EHB 328E – Machine Learning for Signal Processing

040160237 – İlker Baltacı

040160050 – Ahmet Bedri Yorulmaz

040160206 – Burak Şimşek

```
musicw = audioread('musicf1.wav');
speechw = audioread('speechf1.wav');
mixedw = audioread('mixedf1.wav');
```

First of all, we read our audio files.

```
music_spectrum = stft(musicw', 2048, 256, 0, hann(2048));
music_spec = abs(music_spectrum);


speech_spectrum = stft(speechw', 2048, 256, 0, hann(2048));
speech_spec = abs(speech_spectrum);
```

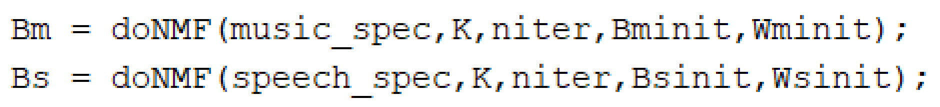
After that, we used our stft function from our previous homeworks for creating the spectrum of the audios.

```
% ADD CODE TO CPMPUTE MAG. SPECTROGRAM AND PHASE OF MIXED
mixed_spectrum = stft(mixedw', 2048, 256, 0, hann(2048));
mixed_spec = abs(mixed_spectrum);
mixed_phase = mixed_spectrum ./ (abs(mixed_spectrum) + eps);
```

Spectrum and phase signals of mixed audio has been created.

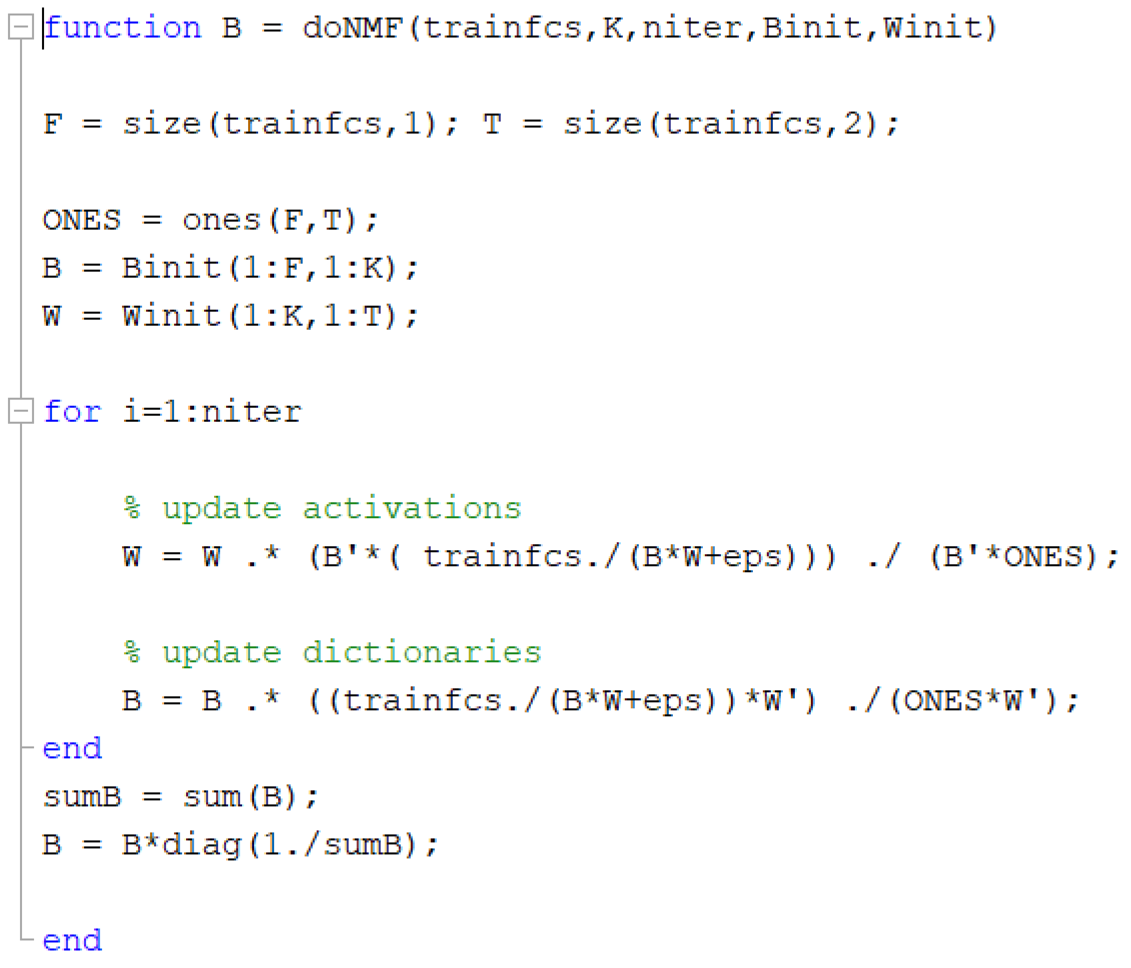
```
K = 200;
niter = 250;

Bminit = load('Bminit.mat');
Bminit = Bminit.Bm;
Wminit = load('Wminit.mat');
Wminit = Wminit.Wm;

Bsinit = load('Bsinit.mat');
Bsinit = Bsinit.Bs;
Wsinit = load('Wsinit.mat');
Wsinit = Wsinit.Ws;
```

