令和元年度

卒業論文

Raspberry Piを使用したドローンの駆動音低減法の検討

苫小牧工業高等専門学校 電気電子工学科 第16期生 18番

瀧澤 哲

指導教員 工藤 彰洋

Raspberry Piを使用したドローンの駆動音低減法の検討

論文要旨

本研究では, ドローンの応用分野の 1 つとして音声収録に着目し、ドローンの駆動音を低減する信号処理法の実装を目的とした。

駆動音の低減法を検討するにあたり, ドローン・バイノーラルマイクの組み立て, Raspberry Pi のセットアップなどハードウェアの準備を行った. Raspberry Pi には AD 変換が搭載されていないため, AD 変換器を内蔵した拡張ボードとして Pumpkin Pi を使用し, 初期設定を行った.

ソフトウェアに関しては, まず Python を使用して, 静的な FIR フィルタ・適応フィルタ（ADF）の評価を行った. 次に ADF のライブラリを Go 言語で作成し, インターネットにて公開した. また, Python・Go 言語を使用して波形表示や音声編集用のソフトウェアを制作した.

次にドローンの駆動音に対する各適応アルゴリズムの有効性を検証するために, 駆動音のサンプル収音を行い, ADF の収束特性を試験した. 結果的に NLMS・AP アルゴリズムに比べて RLS アルゴリズムの収束誤差は 20dB 小さいが, 収束速度が 4167ms と遅くリアルタイム処理には向かないことが判明した.

制作した ADF ライブラリのベンチマークを取ると, Raspberry Pi を計算媒体とした場合, 一番高速な NLMS アルゴリズムでも計算速度が 1.9ms/op が遅く, アクティブノイズコントロールの実装は難しいことが判明した.

最後に, 実際の使用の際に入力される音響信号を模擬し, ADF による雑音低減の効果を検討した. 音声の信号を抽出するのに成功したのは SN 比 0dB の RLS アルゴリズムと一部のフィルタ長の AP アルゴリズムのみであった.

以上より本研究で検討した Raspberry Pi を有する構成での雑音低減の実現可能性は低いことを結論付ける.

目 次

図 目 次

表 目 次

第1章　序論

1.1　研究背景

近年, ドローンの開発が進み, 着実に産業として根付いてきている. 「ドローン (Drone)」はもともと「無人機」全般を指す言葉であるが, 日本では慣例的に「マルチコプター (Multicopter)」を指す. ドローンは従来の有人ヘリコプターや大型機と比べて小型・軽量で, 低コストで製造が可能という特徴から, 空撮, 農業, 測量, 災害救助, デリバリー等, 様々な用途を想定して開発が進められている.

しかしながら, ドローンを使用するうえでは騒音の大きさ, 駆動時間の短さ, 悪天候に対する不適性といった技術的課題も多く残る.

本研究ではドローンの応用分野の 1 つとして音声収録に着目し, 収録した音声信号からドローンの駆動音を取り除く手法について検討する.

1.2　研究目的

本研究で想定する手法は大きく分けて, ドローンの駆動音に対して外部から逆位相の音を発生させ, 駆動音そのものを打ち消すことで低減する手法と, 計算機に入力された音を内部で処理することで駆動音を低減する手法の 2 つに分かれる.

これらの手法を実現するためには, 駆動音と目的信号の混合信号から, 駆動音のみを取り除く必要がある. この処理を行うためには, 動的にフィルタの係数を変化させる適応フィルタ（ADF, Adaptive Filter) を使用ことが一般的である.

適応フィルタのフィルタ係数を計算するアルゴリズムは複数知られているが, アルゴリズムの収束誤差と収束速度は相反する関係にある. したがって, 実装するハードウェアの規模や, 求められる収束速度などを考慮して, アルゴリズムを選択する必要がある.

本研究では, 計算機のハードウェアとして Raspberry Pi を, ソフトウェアとして Go 言語で制作したプログラムを実行し, 代表的な適応アルゴリズムを使用した適応フィルタによるドローンの駆動音低減の評価を行うことを目的とする.

第2章　理論

2.1　適応フィルタの基礎

2.1.1　ウィナーフィルタ

(1)　ウィナーフィルタの構造

ウィナーフィルタは線形MMSE法（最小平均二乗誤差法）を定常な時系列に適用した特殊型であり, 後述の適応アルゴリズムの基礎となる.

ウィナーフィルタのブロック図を図2.1.1に示す.

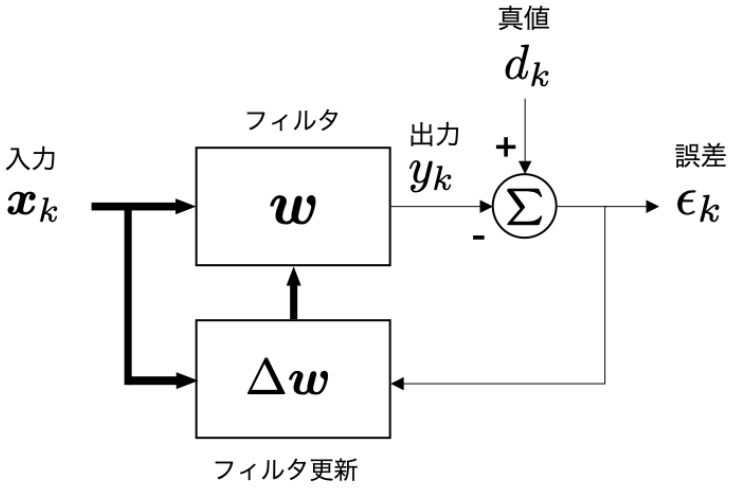

図 2.1.1 ウィナーフィルタのブロック図

ウィナーフィルタの入出力は次式で表される.

$$y_k = \boldsymbol{w}^H \boldsymbol{x}_k \tag{2.1.1}$$

ここで, $\boldsymbol{w}^H$ は次式で与えられるフィルタ係数ベクトルである.

$$\boldsymbol{w}^H = [w_1^*, \cdots, w_K^*]^T \tag{2.1.2}$$

一方, フィルタ入力は次式のような時系列 x_k を逆順に並べたものとなる.

$$\boldsymbol{x}_k = [x_k, x_{k-1}, \cdots, x_{k-K+1}]^T \tag{2.1.3}$$

これは, (2.1.1) 式により, 次式のような時間領域の畳み込み演算を表すためである.

$$y_k = \sum_{i=1}^{K} w_i^* x_{k-i+1} \tag{2.1.4}$$

これにより, 式はフィルタ係数 w_i^* を持つ時間領域の FIR フィルタを表すことになる. 入力時系列 x_k については, 平均値 $E[x_k] = 0$ の実数あるいは複素数を仮定している.

ウィナーフィルタで推定すべき値は, 実数あるいは複素数のスカラー量 d_k となる. d_k は望みの応答と呼ばれ, フィルタ出力 y_k により, 望みの応答を推定する. すなわち, $\hat{d}_k = y_k$ となる. フィルタの推定過程では, 次式で定義される推定誤差を, 次式で述べる規範のもとに最小化する.

$$\epsilon_k := d_k - y_k \tag{2.1.5}$$

上述のように, 適応フィルタの目的は, 入力信号 $\boldsymbol{x}_k$ から d_k を推定することである. これを達成するための過程として, フィルタ $\boldsymbol{w}$ の係数を決定する学習過程と, (2.1.1) 式により $\hat{d}_k$ を推定するフィルタリング過程とに分けられる. フィルタの学習過程では, 信号 d_k を教師として与え, d_k と $\boldsymbol{x}_k$ の関係を表す係数 $\boldsymbol{w}$ を学習する. 一方, フィルタリング過程では, 信号 d_k が未知の場合について, 観測値 $\boldsymbol{x}_k$ と学習済みのフィルタ係数 $\boldsymbol{w}$ から, 信号の推定値 $\hat{d}_k (= y_k)$ を得る. 学習過程で望みの応答 d_k をどうやって与えるかは, 応用に依存する [1].

(2) ウィナーフィルタの導出

ウィナーフィルタでは, 観測値 $\boldsymbol{x}_k$ から信号 d_k を推定する. フィルタの学習過程では, 次式の二乗平均誤差をコスト関数として最小化する.

$$\begin{aligned} J &= E[|\epsilon_k|^2] \\ &= E[(d_k - \boldsymbol{w}^H \boldsymbol{x}_k)(d_k - \boldsymbol{w}^H \boldsymbol{x}_k)^H] \\ &= E[d_k d_k^H] - \boldsymbol{w}^H E[\boldsymbol{x}_k d_k^H] - E[d_k \boldsymbol{x}_k^H] \boldsymbol{w} \end{aligned} \tag{2.1.6}$$

ここで次式を定義する.

$$\sigma_d^2 := E[d_k d_k^H], \boldsymbol{r}_{xd} := \boldsymbol{x}_k d_k^H, \boldsymbol{r}_{dx} := d_k \boldsymbol{x}_k^H, \boldsymbol{R}_x := E[\boldsymbol{x}_k \boldsymbol{x}_k^H] \tag{2.1.7}$$

$\boldsymbol{r}_{xd}$ および $\boldsymbol{r}_{dx}$ は, 入力ベクトル $\boldsymbol{x}_k$ と望みの応答 d_k との相互相関ベクトル, $\boldsymbol{R}_u$ は, $\boldsymbol{x}_k$ の自己相関行列である. これらを用いて (2.1.6) 式は次式のように書き直せる.

$$J = \sigma_d^2 + -\boldsymbol{w}^H \boldsymbol{r}_{xd} - \boldsymbol{r}_{dx} \boldsymbol{w} + \boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{w} \tag{2.1.8}$$

コスト関数を $\boldsymbol{w}^*$ について偏微分すると次式のようになる.

$$\frac{\partial J}{\partial \boldsymbol{w}^*} = -\boldsymbol{r}_{xd} + \boldsymbol{R}_x \boldsymbol{w} \boldsymbol{w} \tag{2.1.9}$$

(2.1.9) 式を $\boldsymbol{0}_{K\times 1}$ とおくと, 次式を得る.

$$\boldsymbol{R}_x \boldsymbol{w} = \boldsymbol{r}_x d \tag{2.1.10}$$

(2.1.10) 式は, 正規方程式あるいはウィナー・ホッフ方程式と呼ばれる. (2.1.10) 式を解くことにより最適フィルタは, 次式のように求まる [1].

$$\hat{\boldsymbol{w}}_{WF} = \boldsymbol{R}_u^{-1} \boldsymbol{r}_{xd} \tag{2.1.11}$$

2.1.2 適応フィルタにおける最小二乗法（LS 法）

適応フィルタにおける LS 法は, ウィナーフィルタの近似解を有限のサンプルから求めるものであり, 2.2.3 節で述べる RLS 法の基礎となっている.

