Report Homework 2
Computer Music - Representations and models

Alice Portentoso

January 9, 2024

1 Introduction

The aim of this homework is to develop a music source separation system specifically for chorales. We have to decomposing a mixed audio wav file into its constituent canonical voices found in chorale compositions.

Music source separation refers to the process of isolating individual instruments or vocal parts from a combined audio signal. In the case of chorales, which typically feature distinct voice parts, the goal is to extract the soprano, alto, tenor, and bass voices from the overall mixture.

2 Question 1

In this question we work with chorale number 1 and we have to extract the annotations. By annotations we mean information relating to the music, for example note duration and note pitch.

2.1 Import and annotation function

The code starts with the import of the libraries and the load of the audio file related to chorale mix number 1, *Chorale BWV 360 written by J. S. Bach.*

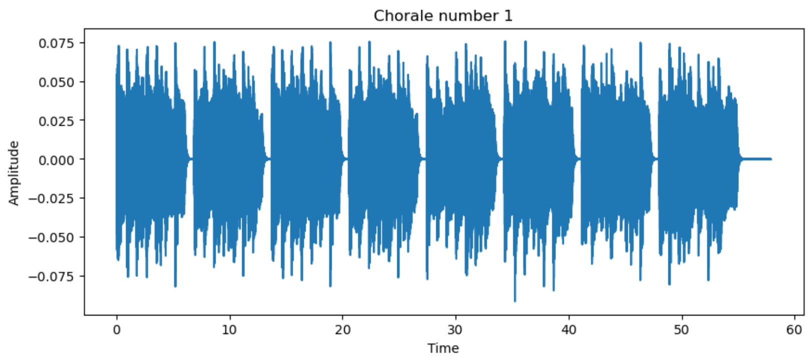
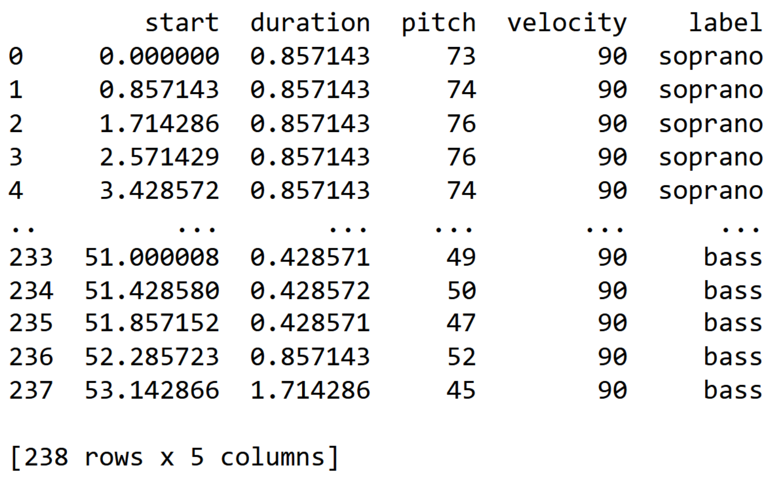


Figure 1: Chorale 1 wav file

We define and call the function `extract_annotations`. Starting from the midi files associated to each source, the function creates a list composed by other lists in this form: `[start, duration, pitch, velocity, label]`. This is the Dataframe associated.

