知能情報メディア実験
「複数話者の同時発話音声からの個別音声の抽出」最終レポート

情報科学類, 3 年, 2 クラス, 学籍番号：201311403　山岡 洸瑛, Yamaoka Kouei

はじめに

知能情報メディア実験「複数話者の同時発話音声からの個別音声の抽出」の最終レポートとして，与えられた各課題の結果と考察をまとめる．このレポート内において，スカラーは x，ベクトルは太字で $\boldsymbol{x}$, 行列は大文字の太字で $\boldsymbol{X}$ と表し，行列の i 行 j 列成分を X_{ij} と表すとする．また単位行列を $\boldsymbol{I}$ とする．ソースコードについては，単純な計算のみのものや，与えられた式をそのまま用いたものなどは示さない．独自に作成したものや，説目にに必要なもののみ示す．また作成した音源については TA の杉本さんに確認していただいた．

課題 1: 瞬時混合 ICA の実装

ステップ 1

2 つの音声サンプルを $\boldsymbol{x_1}, \boldsymbol{x_2}$ とし，式 (1) の混合行列を用いて (2) の式で混合した．行列の形から 1 行 2 列成分と 2 行 1 列成分の絶対値が小さいため，出力 $\boldsymbol{y_1}, \boldsymbol{y_2}$ について，$\boldsymbol{y_1}$ は $\boldsymbol{x_1}$ の，$\boldsymbol{y_2}$ は $\boldsymbol{x_2}$ の音声が他方に比べて振幅が大きくなることが予想される．

結果

作成した混合を再生すると，確かに混合されていた．また，予想通り $\boldsymbol{y_1}$ においては $\boldsymbol{x_1}$ の音が $\boldsymbol{x_2}$ よりも大きく聞こえ，$\boldsymbol{y_2}$ についても $\boldsymbol{x_2}$ の方が大きく聞こえた．

考察

結果より，音声をベクトル化した数値で扱うことができるという事がわかった．また，音声ベクトルに対して演算を行うことが可能という事もわかった．使用した混合行列の値を式 (3) のように変更して混合したところ，予想通り $\boldsymbol{y_1}$ では $\boldsymbol{x_2}$ の，$\boldsymbol{y_2}$ では $\boldsymbol{x_1}$ の音が大きくなった．これより，瞬時混合は式 (4) のように単純な線形結合で表され，かつ各々の係数によって音声の大きさが決まるという事が確認できた．これは音源の数が増えても $\boldsymbol{H}$ が I 行 J 列になるだけであり，本質的には変わらないと言える．

$$\boldsymbol{H} = \begin{pmatrix} 0.9 & 0.4 \\ -0.3 & 1 \end{pmatrix} \tag{1}$$

$$\begin{pmatrix} \boldsymbol{y_1} \\ \boldsymbol{y_2} \end{pmatrix} = \boldsymbol{H} * \begin{pmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \end{pmatrix} \tag{2}$$

$$\boldsymbol{H} = \begin{pmatrix} 0.1 & 0.9 \\ 0.9 & 0.1 \end{pmatrix} \tag{3}$$

$$\begin{cases} \boldsymbol{y_1} = H_{11} * \boldsymbol{x_1} + H_{12} * \boldsymbol{x_2} \\ \boldsymbol{y_2} = H_{21} * \boldsymbol{x_1} + H_{22} * \boldsymbol{x_2} \end{cases} \tag{4}$$

ステップ 2

混合行列 $\boldsymbol{H}$ の逆行列を用いて分離信号を作成し，それが元の音源と一致することを確認する．分離は混合音声 $\boldsymbol{y_1}, \boldsymbol{y_2}$ を用いて式 (5) のように行った．分離した音声を $\boldsymbol{s_1}, \boldsymbol{s_2}$ とした．

結果

実際に分離を行い，元の音源と一致することが確認しできた．

考察

この実験より，行列によって混合された音源は，その逆行列によって分離できるという事が確認できた．これより，混合行列が不明であってもその逆行列の近似的な値がわかれば分離できると予想される．

$$\begin{pmatrix} \boldsymbol{s_1} \\ \boldsymbol{s_2} \end{pmatrix} = \boldsymbol{H^{-1}} * \begin{pmatrix} \boldsymbol{y_1} \\ \boldsymbol{y_2} \end{pmatrix} \tag{5}$$

ステップ 3

瞬時混合 ICA を用いて分離行列 $\boldsymbol{W}$ を学習する．インフォマックスという手法を用いて，式 (6) と式 (7) のような勾配法の更新を繰り返すことによって学習する．ここで活性関数としては式 (8) を用いた．音源が 2 つの場合，$\boldsymbol{I}$ は $2 * 2, i = 1, 2$ であり，N は音源のフレーム数である．$\boldsymbol{W}$ の初期値は式 (9) とし，ステップサイズ $\mu = 0.1$, 活性関数で用いる適当な正の値 $\gamma = 0.5$ とした．