適応フィルタにおける LS 法では, 次式に示す誤差の二乗和が最小化される.

$$J = \sum_{k=1}^{L_s} |\epsilon_k|^2 = \sum_{k=1}^{L_s} |d_k - \boldsymbol{w}^H \boldsymbol{x}_k|^2 \tag{2.1.12}$$

ここで, L_s はサンプル数である. ウィナーフィルタの導出過程における期待値をサンプル平均に置き換え, 同様に正規方程式を求めると次式が導かれる.

$$\hat{\boldsymbol{R}}_x \boldsymbol{w} = \hat{\boldsymbol{r}}_x d \tag{2.1.13}$$

これより, LS 法における最適解は, 次式のようになる [1].

$$\hat{\boldsymbol{w}}_{LS} = \hat{\boldsymbol{R}}_x^{-1} \hat{\boldsymbol{r}}_x d \tag{2.1.14}$$

2.1.3 最急降下法

2.1.1 節で述べたウィナーフィルタでは, 二乗平均誤差を最小とするフィルタ $\boldsymbol{w}$ を, 正規方程式の解として求めた. ここでは, この解を反復法を用いて逐次的に求める.

反復法では, フィルタ係数の初期値を適当に定め, コスト関数 $J(\boldsymbol{w})$ の最小点を目指して, フィルタ係数を少しずつ変化させていく. (2.1.8) 式で示したコスト関数の場合, $J(\boldsymbol{w})$ は w についての 2 次関数となり, 下に凸の誤差特性曲面となる. 最急降下法では, $k-1$ 回目の反復におけるフィルタ係数を $\boldsymbol{w}_{k-1}$ とした場合, $\boldsymbol{w}=\boldsymbol{w}_{k-1}$ での誤差特性曲面の勾配を推定し, この勾配と逆の方向にフィルタ係数を変化させる. $\boldsymbol{w}=\boldsymbol{w}_{k-1}$ における勾配ベクトルは (2.1.9) 式から, 次式のように求まる.

$$\nabla J(\boldsymbol{w}_{k-1}) := \frac{\partial J(\boldsymbol{w})}{\partial \boldsymbol{w}^*}|_{\boldsymbol{w}=\boldsymbol{w}_{k-1}} = -\boldsymbol{r}_{xd} + \boldsymbol{R}_x \boldsymbol{w}_{k-1} \tag{2.1.15}$$

次回の反復で勾配とは逆方向に係数を変化させるため, フィルタ係数ベクトルの変化分は次式のようになる.

$$\Delta \boldsymbol{w} = -\mu \nabla J(\boldsymbol{w}_{k-1}) \tag{2.1.16}$$

μ は 1 回の更新量を決定する性の定数であり, ステップサイズパラメータと呼ばれる. これを用いて, フィルタの更新式は, 次式のようになる.

$$\begin{aligned} \boldsymbol{w}_k &= \boldsymbol{w}_{k-1} + \Delta \boldsymbol{w} \\ &= \boldsymbol{w}_{k-1} + \mu(\boldsymbol{r}_{xd} - \boldsymbol{R}_x \boldsymbol{w}_{k-1}) \end{aligned} \tag{2.1.17}$$

また, (2.1.17) 式は次式のように書き直すことができる.

$$\begin{aligned} \boldsymbol{w}_k &= \boldsymbol{w}_{k-1} + \mu(E[\boldsymbol{x}_k d_k^*] - E[\boldsymbol{x}_k \boldsymbol{x}_k^H]\boldsymbol{w}_{k-1}) \\ &= \boldsymbol{w}_{k-1} + \mu E[\boldsymbol{x}_k (d_k^* - \boldsymbol{x}_k^H \boldsymbol{w}_{k-1})] \end{aligned} \tag{2.1.18}$$

ここで, 次式の事前推定誤差を定義する.

$$e_k := d_k - \boldsymbol{w}_{k-1}^H \boldsymbol{x}_k \tag{2.1.19}$$

(2.1.18) 式に (2.1.19) 式を代入して, 次式を得る [1].

$$\boldsymbol{w}_k = \boldsymbol{w}_{k-1} + \mu E[\boldsymbol{x}_k e_k^*] \tag{2.1.20}$$

2.2 代表的な適応アルゴリズム

2.2.1 最小二乗法平均法（LMS 法）

リアルタイムシステムを構築する場合などは, メモリや演算量に制約があることがある. 最急降下法の更新式はシンプルだが, 期待値演算をサンプル平均で実装することになり, メモリや演算量を必要とする. この点を改良したのが最小二乗平均法（LMS 法）である. LMS 法では, (2.1.20) 式における期待値演算を用いた勾配の推定を, 次式のように, 瞬時値の推定値に置き換える.

$$\boldsymbol{w}_k = \boldsymbol{w}_{k-1} + \mu \boldsymbol{x}_k e_k^* \tag{2.2.1}$$

また, (2.2.1) 式を入力のパワーで正規化したものは, 学習同定法（NLMS 法）と呼ばれる. NLMS 法は, 次式の拘束付き最適化問題から導くことができる.

$$\begin{aligned} & min||\boldsymbol{w}_k - \boldsymbol{w}_{k-1}||^2 \\ & \qquad \text{subject to } \boldsymbol{w}_k^H \boldsymbol{x}_k = d_k \end{aligned} \tag{2.2.2}$$

この最適化問題は, ラグランジュの未定乗数法を用いて解くことができる. ラグランジュの未定乗数法では, 次式のコスト関数を最小化する.

$$\begin{aligned} J &= ||\boldsymbol{w}_k - \boldsymbol{w}_{k-1}||^2 + Re(\lambda(d_k - \boldsymbol{w}_k^H \boldsymbol{x}_k)) \\ &= (\boldsymbol{w}_k - \boldsymbol{w}_{k-1})^H (\boldsymbol{w}_k - \boldsymbol{w}_{k-1}) + \lambda(d_k - \boldsymbol{w}_k^H \boldsymbol{x}_k) + (d_k - \boldsymbol{w}_k^H \boldsymbol{x}_k))^H \lambda^H \end{aligned} \tag{2.2.3}$$

コスト関数 J を $\boldsymbol{w}_k^*$ について偏微分すると

$$\frac{\partial J}{\partial \boldsymbol{w}_k^*} = \boldsymbol{w}_k - \boldsymbol{w}_{k-1} - \lambda \boldsymbol{x}_k \tag{2.2.4}$$

これを $\boldsymbol{0}_{K\times 1}$ とすることにより, 次式を得る.

$$\boldsymbol{w}_k - \boldsymbol{w}_{k-1} = \lambda \boldsymbol{x}_k \tag{2.2.5}$$

(2.2.5) 式の左から $\boldsymbol{x}_k^H$ を乗じ, λ について解くと

$$\lambda = \frac{1}{\boldsymbol{x}_k^H \boldsymbol{x}_k} \boldsymbol{x}_k^H (\boldsymbol{w}_k - \boldsymbol{w}_{k-1}) \tag{2.2.6}$$

これに拘束条件（(2.1.18) 式）および事前推定誤差（(2.1.19) 式）

$$\lambda = \frac{1}{||\boldsymbol{x}_k||^2}(d_k^* - \boldsymbol{x}_k^H \boldsymbol{w}_{k-1}) = \frac{1}{||\boldsymbol{x}_k||^2} e_k^* \tag{2.2.7}$$

(2.2.7) 式を再び (2.2.5) 式に代入することにより, フィルタの変化量 $\Delta \boldsymbol{w}$ は次式のように求まる.

$$\Delta \boldsymbol{w} = \frac{1}{||\boldsymbol{x}_k||^2} \boldsymbol{x}_k e_k^* \tag{2.2.8}$$

以上より, NLMS 方のフィルタベクトル更新式は次式のようになる.

$$\boldsymbol{w}_k = \boldsymbol{w}_{k-1} + \lambda \frac{1}{||\boldsymbol{x}_k||^2} \boldsymbol{x}_k e_k^* \tag{2.2.9}$$

実際の運用では, 入力信号のパワー $||\boldsymbol{x}_k||^2$ が非常に小さいときに更新式が不安定になるのを防ぐため, $1/||\boldsymbol{x}_k||^2$ の分母に小さい正の定数 α を加えた, 次式を用いる [1].

$$\boldsymbol{w}_k = \boldsymbol{w}_{k-1} + \lambda \frac{1}{\alpha + ||\boldsymbol{x}_k||^2} \boldsymbol{x}_k e_k^* \tag{2.2.10}$$

2.2.2 アフィン射影法（APA 法）

アフィン射影法は, NLMS 法には, NLMS 法における拘束付き最適化問題の拘束条件を次式のように複数に拡張することにより導くことができる.

$$\begin{aligned} &min||\boldsymbol{w}_k - \boldsymbol{w}_{k-1}||^2 \\ &\text{subject to } \boldsymbol{w}_k^H \boldsymbol{X}_k = d_k \end{aligned} \tag{2.2.11}$$

ここで, $\boldsymbol{X_k}$（$K \times L_s$ の行列）および $\boldsymbol{d_k}$（$1 \times L_s$ の行ベクトル）は次式で定義される.

$$\boldsymbol{d}_k := [d_{k-L_s+1}, \cdots, d_k] \tag{2.2.12}$$

$$\boldsymbol{X}_k := [\boldsymbol{x}_{k-L_s+1}, \cdots, \boldsymbol{x}_k] \tag{2.2.13}$$

$$\boldsymbol{x}_k := [\boldsymbol{x}_k, \cdots, \boldsymbol{x}_{k-L_s+1}]^T \tag{2.2.14}$$

拘束条件は, 次式の L_s 個の拘束条件を行列・ベクトル形式で表したものである.