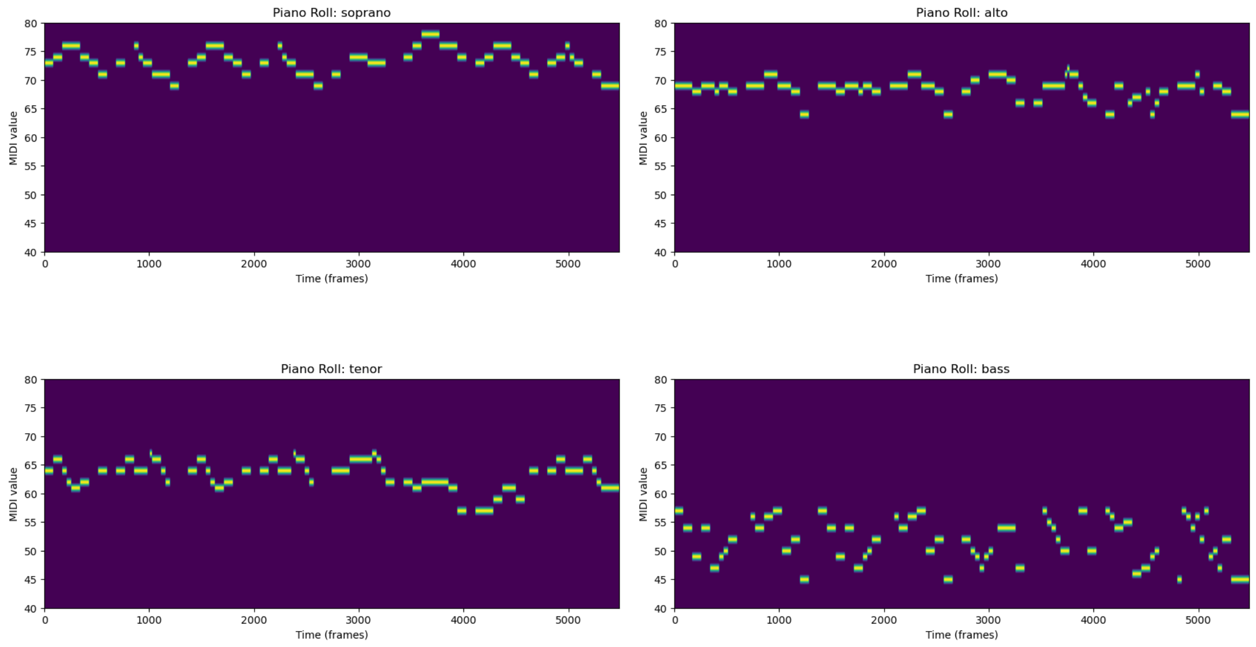


```
          start  duration  pitch  velocity    label
0      0.000000  0.857143     73        90  soprano
1      0.857143  0.857143     74        90  soprano
2      1.714286  0.857143     76        90  soprano
3      2.571429  0.857143     76        90  soprano
4      3.428572  0.857143     74        90  soprano
..          ...       ...    ...       ...      ...
233   51.000008  0.428571     49        90     bass
234   51.428580  0.428572     50        90     bass
235   51.857152  0.428571     47        90     bass
236   52.285723  0.857143     52        90     bass
237   53.142866  1.714286     45        90     bass

[238 rows x 5 columns]
```

Figure 2: DataFrame annotations

2.2 Piano Roll

These are the piano roll plot for every source, obtained from midi files.



3 Question 2

In question 2 we accomplish separation with Non-Negative-Factorization method.

3.0.1 Short-Time Fourier Transform

First of all we compute the Short-Time Fourier Transform (STFT) of the audio file x, and apply logarithmic compression on the magnitude.

```
    X =librosa.stft(x,n_fft=N_fft, win_length=N_fft,
            hop_length=H_fft, window='hann')
    V = np.log10(1 + np.abs(X))
```

3.1 Non-Negative Matrix Factorization

We use the Non-Negative Matrix Factorization (NMF) method. The basic idea behind NMF is to factorize a given matrix into two lower-rank matrices, where all elements are constrained to be non-negative.

Given a magnitude spectrum `V` (real matrix `KxN`) and `P` (natural number), classical NMF derives two non-negative matrices W (real positive matrix `K×P`) and `H` (real positive matrix `PxN`) such that a distance `D(V, W*H)` is minimized.