結果

学習した結果推定された分離行列 $\boldsymbol{W}$ を得た．確認のため $\boldsymbol{W} * \boldsymbol{H}$ を計算したところ，1 行 2 列成分と 2 行 1 列成分が共に 0 に近くなったため，近似的な値は求まっていると言える．ただし 1 行 1 列成分，2 行 2 列成分が大きく，単位行列からは大きく離れていた，また，学習は勾配法で行ったため，収束条件を $\boldsymbol{W}$ のすべての要素について更新前の値と更新後の値の差の絶対値が 10^{-4} 以下となること，とし反復計算を行った結果，今回のパラメータの場合 179 回の反復で終了した．

考察

インフォマックスという手法を用いると分離行列が求まることが確認できた．この分離行列で実際に分離できるのかどうかはわからないが，結果で確認したように正しい逆行列の近似にはなっているようなので，分離できることが期待される．ただし，単位行列に近づかなかったことや，$\boldsymbol{W}$ の要素が非常に大きいことから，元の音源よりも音量が大きくなっていると予想され，これがスケーリングの問題であると言える．

$$\begin{pmatrix} \boldsymbol{y_1} \\ \boldsymbol{y_2} \end{pmatrix} = \boldsymbol{W} * \begin{pmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \end{pmatrix} \tag{6}$$

$$\boldsymbol{W} \leftarrow \boldsymbol{W} + \mu \left(\boldsymbol{I} - \frac{1}{N} \sum_{n=0}^{N-1} \psi(\boldsymbol{y_i}[n]) \boldsymbol{y_i}[n]^t \right) \boldsymbol{W} \tag{7}$$

$$\psi(\boldsymbol{x}) = \left(tanh\left(\gamma x_1\right), \cdots, tanh\left(\gamma x_i\right) \right)^t \tag{8}$$

$$\boldsymbol{W} = \begin{pmatrix} 0.3 & 0.7 \\ 0.5 & 1 \end{pmatrix} \tag{9}$$

ステップ 4

ステップ 1 で作成した混合音声と，ステップ 3 で作成した分離行列を用いて実際に分離を行った．この時スケーリング解決を行った．また分離性能を上げるために数回分離行列学習時のパラメータを調整した．スケーリング解決及び分離は式 (10) で行った．$\boldsymbol{y_i}[n], \boldsymbol{W}$ はステップ 3 で得た値で，$\boldsymbol{Z}[n]$ はスケーリング解決を行った分離信号である．diag() は引数を対角要素とする対角行列を表す．

結果

分離し，スケーリング解決を行った音声を聞くと元の音源と全く同じように聞こえた．サンプル音源 1 の裏でサンプル音源 2 が少し聞こえる，などという事はなく，完全に分離されていた．スケーリングの解決もでき元の音源と同じような音量となっていた．

考察

ステップ 3 で考察したように分離行列を推定することで分離することができた．パラメータの調整は必要だが，瞬時混合の場合，正確な逆行列でなくても，非常に綺麗に分離できるといえる．ここまでの計算は音源の数を 2 つとして行なってきたが，音源の数が変わったとしても同じように計算できることは使用した計算式からわかる．これより瞬時混合ならば，混合音声を分離することができると言える．

$$\boldsymbol{Z}[n] = \boldsymbol{W}^{-1} * diag(\boldsymbol{y_i}[n]) \tag{10}$$

ステップ 5

混合行列が与えられていない未知の瞬時混合波形に対して ICA を行い音源分離を行え，という課題であったが未知の瞬時混合音声が取得出来なかったため，混合行列 $\boldsymbol{H}$ の値を変えて混合音声を作成し，$\boldsymbol{H}$ が未知であるとして音源分離を行った．混合行列を作成した翌週に分離を行うことで混合行列を未知であるとした．実際に混合行列の値は覚えていなかったため，未知であったとしても良いと考える．

結果

ステップ 4 同様に分離できた．しかしステップ 4 と違い混合行列の情報がないとしたためパラメータの調節に時間がかかってしまった．これは混合行列が未知であるため，分離行列との積を求めることで確認する，という方法がとれなかったため，目安がなかったことが原因である．時間がかかってしまったものの分離できたためステップ 4 の考察で述べたように瞬時混合音声であれば分離できると言える．

課題 2-1: 畳み込み混合の分析

0.