NLMS 法と同様にラグランジュの未定乗数法を用いた, 最適解を導出により, 次式のフィルタ更新式が得られる [1].

$$\boldsymbol{w}_k = \boldsymbol{w}_{k-1} + \mu \boldsymbol{X}_k (\alpha \boldsymbol{I} + \boldsymbol{X}_k^H \boldsymbol{X})^{-1} e_k^H \tag{2.2.15}$$

2.2.3 再帰最小二乗法（RLS 法）

再帰最小二乗法（RLS 法）は. LS 法の解を再帰的に求める手法である.

サンプル平均による自己相関行列および相互相関ベクトルの推定値 $\hat{\boldsymbol{R}}_x$ および $\hat{\boldsymbol{r}}_{xd}$ を次式に示す.

$$\hat{\boldsymbol{R}}_k := \sum_{i=1}^{k} \boldsymbol{x}_i \boldsymbol{x}_i^H = \boldsymbol{X}_{1:k} \boldsymbol{X}_{1:k}^H \tag{2.2.16}$$

$$\hat{\boldsymbol{r}}_k := \sum_{i=1}^{k} \boldsymbol{x}_i d_i^* = \boldsymbol{X}_{1:k} \boldsymbol{d}_{1:k}^H \tag{2.2.17}$$

ここで, $\boldsymbol{d}_{1:k}$ および $\boldsymbol{X}_{1:k}$

$$\boldsymbol{d}_{1:k} := [d_1, \cdots, d_k] \tag{2.2.18}$$

$$\boldsymbol{X}_{1:k} := [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k] \tag{2.2.19}$$

$$\boldsymbol{x}_k := [x_k, \cdots, u_{k-K+1}]^T \tag{2.2.20}$$

時刻 k における $\hat{\boldsymbol{R}}_x$ および $\hat{\boldsymbol{r}}_x$ は, 一時刻前 $(k-1)$ の値を用いて, 次式のように再帰的に表すことができる.

$$\hat{\boldsymbol{R}}_k = \hat{\boldsymbol{R}}_{k-1} + \boldsymbol{x}_k \boldsymbol{x}_k^H \tag{2.2.21}$$

$$\hat{\boldsymbol{r}}_k = \hat{\boldsymbol{r}}_{k-1} + \boldsymbol{x}_k d_k^* \tag{2.2.22}$$

逆行列の補助定理を用いると, 自己相関行列の逆行列 $P_k := \hat{\boldsymbol{R}}_k^{-1}$ も, 次式のように再帰的に求めることができる.

$$\boldsymbol{P}_k = \boldsymbol{P}_{k-1} - \frac{\boldsymbol{P}_{k-1} \boldsymbol{x}_k \boldsymbol{x}_k^H \boldsymbol{P}_{k-1}}{1 + \boldsymbol{x}_k^H \boldsymbol{P}_{k-1} \boldsymbol{x}_k} \tag{2.2.23}$$

(2.2.22) 式, (2.2.23) 式 を用いて, 求めるべきフィルタ系ルウは, 次式のようになる.

$$\boldsymbol{w}_k = \boldsymbol{P}_k \hat{\boldsymbol{r}}_k \tag{2.2.24}$$

ここで, 更新式における演算を簡略化するため, 次式のゲインベクトルを定義する.

$$\boldsymbol{g}_k := \frac{\boldsymbol{P}_{k-1}\boldsymbol{x}_k}{1 + \boldsymbol{x}_k^H \boldsymbol{P}_{k-1}\boldsymbol{x}_k} \tag{2.2.25}$$

これを用いて (2.2.23) 式を書き直すと

$$\boldsymbol{P}_k = \boldsymbol{P}_{k-1} - \boldsymbol{g}_k \boldsymbol{x}_k^H \boldsymbol{P}_{k-1} = (\boldsymbol{I} - \boldsymbol{g}_k \boldsymbol{x}_k^H)\boldsymbol{P}_{k-1} \tag{2.2.26}$$

一方, (2.2.25) 式から, 次式を得る.

$$\boldsymbol{g}_k = (\boldsymbol{I} - \boldsymbol{g}_k \boldsymbol{x}_k^H)\boldsymbol{P}_{k-1}\boldsymbol{x}_k \tag{2.2.27}$$

(2.2.26) 式, (2.2.27) 式を用いて, ゲインベクトルは次式のように書き直せる.

$$\boldsymbol{g}_k = \boldsymbol{P}_k \boldsymbol{x}_k \tag{2.2.28}$$

(2.2.26) 式および (2.2.28) 式を (2.2.24) 式に代入することにより, フィルタ係数ベクトル $\boldsymbol{w}_k$ の更新式は, 次式のように求まる.

$$\begin{aligned} \boldsymbol{w}_k &= \boldsymbol{P}_k(\hat{\boldsymbol{r}}_{k-1} + \boldsymbol{x}_k d_k^*) \\ &= \boldsymbol{w}_{k-1} + \boldsymbol{g}_k(d_k^* - \boldsymbol{x}_k^H \boldsymbol{w}_{k-1}) \end{aligned} \tag{2.2.29}$$

最後に, (2.2.29) 式に (2.2.27) 式を代入して, 次式の更新式を得る [1].

$$\boldsymbol{w}_k = \boldsymbol{w}_{k-1} + \boldsymbol{g}_k e_k^* \tag{2.2.30}$$

2.2.4 適応アルゴリズムの比較

表 2.2.1 は代表的な適応アルゴリズムの関係をニュートン法を使用して整理したものである.

表 2.2.1 代表的な適応アルゴリズムの係数更新式

	$\Delta \boldsymbol{w}$
LMS	$\mu \boldsymbol{x}_k \boldsymbol{\epsilon}_k$
NLMS	$\mu \left(\alpha \boldsymbol{I} + \boldsymbol{x}_k \boldsymbol{x}_k^H\right)^{-1} \boldsymbol{x}_k \boldsymbol{\epsilon}_k$
AP	$\mu \boldsymbol{X}_k \left(\alpha \boldsymbol{I} + \boldsymbol{X}_k^H \boldsymbol{X}_k\right)^{-1} \boldsymbol{\epsilon}_k$
RLS	$\mu \boldsymbol{X}_k \left(\alpha \boldsymbol{I} + \boldsymbol{X}_{1:k}^H \boldsymbol{X}_{1:k}\right)^{-1} \boldsymbol{\epsilon}_k$

表 2.2.1 のアルゴリズムは下の手法ほど予測に用いるサンプル数が増加する. これに伴い, 一般に収束速度も速くなる. 一方, この代償として計算量が増大する. したがって, 実装するハードウェアの規模や, 求められる収束速度などを考慮して, アルゴリズムを選択する必要がある [2].

2.3　適応フィルタを使用した雑音低減の手法

2.3.1　アクティブノイズコントロール（ANC）

アクティブノイズコントロール（ANC）とは, マイクやスピーカ等の機器を利用し, 対象とする騒音と逆位相の音を重ね合わせることで音を低減する手法である.

ANC のブロック図を図 2.3.1 に示す.

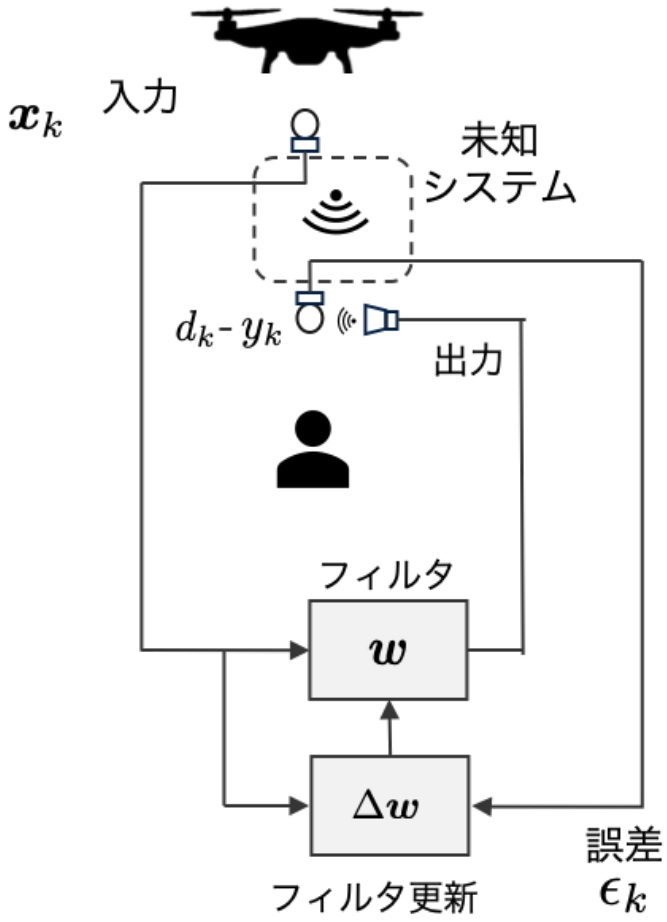

図 2.3.1 ドローンの駆動音に対する ANC の想定ブロック図

ドローンに駆動音に対する ANC は図 2.3.1 における参照信号 $\boldsymbol{x}_k$ と誤差信号 ϵ_k から ADF（$\boldsymbol{w}$）で未知システムを推定することで次サンプルの信号を予測し, スピー

カーなどの音響機器からフィルタ出力 y_k を逆位相で発生させて打ち消す動作を繰り返す.