We initialize the activation matrix and the template matrix. We instantiate the NMF model and perform NMF on the spectrogram V. In the end we compute the spectrogram approximation by a matrix moltiplication between two matrices W and H.

```
    # Initialize activations and templates
    H_init,pitch_set,label_pitch = nmf_utils.initialize_activation(N, annotations,
     frame_res, tol_note=[0.1,0.5], tol_onset=[0.2,0.1])
    W_init = nmf_utils.initialize_template(K, pitch_set, freq_res, tol_pitch=0.05)

    # Instantiate NMF model
    model = decomposition.NMF(n_components=H_init.shape[0],
            init='custom', solver='mu', max_iter=1000)
    W = model.fit_transform(X=V.astype(np.float64), H=H_init, W=W_init)
    H = model.components_

    # NMF factorization
    approximated_spectrogram = np.dot(W, H)
```

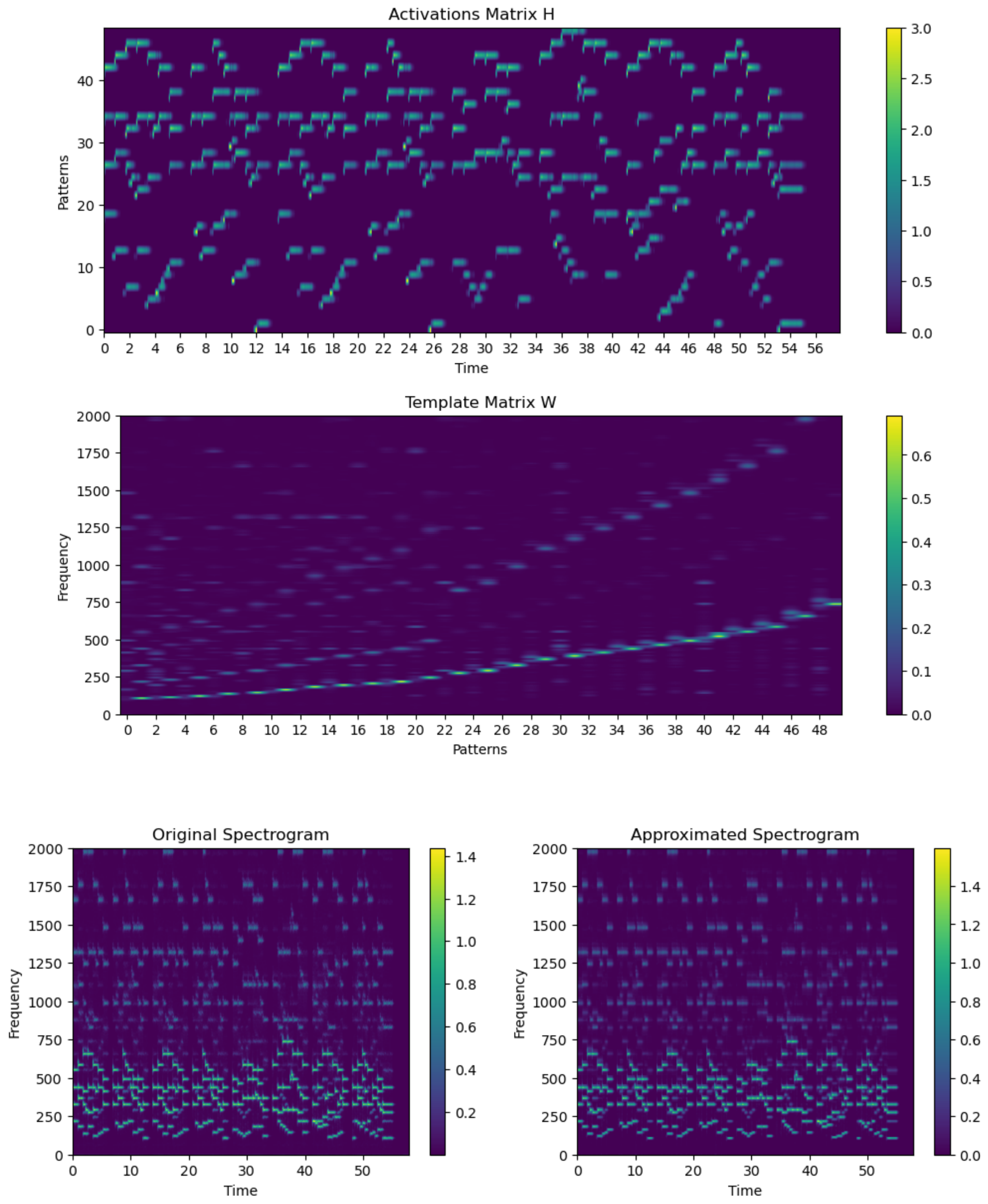


Figure 3: Activate matrix, template matrix, original and approximated spectrogram of signal

The H matrix gives information about the activation of notes in time, where patterns are divided. The plot is similar to the piano roll and the horizontal lines represent the sustained notes in time. The W matrix provides information about the spectral characteristics related to the patterns, which are related to pitch set.

Below, there is the comparison between the original compressed spectrogram V and the matrix product of W and H. The two plots look very similar, this means that the factorization occurred successfully. I can improve that by act for example on `tol_note, tol_onset, tol_pitch` parameters.