課題 2-1 で使用する畳み込み信号を自分で作成するという追加課題を行った．2 音源 2 マイク環境下での畳み込みを行った．インパルス応答は与えられたものを使い，音源には課題 1 で使用したサンプル音源をそのまま使用した．

音源の長さを N, インパルス応答の長さを R として $K = N + R - 1$ とする．まず畳み込みを周波数領域で行うために音源とインパルス応答を離散フーリエ変換 (以下 DFT とする) する．DFT の方法としては高速フーリエ変換 (以下 FFT とする) を用いる．音源 1,2 を K 点 FFT したものを s_1, s_2, 音源 i からマイク j へのインパルス応答を K 点 FFT したものを imp_{ij} とする．また周波数領域でのインデックスを f とすると，畳み込みは式 (11) で表される．畳み込み混合信号を x_i として，x_i を逆フーリエ変換 (以下 iFFT とする) することで時間領域の畳み込み混合信号を得る．

結果

畳み込み混合信号を得た．実際に聞いてみると音声に少しディレイがかかっているように聞こえ，畳み込みに成功していることが確認できた．

考察

周波数領域における畳み込みを実装したところ非常に高速に動くことが確認できた．時間領域の畳み込みを MATLAB 関数の conv() で行ったところメモリが足りずに計算出来なかったことから，周波数領域での畳み込みの威力が分かる．また K 点 FFT をする理由は，畳み込みによって $R-1$ だけ音声が伸びるからである．これは実際に短い信号で畳み込みを行なってみると確認できる．以上より以下の課題で使用する畳み込み混合信号が作成できた．

$$\begin{pmatrix} \boldsymbol{x_1}[f] \\ \boldsymbol{x_2}[f] \end{pmatrix} = \begin{pmatrix} imp11[f] & imp21[f] \\ imp12[f] & imp22[f] \end{pmatrix} * \begin{pmatrix} \boldsymbol{s_1}[f] \\ \boldsymbol{s_2}[f] \end{pmatrix} \tag{11}$$

1.

作成した畳み込み混合信号を短時間フーリエ変換 (以下 STFT とする) を用いて分析する．また逆 STFT(以下 iSTFT とする) によって元の波形に戻ることを確認する．

実装

フレーム長 p, フレームシフトを $ol = p * \frac{1}{o}$, 総フレーム数を al, 窓関数を $\boldsymbol{w}$ とする．これらを用いて p フレーム分 $\boldsymbol{w}$ をかけて取り出し FFT, ol だけずらして同様の計算を行う．これを繰り返すことで STFT が実装できる．iSTFT では切り出したフレームを iFFT して加算，ol だけずらして同様の計算を行うことで実装できる．しかし，STFT の時にフレームシフトや窓関数によってオーバーラップが生まれてしまい，その場合このままでは iSTFT は正しく行えていない．例えば $ol = \frac{1}{2}$ で窓関数がハミング窓の場合，オーバーラップはなくこのままで良いが，窓関数が箱形窓の場合，オーバーラップは 2 となり，前後 $\frac{p}{2}$ 以外は振幅が 2 倍になってしまう．これを解決するためにカウント用配列を用意する．これは切り出した回数をカウントする配列で，信号の第 k インデックスを n 回切り出した，という情報を格納する．iSTFT 時にこの配列を用いて割ることでオーバーラップ問題を解決できる．これに関する図を図 1 に示した．以上より STFT，iSTFT が実装できた．ソースコードをリスト 1, 2 に示す．ただし，X は STFT した信号，Y は iSTFT した信号で，count がカウント用配列である．なお，MATLAB の都合上 ifft() を用いてもごく小さな値が虚数として残ってしまうことがあるため，real() 関数を用いて虚数を消してある．

結果

STFT を行い iSTFT によって元の音源に戻ることが確認できた．音源信号をプロットした図を図 2 に示す．左側が元の音源 s_1, s_2 で右側が STFT,iSTFT した音源である．これより左右の信号がほぼ一致していることから，視覚的にも成功していることが確認できた．

考察

図 2 の左右を見比べると僅かに異なっている部分が稀にある．これは計算機で計算しているために丸め誤差などの僅かな誤差が生じた結果だと考えられる．左右の音源を聞いてみると，その差は知覚出来なかったため，この程度の誤差であれば無視しても構わないと言える．また今回の実装で STFT,iSTFT をそれぞれ関数化したため，今後 STFT 分析は関数 1 つで行えるようになった．

