ɝ	R	ER	bird
		ə	x	AX	ago
		ʌ	A	AH	but
Diphthongs		ɑI	Y	AY	buy
		ɑU	W	AW	down
		ɔI	O	OY	boy
		ɨ	X	IX	roses
Stop Consonants	Voiced	b	b	B	bat
		d	d	D	deep
		g	g	G	go
	Unvoiced	p	p	P	pea
		t	t	T	tea
		k	k	K	kick
Fricatives	Voiced	v	v	V	vice
		ð	D	DH	then
		z	z	Z	zebra
		ʒ	Z	ZH	measure
	Unvoiced	f	f	F	five
		θ	T	TH	thing
		s	s	S	so
		ʃ	S	SH	show
Semivowels	Liquids	l	l	L	love
		l	L	EL	cattle
		r	r	R	race
	Glides	w	w	W	want
		ʍ	H	WH	when
		j	y	Y	yard
Nasal	Non vocalic	m	m	M	mom
		n	n	N	noon
		ŋ	G	NX	sing
	Vocalic	m	M	EM	some
		n	N	EN	son
Affricates		tʃ	C	CH	church
		ʤ	J	JH	just
Others	Whisper	h	h	HH	help
	Vocalic	ɾ	F	DX	batter
	Glottal stop	ʔ	Q	Q	