したがって, 適応アルゴリズムの予測性能が駆動音抑圧の効果を決定づける要因となる.

2.3.2 自動等化器

自動等化器はアクティブノイズコントロールとは異なり, 計算機内部で信号の処理を行う手法である. 一般に, 適応フィルタに対する入力信号は長時間で平均をとると, ドローンの駆動音に対して音声のエネルギーが小さい. 自動等化器では, フィルタの更新係数を小さくすることでフィルタ係数がドローンの駆動音のみに適応することを利用し, フィルタ出力 y_k との誤差 e_k を目的信号の予測値として取り出す手法である.

自動等化器のブロック図を図 2.3.2 に示す.

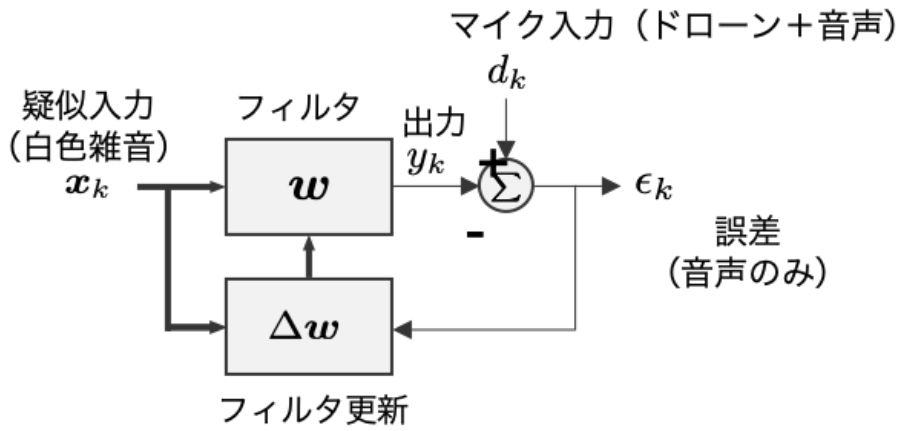

図 2.3.2 ドローンの駆動音に対する自動等化器の想定ブロック図

第3章　ハードウェアの製作

この章では本研究で使用するドローン, マイク, Raspberry Pi といったハードウェアの構成と製作について述べる.

3.1　ドローンについて

ドローンは Lynxmotion 株式会社の組み立て式ドローン (HQuad500) を使用した. このドローンは機体にカーボンファイバーを採用しており, 軽量, 高強度, 高剛性を兼ね備えている. また, 拡張性が高く, 容易に機能の追加が可能で様々な研究用途に適した製品である.

3.1.1　使用機器

使用機器を以下に示す.

(1) ドローンの機体 HQuad500 Hardware kit Lynxmotion 株式会社

図 3.1.1 ドローンの機体

(2) ESC (Electronic Speed Controller) 12A ESC (SimonK) Lynxmotion 株式会社

図 3.1.2 ESC

(3) ブラシレスモータ Brushless Motor 28x30 1000kv Lynxmotion 株式会社

図 3.1.3 ブラシレスモータ

(4) フライトコントローラ Quadrino Nano Lynxmotion 株式会社

図 3.1.4 フライトコントローラ

(5) リポバッテリー充電器 18W LiPo Battery Charger Lynxmotion 株式会社

図 3.1.5 リポバッテリー充電器

(6) リポバッテリー 11.1V (3S), 3500mAh 30C LiPo Battery Pack Lynxmotion 株式会社

図 3.1.6 リポバッテリー

(7) ラジオレシーバ R9DS 10 channels 2.4GHz DSSS FHSS Receiver RadioLink 株式会社

図 3.1.7 ラジオレシーバ

(8) トランスミッタ AT9S 2.4GHz 10CH transmitter RadioLink 株式会社

図 3.1.8 トランスミッタ

3.1.2 組み立ておよび動作確認

ドローンの組み立ておよび動作確認はフライトコントローラの説明書 [3] に従い, 次のように行った.

(1) 内容物の確認

(2) 機体の組み立て・源および信号線の配線

(3) 動作確認

図 3.1.9 ドローンの部品

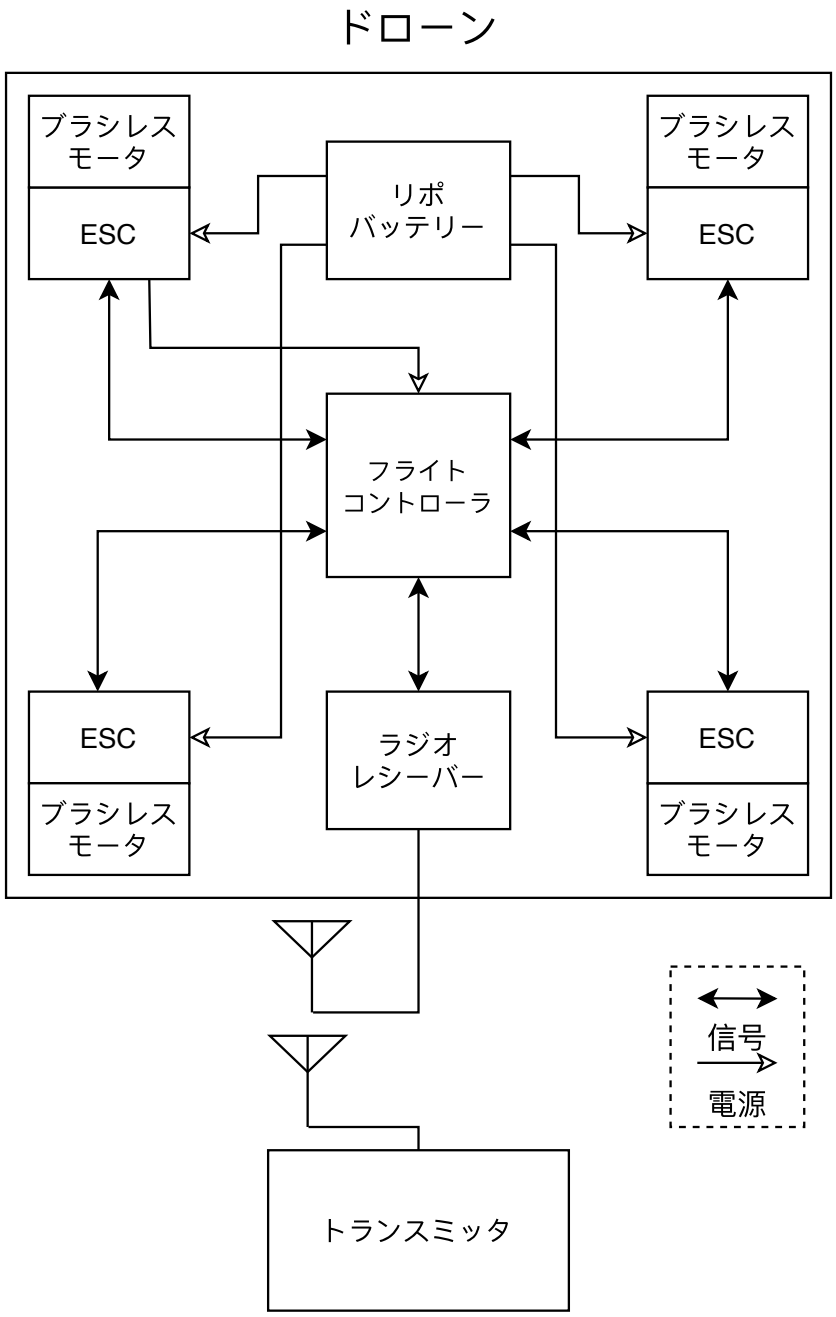

図 3.1.10 ドローンのブロック図

3.2　バイノーラルマイクの製作

3.2.1　マイクについて

収音には他の研究で行った, 音像定位実験のために製作したバイノーラルマイクを流用した.

使用した素子は秋月電子のエレクトレットコンデンサマイクロホン（SPL (Hong Kong) Limited, XCM6035）である. 左右用にそれぞれエレクトレットコンデンサマイクはロボットケーブルとはんだ付けし, ケーブルの端をステレオミニプラグと接続して製作した.

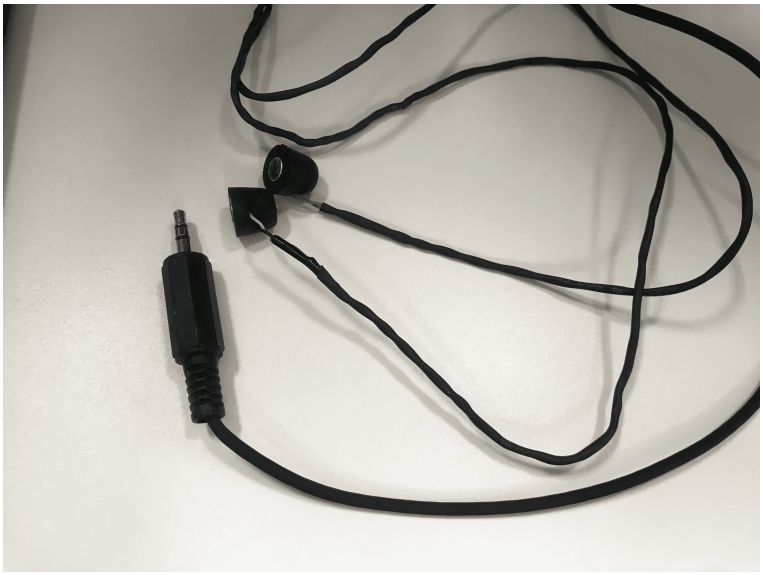

図 3.2.1 バイノーラルマイク

3.2.2　使用機器

(1) エレクトレットコンデンサマイク XCM6035 株式会社秋月電子通商

(2) シールドスリムロボットケーブル KRT-SW 株式会社秋月電子通商

(3) 3.5mm ステレオミニプラグ MP-319 株式会社秋月電子通商

3.3 Raspberry Piについて

Raspberry Pi は英国のラズベリーパイ財団によって開発されている, ARM プロセッサを搭載したシングルボードコンピュータである. Raspberr Pi は教育用として製作されたが, 現在では IoT 製品開発などの業務や人工衛星の OBC (On Board Computer) にも使用されている.