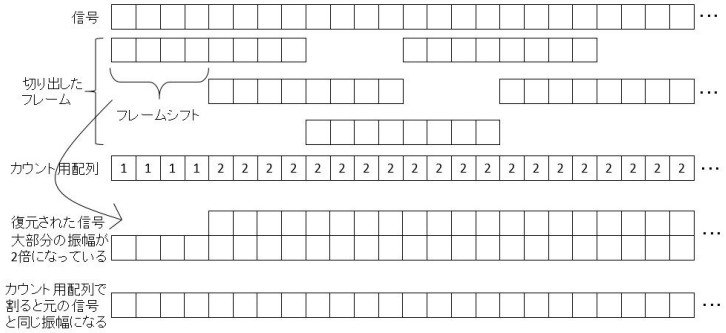

図 1: オーバーラップ問題の解決

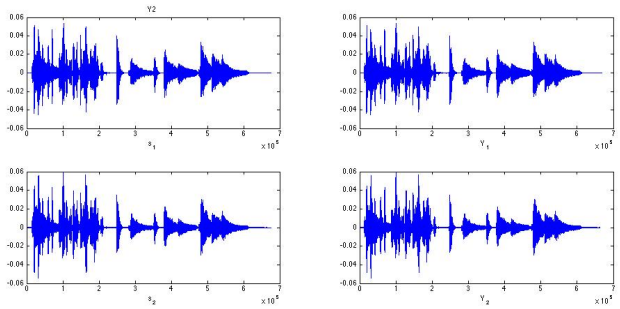

図 2: 元の音源と STFT,iSTFT 後の音源

リスト 1: stft.m

```
for k = 1:al
    % 窓をかけ、fft
    X(k,:) = s(1 + ol*(k-1) : p + ol*(k-1)) .* w';
    X(k,:) = fft(X(k,:));
    % 今のカウント用配列のインデックスに1加算
    count(1 + ol*(k-1) : p + ol*(k-1)) = count(1 + ol*(k-1) : p + ol*(k-1)) + 1;
end
```

リスト 2: istft.m

```
for k = 1:al
    % ifftし窓をとる
    Y(1 + ol*(k-1) : ol*(k-1) + p) = Y(1 + ol*(k-1) : ol*(k-1) + p) + real(ifft(X(k,:))) ./ w';
end
Y = Y ./ count;
```

2.

与えられたインパルス応答を DFT し周波数毎の分離行列を作成する．また STFT した混合信号を分離行列を用いて分離し，それを iSTFT することで時間領域の分離信号を得る．各変数は 1. と同じとし，サンプル音源 1,2 を STFT したものを $\boldsymbol{X_1}, \boldsymbol{X_2}$ とする．ただし i 行に第 i フレームが格納されている．また，音源 i からマイク j へのインパルス応答を p 点 FFT したものを imp_{ij} とする．分離した信号を $\boldsymbol{X_i}$ と同じサイズの行列 $\boldsymbol{Y_1}, \boldsymbol{Y_2}$ に格納すると式 (12) で分離することができる．ただし周波数領域のインデックスは f で表している．計算した $\boldsymbol{Y_i}$ を iSTFT

することで時間領域の分離信号を得ることができる．STFT, iSTFT は 1. で作成した関数を用いるが，iSTFT において 1 部変更を加えた．STFT した信号に分離行列をかけているため窓関数で割る必要がなくなったので，割らないように変更をした．

結果

分離に成功した，プロットした図を図 3 に示す．これより分離されていることが視覚的にも確認できる．しかし本来無音であるべき時間であっても僅かに振幅が存在することも確認できる．これは分離の過程で生じてしまったノイズである．これは分離した音声を聞いても聞き取れる程度には大きく．無視することはできない．

考察

時間周波数領域での分離に成功したため，ノイズを考慮しなければ今後 DFT の代わりに STFT を使うことができる．問題のノイズが生じた理由について考察する．フーリエ変換は切り出した範囲が周期関数であることを仮定して計算しているため，実装した STFT においては窓関数をかけることによってフレームの両端が綺麗につながるようにしている．しかし，切り出したフレームは実際には周期関数ではなく，そのの続きは今のフレームとは異なる信号であるため，周期関数を仮定して DFT すると実際とは異なる周波数領域の情報が得られてしまう．これを各フレームについて行なっているためすべてのフレームについてこのことが言える．STFT 分析した信号をそのまま iSTFT するのであれば各フレーム毎に fft() し ifft() しただけなので一致する．しかし，今回は STFT 分析した信号に分離行列をかけるという演算を行なっているため，誤差が生じたまま分離を行なっていることになる．そのため分離信号には誤差があり，それを iSTFT しても本来の時間領域の分離信号は得られず，誤差，つまりノイズがのった信号が得られてしまう．これが理由であると考えられる．

これを解決する方法は考えつかなかった．この問題は各フレーム毎の誤差が原因であるため，フレーム長を長くすればするほどノイズは減少すると予想される．実際にフレーム長を長くして実行した結果を図 4 に示す．外形自体が変わってしまっている部分があるが，ノイズ部分は減少していることが読み取れる．音声のノイズも減少していると感じた．これより解決はできていないが軽減はできたと言える．しかし STFT には不確定性原理があり，フレーム長を長くすると周波数分解能は良いが時間分解能は低くなってしまっている．フレーム数を最大にすればこの問題は最も軽減されるが，それは DFT であり，STFT する意味がなくなってしまう．これよりこの問題の根本的な解決方法が見つからなかった．

$$\begin{pmatrix} \boldsymbol{Y_1}[f] \\ \boldsymbol{Y_2}[f] \end{pmatrix} = \begin{pmatrix} imp_{11} & imp_{21} \\ imp_{12} & imp_{22} \end{pmatrix}^{-1} * \begin{pmatrix} \boldsymbol{X_1}[f] \\ \boldsymbol{X_2}[f] \end{pmatrix} \tag{12}$$