Base and weight matrices of music and speech audios has been determined.



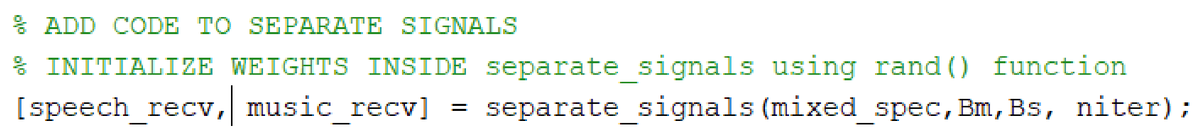
```
Bm = doNMF(music_spec,K,niter,Bminit,Wminit);
Bs = doNMF(speech_spec,K,niter,Bsinit,Wsinit);
```

doNMF function:



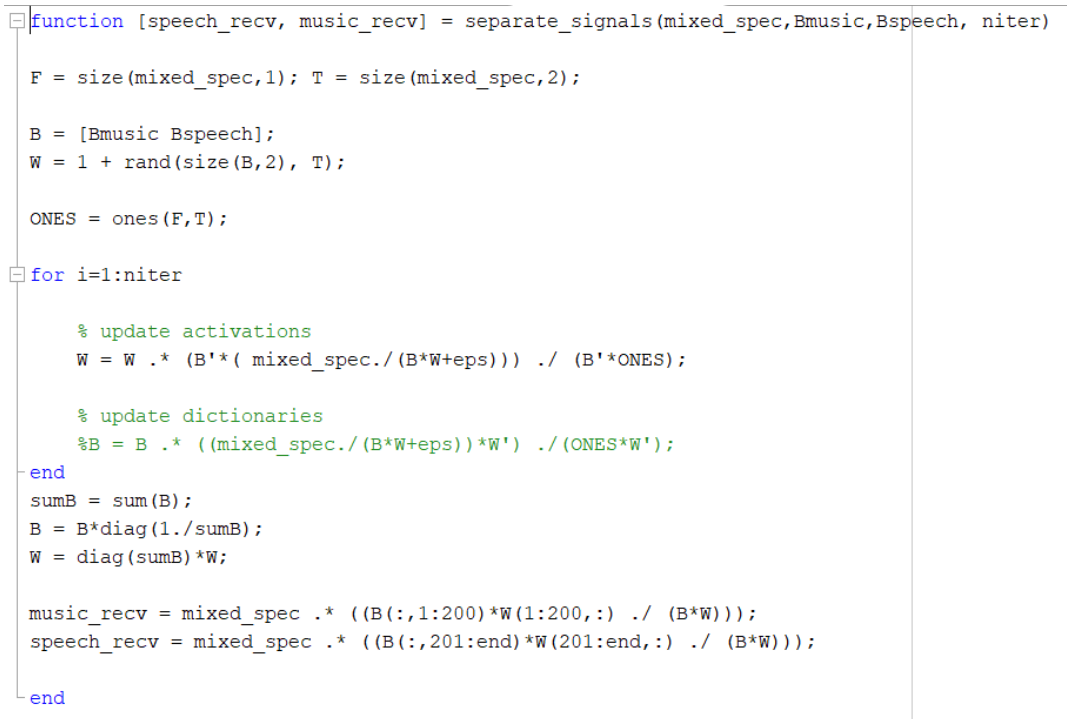
```
function B = doNMF(trainfcs,K,niter,Binit,Winit)

F = size(trainfcs,1); T = size(trainfcs,2);

ONES = ones(F,T);
B = Binit(1:F,1:K);
W = Winit(1:K,1:T);

for i=1:niter

    % update activations
    W = W .* (B'*( trainfcs./(B*W+eps))) ./ (B'*ONES);

    % update dictionaries
    B = B .* ((trainfcs./(B*W+eps))*W') ./(ONES*W');
end
sumB = sum(B);
B = B*diag(1./sumB);

end
```