Raspberry Pi は Linux 系の OS で動作するためソフトウェア開発に強みをもち, GPIO ピンを通して SPI, I2C, I2S などの通信を行えるため, センサなどを用いた開発を容易に行える. また, USB 端子を搭載し, Wi-Fi, Bluetooth 接続も可能で, プロタイプ開発に適したデバイスである.

3.3.1 OSの選定

Raspberry Pi で使用可能な OS には

- 電子工作などに適した公式 OS Raspbian
- Linux ディストリビューションの Ubuntu から派生した Ubuntu MATE
- Microsoft Windows 10

などが存在する.

本研究では主に GPIO を使用して開発を行うため, Raspbian を使用した. なお, OS のバージョンは 10.1 Buster Lite である. また, カーネルのバージョンは 4.19.75-v7 である.

3.3.2 初期設定について

Raspberry Pi を実験で使用する準備として, 以下の手順で初期設定を行った.

(1) OS のインストール

(2) 地域, 言語の設定
3.3.2-1 を実行し, Localization Options を選択して, 地域を日本, 言語を英語に設定した.

ソースコード 3.3.2-1 Raspberry Pi のコンフィグレーションツールの起動コマンド

```
1  sudo raspi-config
```

(3) ssh の設定
3.3.2-1 を実行し, Interfacing Options を選択して, SSH を有効に設定した.

(4) ネットワーク・プロキシに関する設定
`/etc/wpa_supplicant/wpa_supplicant.conf`,
`/etc/apt/apt.conf`, `/etc/dhcpcd.conf`, `~/.bashrc`,
`~/.curlrc`, `~/.wgetrc` を編集し, ネットワーク・プロキシに関する設定を行った.

(5) アップデート
3.3.2-2 を実行し, Raspberry Pi を最新の状態に更新した.

ソースコード 3.3.2-2 Raspberry Pi のアップデートコマンド

```
1  sudo apt update
2  sudo apt upgrade -y
3  sudo apt dist-upgrade
4  sudo rpi-update
5  sudo reboot
```

3.3.3 AD 変換用の拡張ボードについて

Raspberry Pi は ADC（AD コンバータ）を搭載していないため, マイクからの入力信号を扱うには AD コンバータを導入する必要がある.

本研究で用いたのはマルツエレック株式会社の Pumpkin Pi である. PumpkinPi は計測用とオーディオ用のデュアル A-D コンバータを搭載しており, Raspberry Pi にオーディオ入力, アナログ入力機能を加えることが可能となる.

Pumpkin Pi の仕様を以下に示す.

- 対応 OS Raspbian
- 対応機種 Raspberry Pi Model B+/Raspberry Pi 2 Model B/Raspberry Pi 3 Model B

- LED 出力 1 点
- 赤外線リモコン機能 送受信
- オーディオコネクタ ϕ 3.5mm ステレオミニジャック
- オーディオ入力 量子化ビット数=24, サンプリング周波数=48/96kHz
- 計測用 AD 変換 2 チャンネル,16 ビット
- 本体寸法 65(W) × 56(D)mm
- 本体重量 約 25g

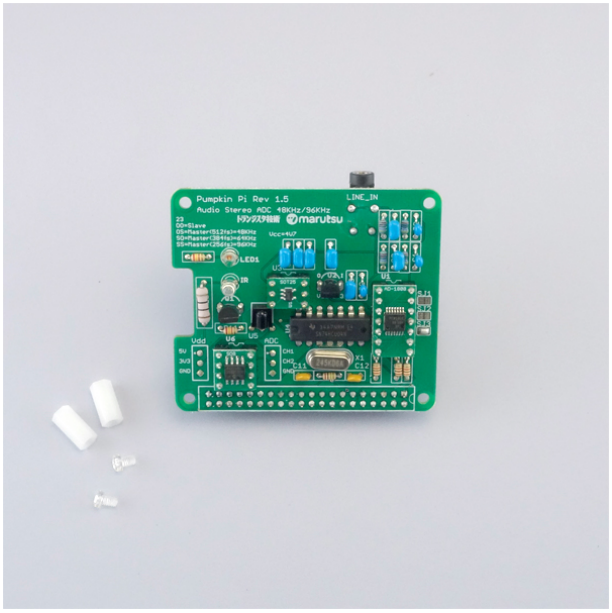

図 3.3.1 PumpkinPi

3.3.4 初期設定

PumpkinPi の初期設定はトランジスタ技術トランジスタ技術 2017 年 1 月号 [4] にしたがって行った. 以下に簡易的な手順を示す.

(1) Pumpkin Pi を使用するための Raspberry Pi 固有の設定

まず適当な作業ディレクトリで以下のコマンドを実行する.

ソースコード 3.3.4-3 PumpkinPi の設定ファイルダウンロードコマンド

```
1 wget http://einstlab.web.fc2.com/RaspberryPi/PumpkinPi.tar
```

```
2 tar xvf PumpkinPi.tar
3 cd PumpkinPi
4 ./PumpkinPi.sh
```

(2) カーネルとデバイス・ドライバのバージョンの確認

カーネルのバージョンとデバイス・ドライバのバージョンは同じである必要がある. カーネルのバージョンは uname -r で, デバイス・ドライバのバージョンは modinfo snd_soc_pcm1808_adc.ko でそれぞれ確認できる.

(3) AD コンバータ用のデバイス・ドライバのインストール

次の 2 つのデバイス・ドライバをインストールする.

1. pcm1808-adc.ko
   PCM1808 固有の動作を決定するドライバ.
2. snd_soc_pcm1808_adc.ko
   Raspberry Pi のサウンドとして属性を決定するドライバ

まず, ホームディレクトリに PumpkinPi.tar をダウンロードして展開する.

ソースコード 3.3.4-4 デバイス・ドライバダウンロードコマンド

```
1 cd
2 wget http://einstlab.web.fc2.com/RaspberryPi/PumpkinPi.tar
3 tar xvf PumpkinPi.tar
4 cd PumpkinPi/Driver
```

次にデバイス・ドライバをインストールする.

ソースコード 3.3.4-5 デバイス・ドライバのインストールコマンド

```
1 sudo cp Backup/pcm1808-adc.bak/ /lib/modules/‘uname -r‘/
      kernel/sound/soc/codecs/pcm1808-adc.ko
2 sudo cp Backup/snd_soc_pcm1808_adc.bak /lib/modules/‘uname -r
      ‘/kernel/sound/soc/bcm/snd_soc_pcm1808_adc.ko
3 sudo depmod -a # 依存関係を調整
```

OS のカーネル 4.4 以降ではデバイス・ツリー構造を導入してあるため, デバイス・ツリー情報ファイルをコピーする.

ソースコード 3.3.4-6 デバイス・ツリー情報ファイルのコピーコマンド

```
1 sudo cp pcm1808-adc.dtbo /boot/oberlays/
```

最後にデバイス・ドライバが電源起動時に自動的に読み込まれるように/boot/config.txt に dtoverlay=pcm1808-ad を追加する.

以上の作業を完了した後, 再起動することで設定が適用される.

第4章　ソフトウェアの製作

4.1　使用するソフトウェア・プログラムについて

本研究ではシミュレーションとデータ処理のためのプログラムを Python と Go 言語を使用して製作した.

4.1.1　Python について

Python は, 汎用のプログラミング言語である. コードがシンプルで扱いやすく設計されており, C 言語などに比べて, さまざまなプログラムを分かりやすく, 少ないコード行数で書けるといった優れた特徴がある.

標準ライブラリやサードパーティ製のライブラリなど, さまざまな領域に特化した豊富で大規模なツール群が用意され, 自らの使用目的に応じて機能を拡張していくことができる.

4.1.2　Go 言語について

Google で開発された, 汎用のプログラミング言語である. Go 言語は, 静的型付け, C 言語の伝統に則ったコンパイル言語, メモリ安全性, ガベージコレクション, 並行性などの特徴を持つ. また, 軽量スレッディングのための機能, Python のような動的型付け言語のようなプログラミングの容易性, などの特徴もある.

また, Go 言語は Python のようにシンプルな記法を有し, 制作者によらず同じコードになりやすく学習しやすいため, 研究用途に適した言語である.

4.1.3　プログラミング言語の使い分けについて

Python は 4.1.1 節で述べた特徴から, 様々な試行をしながら行う開発に適している. 一方で, 実行速度が遅く, リアルタイム処理には向かない. したがって, 本研究ではこの特性を活かし, 予備実験・データ処理・グラフ作成に Python を使用した.

また, Go 言語は言語仕様が現代的で, 実行速度が速く, クロスコンパイルが可能といった特徴から, ADF ライブラリの作成および実装に使用した.

4.2 音声ファイル処理およびグラフ作成プログラムの制作

本節では後述の実験結果の処理に使用したプログラムの制作について述べる.

音声ファイルの処理には行列演算が必要である. この処理には数値計算を効率的に行うための拡張モジュールである NumPy を使用した. また, 波形のプロットには NumPy を基盤にしたグラフ描画ライブラリ matplotlib を使用した. 音声ファイルの入出力には標準ライブラリである wave や外部ライブラリの PySoundFile を使用した.