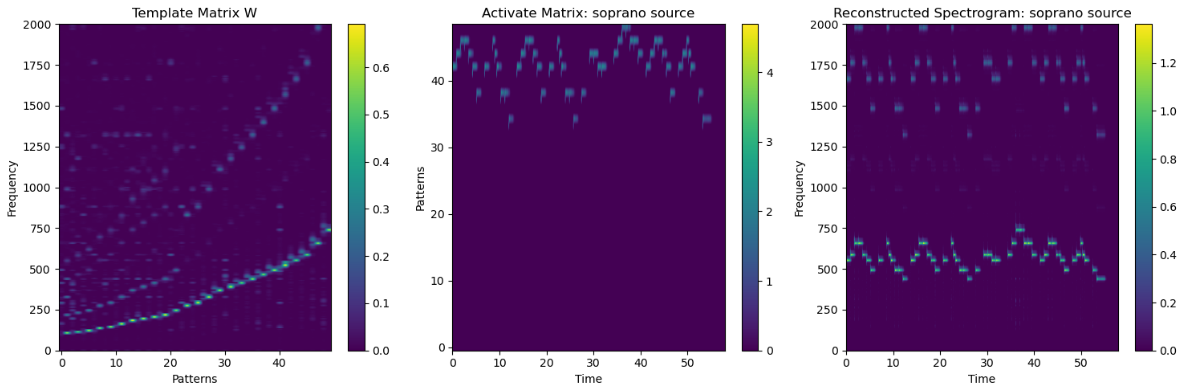
4 Question 3

Now we can apply the NMF model for source separation. In this question we create some spectral masks and applied them to the activation matrix in order to extract a first example of sources.

4.1 Split annotations and spectral masks

We start by splitting the annotations list in 4 others annotations lists containing the different sources. Then we obtain the four spectral masks by computing the following code lines, and apply them to the activation matrix H.

```
    # Obtain the spectral masks for each source
    H_init_sop,_,_ = nmf_utils.initialize_activation(N, ann_sop, frame_res,
        tol_note=[0.1,0.5], tol_onset=[0.2,0.1], pitch_set=pitch_set)
    H_init_alt,_,_ = nmf_utils.initialize_activation(N, ann_alt, frame_res,
        tol_note=[0.1,0.5], tol_onset=[0.2,0.1], pitch_set=pitch_set)
    H_init_ten,_,_ = nmf_utils.initialize_activation(N, ann_ten, frame_res,
        tol_note=[0.1,0.5], tol_onset=[0.2,0.1], pitch_set=pitch_set)
    H_init_bas,_,_ = nmf_utils.initialize_activation(N, ann_bas, frame_res,
        tol_note=[0.1,0.5], tol_onset=[0.2,0.1], pitch_set=pitch_set)

    # Apply spectral masks
    H_sop = H * H_init_sop
    H_alt = H * H_init_alt
    H_ten = H * H_init_ten
    H_bas = H * H_init_bas
```