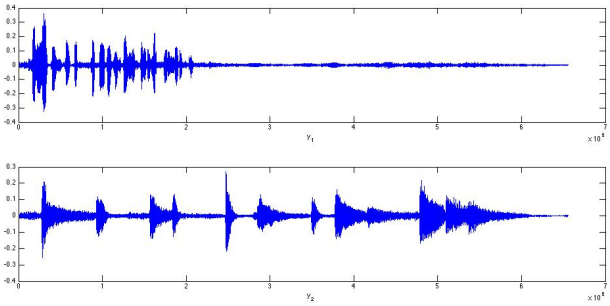

図 3: 時間領域の分離信号

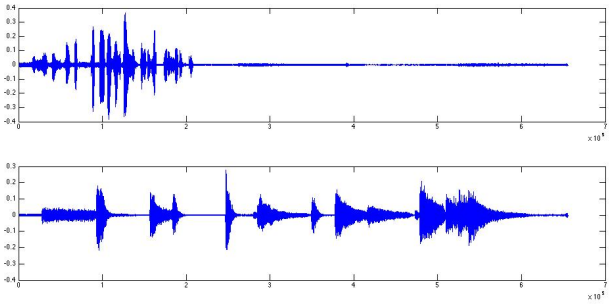

図 4: フレーム長を 4 倍にした場合

課題 2-2: IVA による音源分離

1.

IVA を実装し分離を行う．最大周波数を $\frac{L}{2}$ とする．分離行列 $\boldsymbol{W}$ の学習には課題 1 ステップ 3 と同様にインフォマックスを用いる．その時の分離信号 $\boldsymbol{Y_i}$ の更新は式 (13) であり，勾配法の更新は式 (14) である．ただしこのときの活性関数は式 (15) である．活性関数で使用している $\boldsymbol{normY_i}[m]$ は式 (16) で計算される．またスケーリングの解決は式 (17) で行う．ここで $\boldsymbol{X_i}$, $\boldsymbol{Y_i}$ は第 i 音源の時間周波数領域表現で，周波数インデックス f, 第 m フレームについて $\boldsymbol{X_i}[f,m]$ などと表す．音源数が増えた場合も同様の式で計算できる．

$$\begin{pmatrix} \boldsymbol{Y_1}[f,m] \\ \boldsymbol{Y_2}[f,m] \end{pmatrix} = \boldsymbol{W} * \begin{pmatrix} \boldsymbol{X_1}[f,m] \\ \boldsymbol{X_2}[f,m] \end{pmatrix} \tag{13}$$

$$\boldsymbol{W}[f] \leftarrow \boldsymbol{W}[f] + \mu \left(\boldsymbol{I} - \frac{1}{M} \sum_{m=0}^{M} \psi(\boldsymbol{Y_i}[f,m]) \boldsymbol{Y_i}[f,m]^H \right) \boldsymbol{W}[f] \tag{14}$$

$$\psi(\boldsymbol{X}[f,m]) = \gamma \left(\frac{Y_1[f,m]}{||normY_1[m]||_2}, \cdots, \frac{Y_J[f,m]}{||normY_J[m]||_2} \right)^t \tag{15}$$

$$||normY_j[m]||_2 = \sqrt{\sum_{f=0}^{\frac{L}{2}} |\boldsymbol{Y_j}[f,m]|^2} \tag{16}$$

$$\boldsymbol{Z} = \boldsymbol{W}[f]^{-1} * diag([\boldsymbol{Y_1}[f,m], \boldsymbol{Y_2}[f,m]]) \tag{17}$$

実装

式 (14)〜式 (17) を実装する．しかしこのまま実装すると f,m についてループ計算が必要になるため for 文を多用することになり計算時間が非常に長くなってしまう．そのため式変形と MATLAB 関数を利用して計算速度を速くする．

まず式 (13) については音源毎に f,m についての 2 重ループを計算することになるが，MATLAB 関数を使うことで複数の音源を同時にかつ f に関するループをなくして計算することができる．つまり音源数 $*f*m$ 回の計算を m 回の計算まで減らすことができる．まず $\boldsymbol{X_i}, \boldsymbol{Y_i}$ について，行が m, 列が f となっているが，これを 2 行 m 列 f ページの 3 次元配列に変更し，i 行目に $\boldsymbol{X_i}, \boldsymbol{Y_i}$ を格納する．この 3 次元配列を新たに $\boldsymbol{X}, \boldsymbol{Y}$ とする．また $\boldsymbol{W}$ を 2 行 2 列 f ページの 3 次元配列とする．これを第 f ページについて式に表すと式 (18) となる．この行列計算はこのままだと実行できない．$\boldsymbol{W}[f]$ を対角要素とする対角行列を用意し，$\boldsymbol{X}$ を転置することで計算はできるが，メモリを大量に消費するため，ここは for 文を使うことにする．第 m 列に着目すると式 (19) となり，for 文で実装できる．

$$\begin{pmatrix} Y_1[f,1] & \cdots & Y_1[f,M] \\ Y_2[f,1] & \cdots & Y_2[f,M] \end{pmatrix} = \begin{pmatrix} W_{11}[f] & W_{12}[f] \\ W_{21}[f] & W_{22}[f] \end{pmatrix} * \begin{pmatrix} X_1[f,1] & \cdots & X_1[f,M] \\ X_2[f,1] & \cdots & X_2[f,M] \end{pmatrix} \tag{18}$$