4.2.1 Python で制作したプログラム

Python で制作したプログラムを表 4.2.1 に示す。

表 4.2.1 Python で制作したプログラム

種類	用途	付録番号
モジュール	波形プロット簡易化	1.1.1-1
	音声ファイル入出力	1.1.1-2
	関数の実行時間計測・情報表示	1.1.1-3
ツール	音声ファイル畳み込み	1.1.2-4
	ステレオ音声ファイルを LR モノラル音声に分割	1.1.2-5
	2 つの wav ファイル間の伝達関数を計算	1.1.2-6
	csv ファイルの各列のデータをそれぞれ wav ファイルに変換	1.1.2-7
	目的の音声ファイルからノイズの音声ファイルに含まれる成分を取り除く	1.1.2-8
	指定秒数分の白色雑音をサンプリング周波数 48kHz で生成	1.1.2-9
	csv ファイルから mp4 または gif 画像を生成	1.1.2-10
	csv ファイルから波形をプロット	1.1.2-11
	複数の wav ファイルから一枚の図に波形をプロット	1.1.2-12
	スペクトログラムを表示	1.1.2-13

4.2.2 Go 言語で制作したプログラム

Go 言語で制作したプログラムを表 4.2.2 に示す。

表 4.2.2 Go 言語で制作したプログラム

種類	用途	付録番号
ライブラリ	型変換処理	1.2.1-2
	計算処理	1.2.1-3
	ファイル入出力処理	1.2.1-4
ツール	指定した SN 比で音声を合成	1.2.1-5
	2 つの wav ファイルを畳み込み	1.2.1-6
	2 つの wav ファイルを畳み込み （フーリエ変換を用いた高速版）	1.2.1-7
	2 つの wav ファイルもしくは csv ファイルを畳み込み （フーリエ変換を用いた高速版）	1.2.1-8
	csv ファイルのデータから wav ファイルを生成する	1.2.1-9
	第 6 章の実験で用いる ADF の設定を記述した JSON ファイルを生成	1.2.1-10
	csv ファイルのデータから指定したサンプルの 平均二乗誤差（MSE）を計算し, csv ファイルに新たな列として追記	1.2.1-11
	PortAudio を使用した多チャンネル収音用	1.2.1-12
	DSB ファイルから wav ファイルに変換	1.2.1-13

4.3 ADF ライブラリの制作

4.3.1 ライブラリの設計

Python で記述された適応信号処理のライブラリとして Padasip[5] が公開されている. Padasip は信号のフィルタリング, 予測, 復元, 分類といった適応信号処理を簡易化するために設計されたライブラリであり, LMS・NLMS, AP, RLS をはじめとする主要なアルゴリズムが一通り実装されている.

本研究で制作した ADF ライブラリは Padasip を参考に設計し, Go 言語で実装した.

ライブラリのコードを付録 A.2.2 に示す.

ライブラリの各関数にはユニットテストが書かれており, GitHub Actions を利用して自動テストが行われるため, 一定の質が担保されている.

また, ライセンスは MIT を採用しているため, 使用, 再配布, 商用利用などが許可されている.

4.3.2 インストール方法

製作したライブラリは GitHub[6] で公開したため, 次のコマンドを実行することでインストールすることができる.

```
go get github.com/tetsuzawa/go-adflib/adf
```

4.3.3 使用方法

ライブラリの使用方法は Padasip と同様である。まず、任意のフィルタのインスタンスを生成し、メソッドを実行することでフィルタリングを行える。

詳しい使い方は GoDoc（Go のパッケージリファレンス）[7] を参照されたい.

第5章　駆動音低減のための予備実験

5.1　ドローンの駆動音のサンプル収音

5.1.1　実験目的

本研究で用いるLMS・AP・RLSアルゴリズムは参照信号が有色性を持つと収束性能が低下する特徴を持つ. したがって, リアルタイム処理が目的の場合, 参照信号の周波数特性が処理効果に影響を及ぼすことになる. したがって, ドローンの駆動音の周波数特性を調べるため, サンプル収音を行った.

5.1.2　実験方法

(1)　収音のためのハードウェア構成について

収音はドローンとバイノーラルマイクを使用し, 図5.1.1のブロック図の構成で行った.

ただし, 前述のバイノーラルマイクのみでは十分な入力電圧が得られないため, 収音にはマイクロフォンアンプを使用する必要がある.

なお, 本研究では室内でサンプル収音を行ったため, 据え置き型のマイクロフォンアンプ（オーディオテクニカ, AT-MA2）を使用したが, 実際にドローンに搭載する際は別途小型のマイクロフォンアンプが必要と推察される.

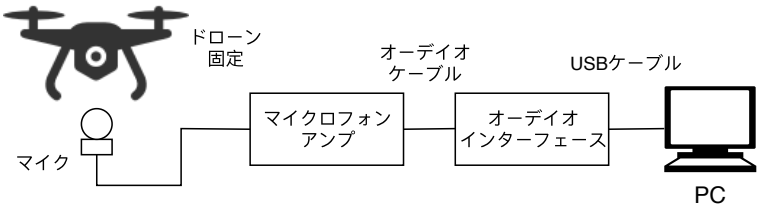

図 5.1.1 ドローンの駆動音サンプル収音のブロック図

(2)　収音に使用するソフトウェアについて

音声の収録には Go 言語で記述した自作の収音用コマンドラインツール（付録 1.2.1-12）を使用した. このプログラムは SoX（コマンドラインベースの音声編集ソフトウェア)[8] の `rec` コマンドを参考に設計した. オーディオ入出力 API の PortAudio[9] を利用しており, 多チャンネルでの収音に対応している他, 独自の機能として音声データを本研究で主に使用されている形式（.DSB - 符号付き整数型 16bit バイナリファイル）で保存することが可能となっている.

本実験では自作のソフトウェアを使用したが, .wav などの一般的な音声データ形式を使用する場合, 前述の SoX を利用しても同様に収音が可能である. また, 同じく PortAudio をベースとした GUI の音声編集ソフトウェアである Audacity[10] も有力な候補となる. いずれのソフトウェアも無料で使用可能である.

5.1.3　使用機器

使用機器を以下に示す.

(1) ドローン
　3.1 節を参照.

(2) マイク
　3.2 節のバイノーラルマイクを 1ch のみ使用した. 詳細は 3.2 節を参照.

(3) マイクロフォンアンプ AT-MA2 株式会社オーディオテクニカ

(4) オーディオインターフェース ローランド株式会社 DUO-CAPTURE EX S/N.Z6C6056

5.1.4　実験結果

収録したドローンの駆動音の波形を付録 1.1.2-12 のプログラムを使用し, 描画した図を図 5.1.2 に示す. また, 付録 1.1.2-13 のプログラムを使用し, 周波数特性を解析した. 駆動音のスペクトログラムを図 5.1.3 に示す.

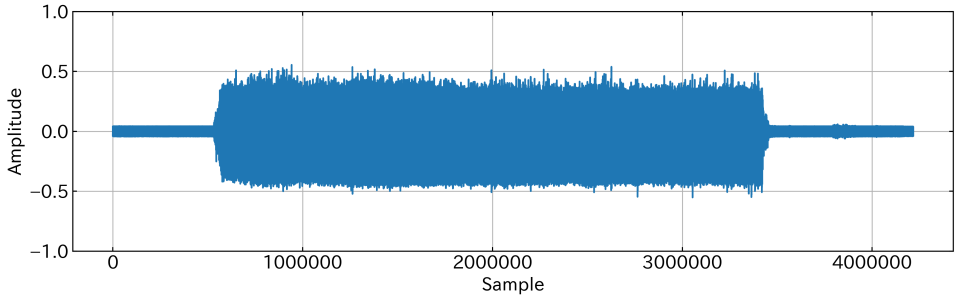

図 5.1.2 ドローンの駆動音の波形観測結果

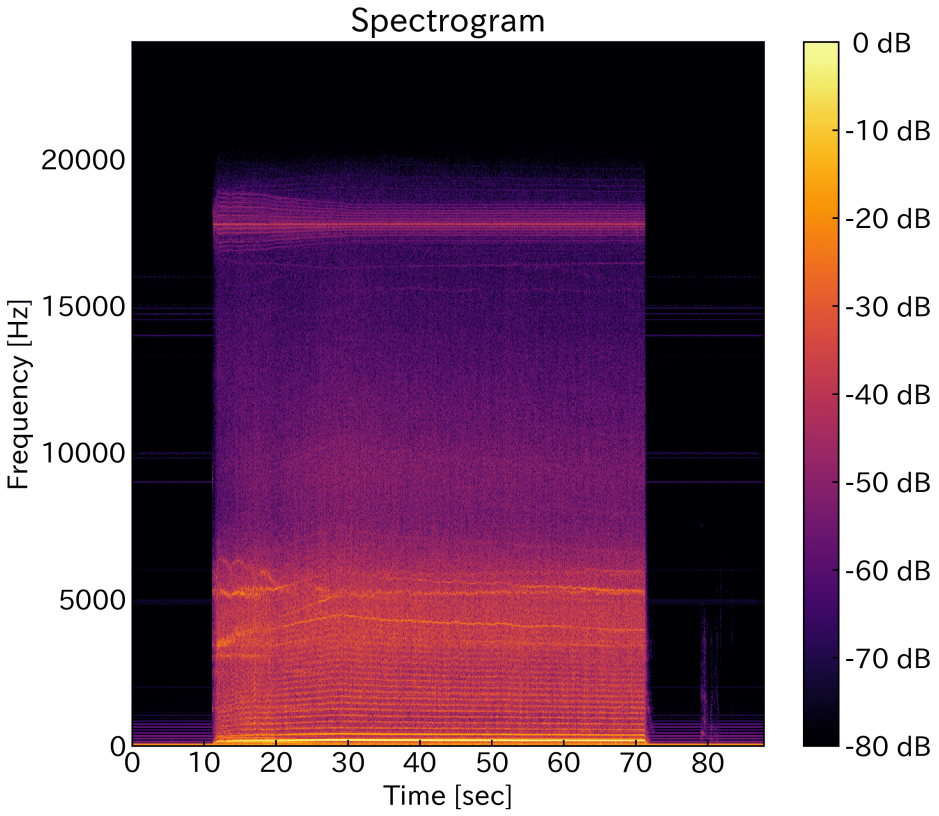

図 5.1.3 ドローンの駆動音のスペクトログラム観測結果

図 5.1.2, 図 5.1.3 における 0~10[sec] および 70[sec] 以降の区間はドローンを停止させた無音区間となっている. しかしながら, 無音区間に何らかの定常な雑音が存在す

るためドローンの駆動音のみを解析するためには, この定常雑音を取り除く必要がある.