Multiplication of base and weight matrices gives us our sound. Iterating the B init and W init matrices, we extract our reference base matrix in doNMF function that we created.



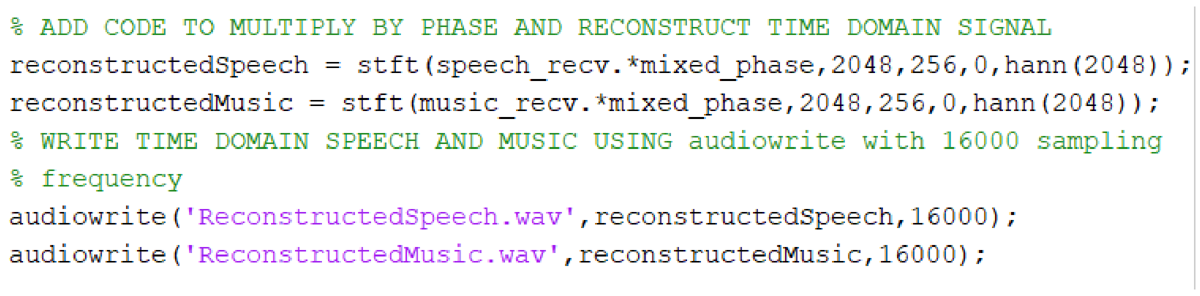
```
% ADD CODE TO SEPARATE SIGNALS
% INITIALIZE WEIGHTS INSIDE separate_signals using rand() function
[speech_recv, music_recv] = separate_signals(mixed_spec,Bm,Bs, niter);
```

Signal_seperate function:



```
function [speech_recv, music_recv] = separate_signals(mixed_spec,Bmusic,Bspeech, niter)

F = size(mixed_spec,1); T = size(mixed_spec,2);

B = [Bmusic Bspeech];
W = 1 + rand(size(B,2), T);

ONES = ones(F,T);

for i=1:niter

    % update activations
    W = W .* (B'*( mixed_spec./(B*W+eps))) ./ (B'*ONES);

    % update dictionaries
    %B = B .* ((mixed_spec./(B*W+eps))*W') ./(ONES*W');
end
sumB = sum(B);
B = B*diag(1./sumB);
W = diag(sumB)*W;

music_recv = mixed_spec .* ((B(:,1:200)*W(1:200,:) ./ (B*W)));
speech_recv = mixed_spec .* ((B(:,201:end)*W(201:end,:) ./ (B*W)));

end
```

Since our music and speech reference base are already set, we combine them and iterate them with random weight values. After each iteration, we updated our weight matrix. After we obtained our final weight matrix, we seperate the base and weight matrices to obtained our seperated music and speech audios.



```
% ADD CODE TO MULTIPLY BY PHASE AND RECONSTRUCT TIME DOMAIN SIGNAL
reconstructedSpeech = stft(speech_recv.*mixed_phase,2048,256,0,hann(2048));
reconstructedMusic = stft(music_recv.*mixed_phase,2048,256,0,hann(2048));
% WRITE TIME DOMAIN SPEECH AND MUSIC USING audiowrite with 16000 sampling
% frequency
audiowrite('ReconstructedSpeech.wav',reconstructedSpeech,16000);
audiowrite('ReconstructedMusic.wav',reconstructedMusic,16000);
```

Using mixed phase values, we reconstructed our seperated music and speech files.