$$\begin{aligned} \begin{pmatrix} Y_1[f,m] \\ Y_2[f,m] \end{pmatrix} &= \begin{pmatrix} W_{11}[f] & W_{12}[f] \\ W_{21}[f] & W_{22}[f] \end{pmatrix} * \begin{pmatrix} X_1[f,m] \\ X_2[f,m] \end{pmatrix} \\ &= \begin{pmatrix} W_{11}[f] * X_1[f,m] + W_{12}[f] * X_2[f,m] \\ W_{21}[f] * X_1[f,m] + W_{22}[f] * X_2[f,m] \end{pmatrix} \end{aligned} \tag{19}$$

しかし行列ならば簡単に計算できる式だが，3 次元配列の場合 MATLAB では計算することができない．そのためこの式を MATLAB で計算できるように式変形すると式 (20) となり，これは式 (19) と一致する．そしてこの式は 3 次元配列のまま MATLAB で計算できる．これによって f に関する for 文の撤去に成功した．ソースコードはリスト 3 に示す．

$$\begin{aligned} & sum\left(\begin{pmatrix} W_{11}[f] & W_{12}[f] \\ W_{21}[f] & W_{22}[f] \end{pmatrix} .* \begin{pmatrix} X_1[f,m] & X_1[f,m] \\ X_2[f,m] & X_2[f,m] \end{pmatrix}^t , 2 \right) \\ = \; & sum\left(\begin{pmatrix} W_{11}[f] * X_1[f,m] & W_{12}[f] * X_2[f,m] \\ W_{21}[f] * X_1[f,m] & W_{22}[f] * X_2[f,m] \end{pmatrix} , 2 \right) \\ = \; & \begin{pmatrix} W_{11}[f] * X_1[f,m] + W_{12}[f] * X_2[f,m] \\ W_{21}[f] * X_1[f,m] + W_{22}[f] * X_2[f,m] \end{pmatrix} \end{aligned} \tag{20}$$

次に式 (14)〜(16) について考える．まず，式 (20) で計算した $\boldsymbol{Y}$ からそれぞれの信号を取り出し，m 行 f 列の $\boldsymbol{Y_1}, \boldsymbol{Y_2}$ とする．これを用いると，要素毎に 2 乗し，sum() を使い，sqrt() を使うことで式 (21) となり，式 (16) を得る．これより式 (16) には f,m どちらの for 文もいらず，音源毎の計算をすれば良い．

$$(||normY_j[1]||_2, \cdots, ||normY_j[M]||_2) = \left(\sqrt{\sum_{f=0}^{\frac{L}{2}} |\boldsymbol{Y_j}[f,1]|^2}, \cdots, \sqrt{\sum_{f=0}^{\frac{L}{2}} |\boldsymbol{Y_j}[f,M]|^2} \right) \tag{21}$$

式 (15),(16) を用いて式 (14) を音源を同時にかつ f に関する for 文のみで計算できるようにする．つまり計算回数が音源数 $2 * f * m$ 回であったのを f 回に減らす．まず式 (14) 中の $\psi(\boldsymbol{Y_i}[f,m])\boldsymbol{Y_i}[f,m]^H$ について式変形をする．以下の式変形において，MATLAB 関数の sum(), 複素共役をとる conj(), 内積を計算する dot() を用いる．なお以下では音源数を J としている．

$$\begin{aligned} \psi(\boldsymbol{Y_i}[f,m])\boldsymbol{Y_i}[f,m]^H &= \gamma \begin{pmatrix} \frac{Y_1[f,m]}{normY_1[m]} \\ \vdots \\ \frac{Y_J[f,m]}{normY_J[m]} \end{pmatrix} \begin{pmatrix} conj(Y_1[f,m]), & \cdots, & conj(Y_J[f,m]) \end{pmatrix} \\ &= \gamma \begin{pmatrix} \frac{Y_1[f,m]*conj(Y_1[f,m])}{normY_1[m]}, & \cdots, & \frac{Y_1[f,m]*conj(Y_J[f,m])}{normY_1[m]} \\ \vdots & \ddots & \vdots \\ \frac{Y_J[f,m]*conj(Y_1[f,m])}{normY_J[m]}, & \cdots, & \frac{Y_J[f,m]*conj(Y_J[f,m])}{normY_J[m]} \end{pmatrix} \\ \sum_{m=1}^{M} \psi(\boldsymbol{Y_i}[f,m])\boldsymbol{Y_i}[f,m]^H &= \gamma \begin{pmatrix} \sum_{m=1}^{M} \frac{Y_1[f,m]*conj(Y_1[f,m])}{normY_1[m]}, & \cdots, & \sum_{m=1}^{M} \frac{Y_1[f,m]*conj(Y_J[f,m])}{normY_1[m]} \\ \vdots & \ddots & \vdots \\ \sum_{m=1}^{M} \frac{Y_J[f,m]*conj(Y_1[f,m])}{normY_J[m]}, & \cdots, & \sum_{m=1}^{M} \frac{Y_J[f,m]*conj(Y_J[f,m])}{normY_J[m]} \end{pmatrix} \\ &= \gamma \begin{pmatrix} \frac{dot\left(\boldsymbol{Y_1}[f],\ conj(\boldsymbol{Y_1}[f])\right)}{sum(\boldsymbol{normY_1})}, & \cdots, & \frac{dot\left(\boldsymbol{Y_1}[\boldsymbol{f}],\ conj(\boldsymbol{Y_J}[\boldsymbol{f}])\right)}{sum(\boldsymbol{normY_1})} \\ \vdots & \ddots & \vdots \\ \frac{dot\left(\boldsymbol{Y_J}[f],\ conj(\boldsymbol{Y_1}[f])\right)}{sum(\boldsymbol{normY_J})}, & \cdots, & \frac{dot\left(\boldsymbol{Y_J}[\boldsymbol{f}],\ conj(\boldsymbol{Y_J}[\boldsymbol{f}])\right)}{sum(\boldsymbol{normY_J})} \end{pmatrix} \end{aligned} \tag{22}$$