そこで, 付録 1.1.2-8 のプログラムを使用し, スペクトラルサブトラクション法 [11] により, 無音区間の周波数成分の平均を全体から取り除いた. スペクトラルサブトラクション法適用後の波形を図 5.1.4 に, スペクトログラムを図 5.1.5 に示す.

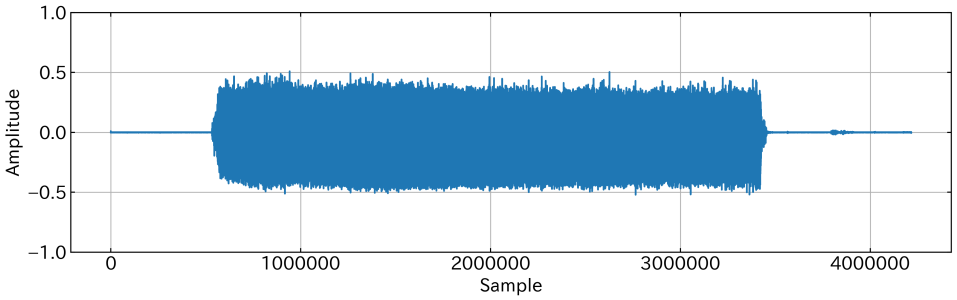

図 5.1.4 ドローンの駆動音の波形観測結果（雑音処理後）

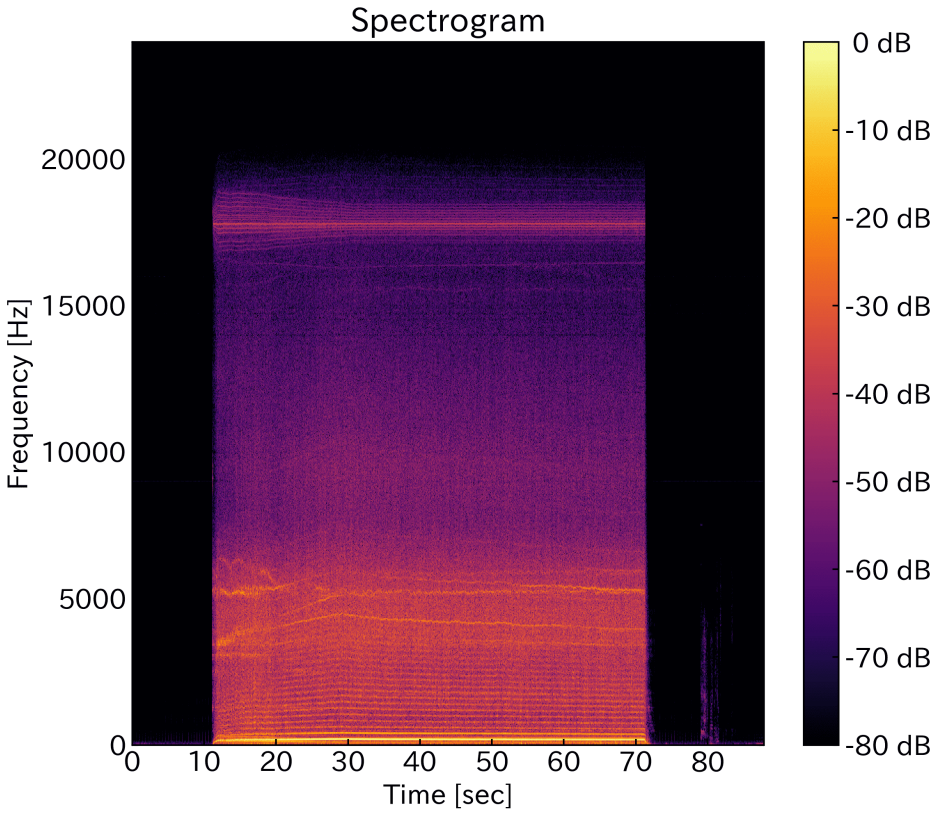

図 5.1.5 ドローンの駆動音のスペクトログラム観測結果（雑音処理後）

5.1.5 考察

図 5.1.5 より, 駆動音は 100Hz 付近の低域成分とその倍音成分, 5kHz 付近の中域成分, 18kHz 付近の高域成分を合わせた周波数特性を示すことがわかる. このうち, 低域と中域の成分はドローンの羽音およびモータの回転音によるもの, また, 高域成分はブラシレスモータの回転数を制御するインバータ回路によるものと思われる.

上記より, ドローンの駆動音を有色性の周波数特性を持つことが確認できた.

5.2　信号の有色性に対する ADF の性能への影響

5.2.1　実験目的

本節では ADF に対する参照信号を 5.1 節で収音したドローンの駆動音とした場合と白色雑音とした場合の収束に要するサンプル数（収束速度）, 収束した後の精度（推定精度）を比較する. これにより, 各適応アルゴリズムの信号の有色性に対する収束性能への影響を評価することを目的とする.

5.2.2　実験方法

実験方法を以下に示す.

(1) 5.1 節で収音した雑音処理完了後の音声（以下, 駆動音とする）に対して, ADF の各アルゴリズムにおける最適なステップサイズを求める.

(2) (1) で求めた ADF に対して, 駆動音を参照信号, 駆動音に白色雑音を加えたものを目的信号として十分な時間フィルタリングを実行

(3) 同様に, 白色雑音を参照信号, 白色雑音にの別の白色雑音を加えたものを目的信号として十分な時間フィルタリングを実行

(4) フィルタ誤差を MSE で平滑化したものをグラフ化し, 各アルゴリズムで比較・評価

ただし, 本実験における ADF のフィルタ長は 4 サンプル, 付与雑音は参照信号に対して-30dB の振幅, MSE のタップ数は 512 サンプルとした.

5.2.3　実験結果

信号に駆動音を使用した場合の各アルゴリズムの収束特性を図 5.2.1 に示す. 同様に, 白色雑音を使用した場合の各アルゴリズムの収束特性を図 5.2.2 に示す.