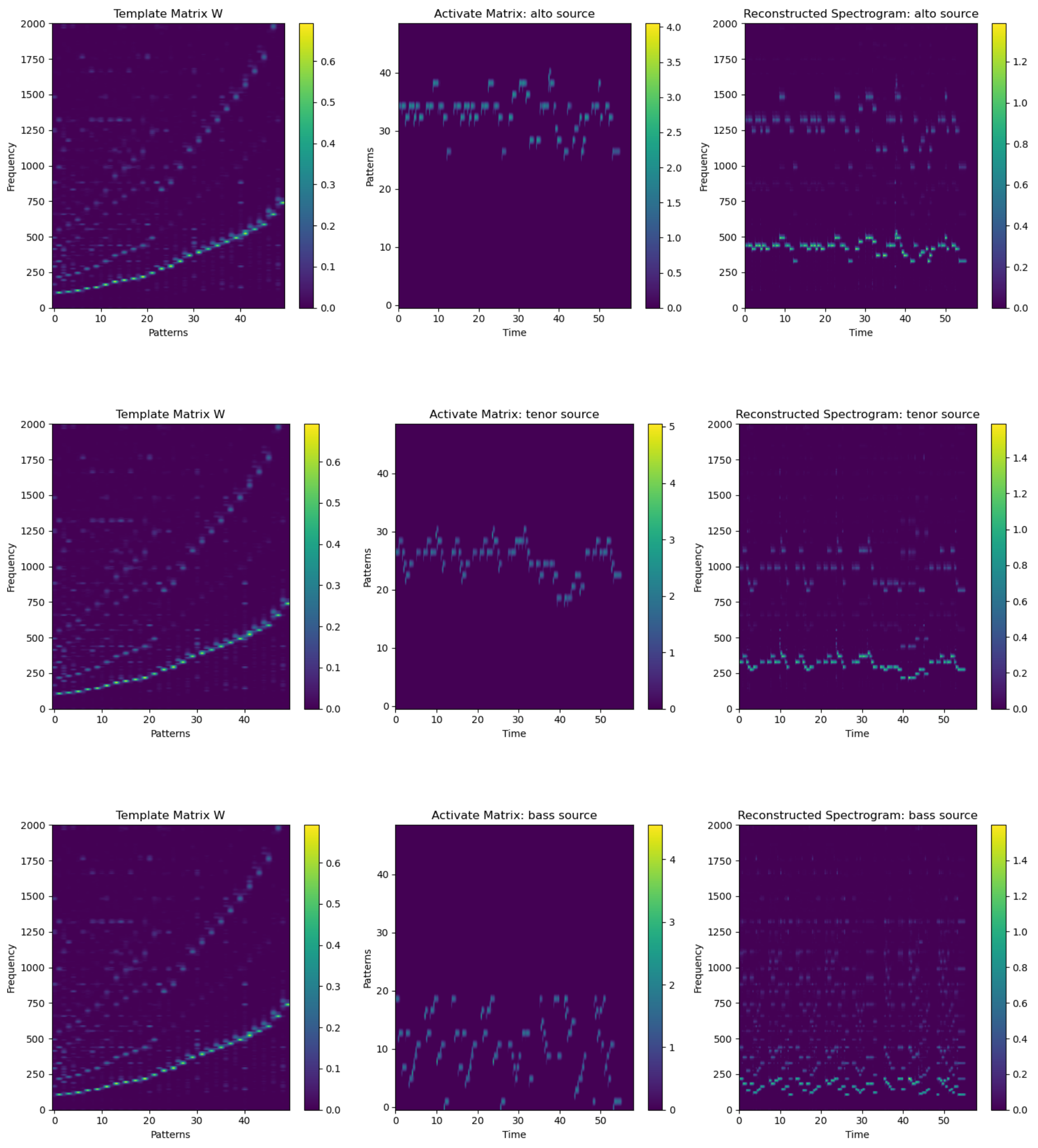


Figure 4: For each source: Template Matrix W, Activation matrix of source, reconstructed signal