式 (22) において $dot\left(\boldsymbol{Y_j}[f],\ conj(\boldsymbol{Y_j}[f])\right)$ は，行列 $\boldsymbol{Y_j}[f,m]$ の第 f 列を取り出したベクトル $\boldsymbol{Y_j}[f]$ とその複素共役の内積であるからスカラーであり，$sum(\boldsymbol{normY_1})$ はベクトルの要素の和なのでこちらもスカラーである．従っ

て $J * J$ の行列である．この行列を $\boldsymbol{\chi}[f]$ とおくと，式 (14) は以下のように表される．これを更に式変形する．

$$\begin{aligned}\boldsymbol{W}[f] &\leftarrow \boldsymbol{W}[f] + \mu\left(\boldsymbol{I} - \frac{1}{M}\boldsymbol{\chi}[f]\right)\boldsymbol{W}[f] \\ &= \left(\boldsymbol{I} + \mu\left(\boldsymbol{I} - \frac{1}{M}\boldsymbol{\chi}[f]\right)\right)\boldsymbol{W}[f]\end{aligned}$$

ここで改めて $\boldsymbol{I} + \mu\left(\boldsymbol{I} - \frac{1}{M}\boldsymbol{\chi}[f]\right)$ を $\boldsymbol{\chi}[f]$ とおくと，最終的に式 (14) は式 (23) と表すことができる．

$$\boldsymbol{W}[f] \leftarrow \boldsymbol{\chi}[f] * \boldsymbol{W}[f] \tag{23}$$

この計算は 3 次元配列であるため行えないが，第 i 列について 1 列ずつ行えば，式 (13) と同じ方法で実行できる．以上より全ての音源を同時に f のループのみで実行できるようになった．ソースコードはリスト 3 に示す．式 (22) を計算する関数 calcm.m はリスト 4 に示す．

これより分離行列の学習が実装できた．これを用いると，混合信号を STFT し分離行列を学習，学習した分離行列を用いて式 (13) のように分離，式 (17) でスケーリング解決を行い iSTFT することで時間領域の分離信号を得ることができる．STFT, iSTFT については実装済みなので省略する．

リスト 3: bunri.m

```
% 反復計算
for k=1:loop
    % 式(13)の計算
    for l = 1:fi
        Y(:,l,:) = sum(W .* repmat(permute(X(:,l,:),[2,1,3]), 2, 1), 2);
    end
    %Yの計算終了 ----------------------------------------------------

    % Wの更新 ------------------------------------------------
    % 準備
    Y1 = squeeze(Y(1,:,:)).';
    Y2 = squeeze(Y(2,:,:)).';
    % 式(16)の計算
    normY1 = sqrt(sum(abs(Y1).^2));
    normY2 = sqrt(sum(abs(Y2).^2));

    % X[f]の計算
    for i = 1:L2
        kai(:,:,i) = calcm(Y1(i,:), Y2(i,:), normY1, normY2, fi, mu, ganma);
    end

    % 式(23)の計算
    W(:,1,:) = sum(kai .* repmat(permute(W(:,1,:),[2,1,3]), 2,1), 2);
    W(:,2,:) = sum(kai .* repmat(permute(W(:,2,:),[2,1,3]), 2,1), 2);

    % Wの更新終了 ----------------------------------------------------
end

% 最後の式(13)の計算
for l = 1:fi
    Y(:,l,:) = sum(W .* repmat(permute(X(:,l,:),[2,1,3]), 2, 1), 2);
end
```

リスト 4: calcm.m

```
function tmpY = calcm(Y1, Y2, normY1, normY2, fi, mu, ganma)

ty1 = ganma * conj(Y1);
ty2 = ganma * conj(Y2);
tmpY = [sum((Y1.*ty1)./normY1), sum((Y1.*ty2)./normY1); ...
        sum((Y2.*ty1)./normY2), sum((Y2.*ty2)./normY2)];
tmpY = eye(2) + mu * (eye(2) - (1/fi) * tmpY);
```

結果

分離に成功した．$\boldsymbol{W}$ の初期値はシード値 509 で生成した乱数とした．μ, γ は分離精度を上げるために変更しつつ調整した．混合音声 $\boldsymbol{X_1}, \boldsymbol{X_2}$, 分離音声 $\boldsymbol{Y_1}, \boldsymbol{Y_2}$ をプロットした図を図**??**に示す．

考察

まず計算回数を減らすためにした式変形によってどの程度計算速度が上がったのかを確認する．残った for 文は f に関するものと m に関するもの両方あるため，for 文の計算速度はトレードオフの関係になってしまっている．今回はフレーム長を 1024 として実行した．この時，式 (13) は 0.2 秒短縮され，式 (14) については 6 秒程度短縮された．フレーム長を長くすると式 (13) は更に短縮されるが，式 (14) は長くなってしまう．これよりフレーム長にかかわらず約 6 秒の短縮になったと言える．ただし混合信号の長さが短くなると短縮される時間も短くなる．これは f,m どちらも小さくなるため for 文自体にかかる時間が減るためである．このことから今回の短縮は混合信号が長いほど威力を発揮すると言える．

また特に式 (13) などはわかりやすくするため音源数が 2 個の場合で式変形しているが，音源数が J 個の場合も同じ方法で実装することができる．これより畳み込み混合信号であっても分離行列を学習し分離できるようになった．ここまでで音源数とマイクの数が適していれば実際の環境下で混合音声を分離できるようになったはずである．必要な情報は話者の数とマイクの数，話者の数以上の混合信号のみである．

2.

この課題はやらない，という事なので行わなかった．配布するという関数も配布されなかったため実装出来なかった．

課題 2-3: 分離性能の客観・主観評価

1.

客観評価に使用する音源が配布されなかったため行わなかった．この課題もやらない，という事であった．

2.

実環境で録音した信号の分離を行う．録音した音声は 2 話者の場合と 3 話者の場合があり，その両方について分離を行った．分離には課題 2-2 で作成したプログラムをその儘使用した．3 話者の場合には同様の手法で音源数を 3 つに拡張したプログラムを使用した．分離した音源を用いて主観評価実験も行う．

結果

分離に成功した．ただしパラメータを調整したが，完全には分離できず 1 人が話している後ろで僅かに別の人間の話し声が残ってしまった．またノイズも大きく，それほど精度が高いとは言えないものとなった．しかし 1 人の話し声だけが強く聞こえるため成功と言えるだろう．2 話者の場合でもノイズなどが大きかったにも関わらず，3 話者のばあいはそれがさらに大きくなってしまい，精度は更に悪くなってしまった．混合信号によってはうまく分離できず，1 人の声が少し大きくなったぐらいにしかならないこともあり，3 話者の分離は困難であった．

考察

今回の分離でノイズが大きくなってしまった一因に元の混合信号にもノイズが乗っていたという事が挙げられる．これに関してはこちらではどうしようもないため，改善は難しいだろう．改善方法としてバンドパスフィルタを通してみた．人間が通常会話で発する音の周波数は 300〜2000Hz 程度の範囲に収まるため，それ以外の周波数をカットすることでノイズが減少するのではないか，という単純な考えから行なってみた．しかし実際にはノイズは減少せず，理由はわからなかったがノイズが大きくなってしまうこともあった．これより単純なバンドパスフィルタではノイズ除去は行えないことがわかった．ノイズの特性を調べてみるとはっきりしたことはわからなかったが，白色雑音も含まれているようで，これからも単純なフィルタでは除去できないことが分かる．

分離精度についてもパラメータを何度も調整したにも関わらず上がらないため，この分離方法ではそれほど精度

を得られないのだと思われる．分離音声と混合音声を聞いていると，話者の声の大きさなど，個人差があるためそれによって精度が変わってくるように感じられた．例えば声が比較的大きい人と小さい人の 2 話者で分離すると，大きい人の方が分離精度が高く，小さい人の場合はそれほど高くなくなってしまっている．これより分離は混合信号に強く左右される，と言える．高精度の分離には限界があり，高精度の混合信号が必要なのではないだろうか．

主観評価実験

最終レポートの提出日が 8/5 であり，主観評価実験を行うのも 8/5 であるため主観評価実験については書くことができない．

終わりに

この実験の結果，混合信号の分離を学ぶ過程で信号処理について理解が深まった．特に式で扱っただけのフーリエ変換を実際に行ったことで，利用方法を学ぶことができた．また混合信号分離についての知識と技術も手に入れることができた．ノイズの除去など課題は多いが，この実験の目標は達成できたと言えるだろう．