We can notice in the activation matrices that pitch set is larger for lower sources. In the reconstructed signals we can see the resonating frequencies: for each fundamental frequency sang, we see, lighter, the presence of its resonating components on its multiple frequencies. For this reason we see more harmonics at resonant frequencies for the lower notes, therefore for tenor and bass.

5 Question 4

NMF-based models provide only a rough approximation of the original magnitude spectrogragram, details are not captured and the audio components reconstructed in this way may contain a number of audible artifacts. Some of these artifacts may be removed or attenuated by considering another masking technique. Instead of directly using the source activation matrices, we use them to compute soft masks. In the case of soprano voice, soft mask is defined in this way:

$$M_{\text{sop}} = (WH_{\text{sop}}) \oslash (WH + \varepsilon)$$

Than we apply this masks to the original spectrogram by a multiplication between them. In the end, we compute the Inverse STFT to obtain the source audio waveform in time-domain.

```
M_sop = np.dot(W,H_sop) / (np.dot(W,H)+eps)
x_sop = librosa.istft(X*M_sop)
M_ten = np.dot(W,H_ten) / (np.dot(W,H)+eps)
x_ten = librosa.istft(X*M_ten)
M_alt = np.dot(W,H_alt) / (np.dot(W,H)+eps)
x_alt = librosa.istft(X*M_alt)
M_bas = np.dot(W,H_bas) / (np.dot(W,H)+eps)
x_bas = librosa.istft(X*M_bas)
```

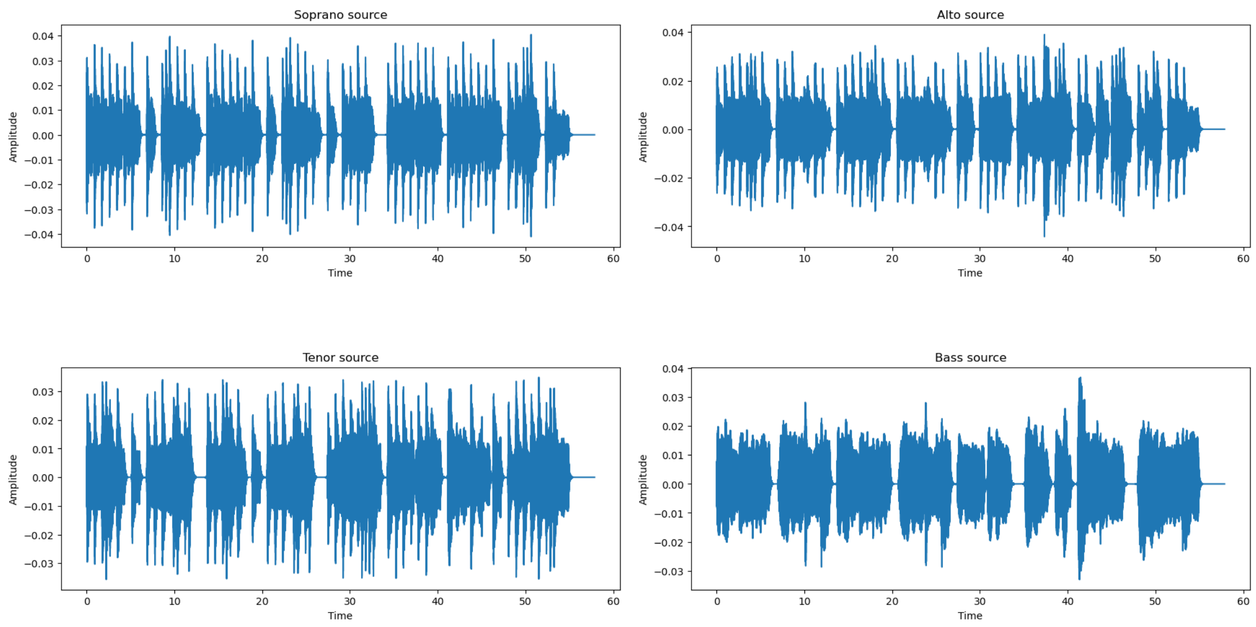


Figure 5: Audio divided into sources in time domain

Listening to the results, we hear that the separation occurred very well for the soprano and alto voices, with some noise for the tenor voice and quite badly for the bass. This difference can be explained due to the distance between the harmonics, which is lower for lower fundamental frequencies, as we can see in the right side in Figure 4.

Therefore due to resonance frequencies overlapping with the fundamental frequencies of other sources, this makes correct separation more difficult.

6 Question 5

In question 5 we are asked to pack everything together inside one single function `separate`, and apply it to different input files. Then we use metrics to quantify the separation performance.

6.1 Signal-To-Distortion-Ratio

Signal-To-Distortion Ratio measures the ratio between the power of the target signal and the introduced interference or artifacts in audio source separation. A higher SDR indicates better source separation performance because it means that the energy of the true source signals is much higher than the energy of the distortion introduced during the separation process.

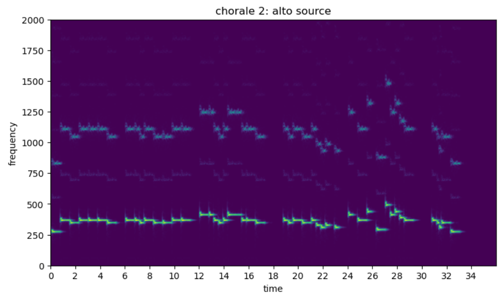
Printing the result we can notice the decrease in value for the lower voices in each source.

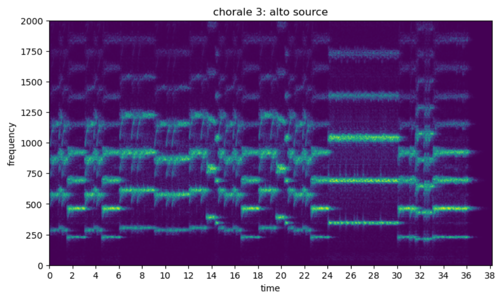
```
Chorale number 1
 Soprane Source  SDR = [18.49518937]
 Alto Source     SDR = [15.96896244]
 Tenor Source    SDR = [13.5769954]
 Bass Source     SDR = [7.44896368]

Chorale number 2
 Soprane Source  SDR = [8.27662898]
 Alto Source     SDR = [8.75356102]
 Tenor Source    SDR = [7.91332929]
 Bass Source     SDR = [6.77779371]
```

```
Chorale number 3
 Soprane Source  SDR = [3.46823939]
 Alto Source     SDR = [0.12298601]
 Tenor Source    SDR = [-0.48861804]
 Bass Source     SDR = [-0.87051897]

Chorale number 4
 Soprane Source  SDR = [5.67322477]
 Alto Source     SDR = [-0.02698159]
 Tenor Source    SDR = [-2.1311843]
 Bass Source     SDR = [-1.90897163]
```

Separation in chorale 1 and 2 occurred much better than 3 and 4 and the cause is the quality of original audio files. We can see the difference in the original separate spectrograms: in the chorale 1 and 2 notes are clear and with few harmonics but in chorale 3 and 4 a lot of harmonics are present and the total spectrogram appears noisy. It makes separation difficult with our algorithm.

For example, we can see below the compare between chorale 2 and 3 for the same alto source: while on the left the voice is simple and clean, on the right we have a lot of harmonics, which overlap with fundamental frequency and some harmonics of the others voices.


chorale 2: alto source
frequency
time


chorale 3: alto source
frequency
time