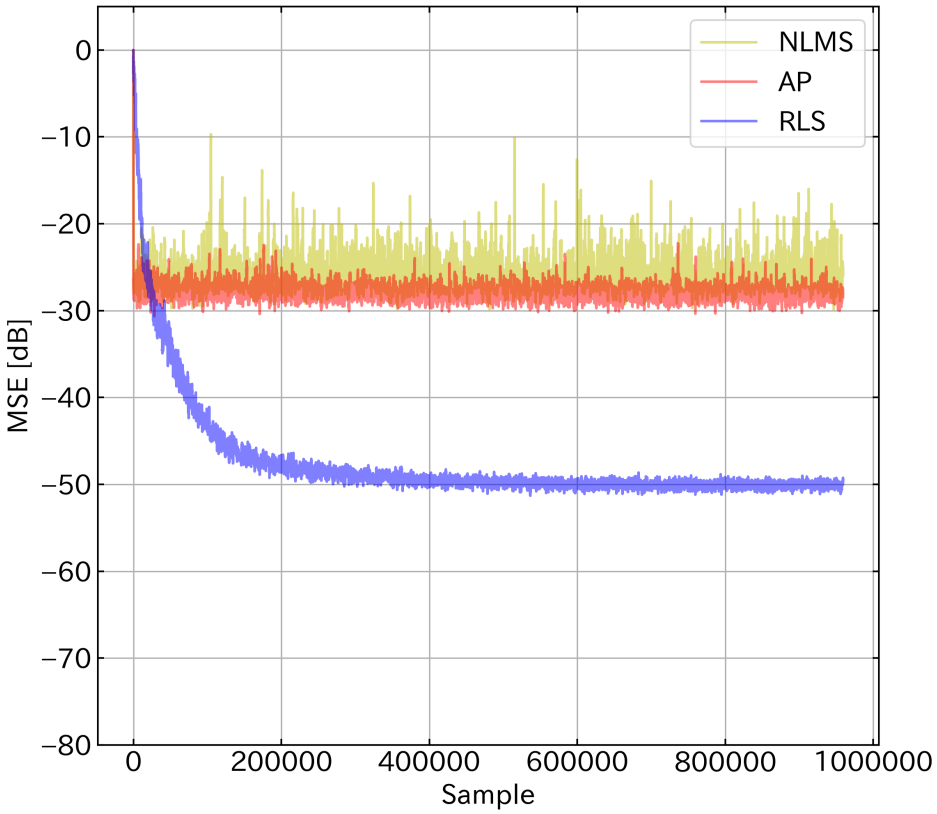

図 5.2.1 信号に駆動音を使用した場合の各アルゴリズムの収束特性測定結果

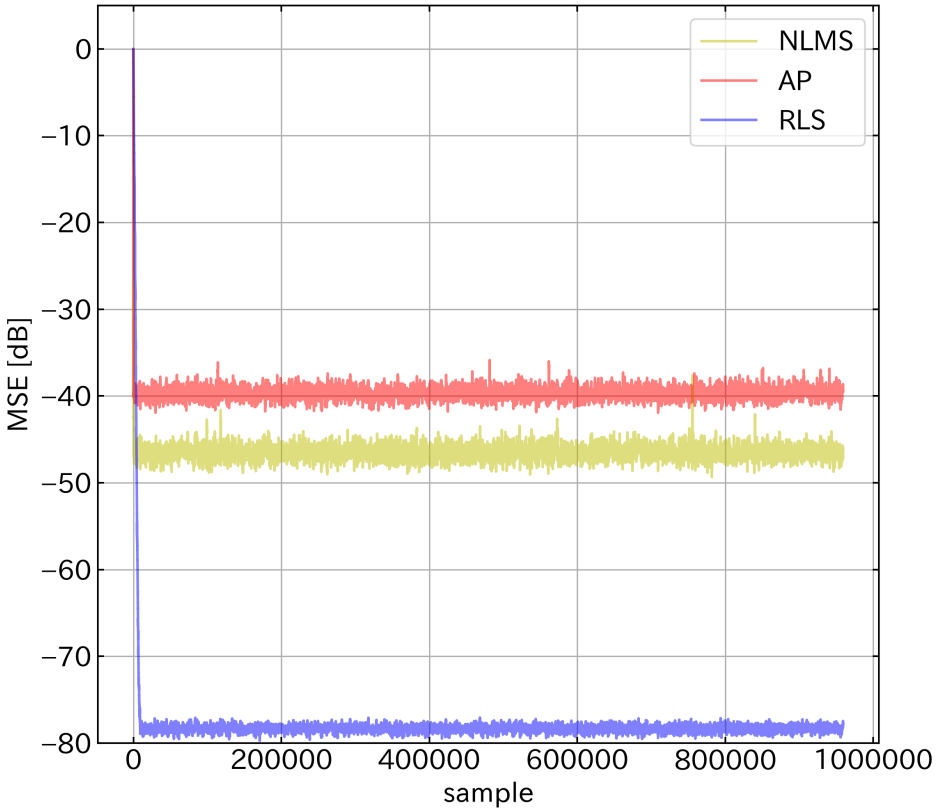

図 5.2.2 信号に白色雑音を使用した場合の各アルゴリズムの収束特性測定結果

5.2.4 考察

図 5.2.1, 図 5.2.2 より, 信号に駆動音を使用した場合, 理論通り収束性能が低下していることがわかる. 特に RLS アルゴリズムの場合, 収束時の MSE が約 80dB から約 50dB となり, 収束するまでに必要なサンプル数も約 300,000 サンプル程度まで増加していることがわかる. サンプリング周波数は 48kHz であるので収束までに約 6 秒かかることになる. 一方, NLMS アルゴリズムと AP アルゴリズムは収束時の MSE は 10dB ほど悪化しているが, RLS アルゴリズムほど大きい性能低下は見られなかった.

以上より, リアルタイム処理を目的とする場合 NLMS・AP アルゴリズムの使用が適していると思われる.

5.3 ADF ライブラリのベンチマーク

5.3.1 実験目的

ADF の性能を評価する尺度の 1 つとして, 収束速度, 推定精度の他に演算時間（演算量）が存在する. 特にリアルタイムでフィルタ処理を行う場合, プログラムの実行速度は雑音低減の性能に大きな影響を及ぼす.

本節では, Go 言語で自作した ADF ライブラリのベンチマークを行い, 各アルゴリズムの実行に要する演算時間を評価することを目的とする.

5.3.2 実験方法

本実験では Go 言語に標準で実装されているベンチマークの仕組みを利用して行った.

Go 言語ではテストファイルに `func BenchmarkXxx(b *testing.B)` のように関数を宣言することで容易にベンチマークを実装することができる. また, `for i := 0; i < b.N; i++`のように for 文を記述することで, 自動的に 1 秒あたりの実行回数を測定・出力することが可能である. なお, ベンチマークはテストファイルと同じディレクトリで `go test -bench . -benchmem` を実行することで行える.

ベンチマークはそれぞれ信号として白色雑音を使用し, フィルタ長は 8 タップとした.

実行環境は Raspberry Pi 3 Model B である. また, 比較対象として MacBook Pro (13-inch, 2019) 上においてもベンチマークを実行した.

5.3.3 実験結果

1 サンプルあたりのフィルタ係数更新・次サンプルの予測にかかる演算時間とメモリ使用量の測定結果を表 5.3.1, 表 5.3.2 に示す.

表 5.3.1 ADF ライブラリのベンチマーク測定結果（Raspberry Pi）

アルゴリズム	実行時間 [ms/op]	メモリ使用量 [B/op]
NLMS	1.9	84
AP	78.4	941
RLS	52.9	2549

表 5.3.2 ADF ライブラリのベンチマーク測定結果（MacBook Pro）

アルゴリズム	実行時間 [ms/op]	メモリ使用量 [B/op]
NLMS	0.13	96
AP	5.40	1248
RLS	3.25	2920

5.3.4 考察

(1) メモリ使用量について

表 5.3.1, 表 5.3.2 より, 各アルゴリズムのメモリ使用量は NLMS アルゴリズムが一番少なく, AP アルゴリズム, RLS アルゴリズムと多くなっている. これは理論式に即した物となっており, 妥当な値である.

Go 言語にはガーベッジコレクションが実装されており, メモリセーフである. また, Raspberry Pi 3 Model B のメモリは 1GB と潤沢であるため, 使用メモリに関しては実用範囲内であると思われる.

(2) 演算時間について

表 5.3.1, 表 5.3.2 より, NLMS アルゴリズムの演算時間は AP・RLS に比べて数十倍高速であることがわかる. 一方, Raspberry Pi の演算時間は MacBook Pro に比べるて十倍程度長いことがわかる.

ここで, これらのアルゴリズムをアクティブノイズコントロールに適用することを考える. 音速が 340m/s、参照信号用のマイクと目的信号用のマイクの距離が 30cm であることを仮定すると, 音波が 2 つのマイク間を伝播するために要する時間は

$$\frac{0.3}{340} \approx 0.88[\mathrm{ms}] \tag{5.3.1}$$

である. したがって, ADF のフィルタ実行時間以外の遅延を無視した理想条件で最低でも 0.88ms/op 以上の演算速度が必要なことになる. また, これはフィルタ長が約 43 タップに相当する距離である. 実験で用いたフィルタ長は 8 タップであるため, フィルタ長が不十分で, システムの同定が不可能であることを示す. したがって, 本実験の条件でのアクティブノイズコントロールの実装は, 制作したライブラリを使用する場合には現実的でないことが結論づけられる. なお, 実現に向けての改善策としては, プログラムの実行速度を上げる, 計算機の性能を高くする, 許容範囲内でサンプリング周波数を低くするなどが挙げられる.

また, AP アルゴリズムの演算時間が RLS アルゴリズムに比べて長いことが表よりわかる. 理論上の演算量は RLS アルゴリズムが上回っているため, アルゴリズムの実装を見直す必要があると考えられる. これについては今後の課題としたい.

第6章　適応フィルタを用いた駆動音低減法の検討

6.1　実験概要

本実験では, 実際に入力される信号を模擬し, ADF による駆動音低減の効果を検討する. 信号の模擬のためには音波の伝達関数を求め、駆動音・音声信号と畳み込み、複数の SN 比で合成する。これらの信号に対して、複数のフィルタ長で ADF を実行することで SN 比・フィルタ長を変化させたときのフィルタリング性能の比較が可能となる。これにより, 予備実験と合わせて雑音低減の実現可能性を総合的に評価する.

6.2　実験方法

本実験は以下の手順で行った.

(1)　ドローンのマイクの取り付け

図 6.2.1 のように簡易的にマイクを木材と粘着テープで固定し, 取り付けた.

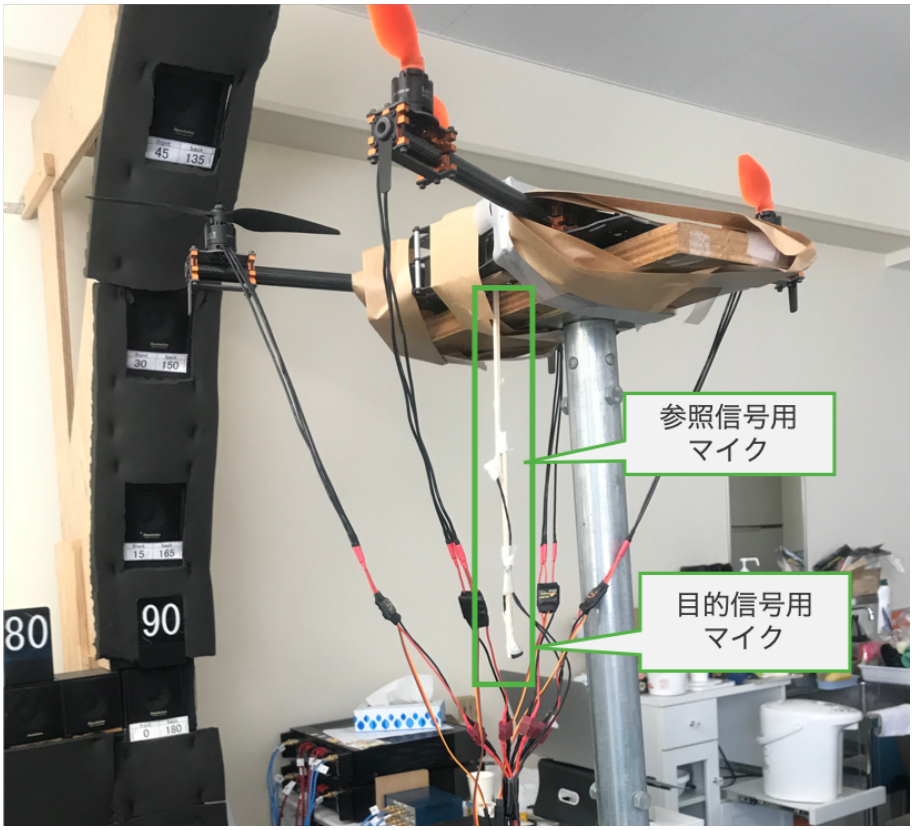

図 6.2.1 ドローンと各マイクの取り付け

(2)　音声信号の音源から 2 つのマイクまでの伝達関数の測定

目的音源からマイクまでの伝達関数を測定した. これにより, 任意の音声信号を畳み込むことで, 様々な条件での雑音低減のシミュレーションを行うことができる.

伝達関数の測定は設備の都合上室内で行った. 実際にドローンを使用するのは屋外を想定しているため, 残響を無効化するために 1000 サンプル以上の伝達関数を切り捨て, 半自由音場を模擬した.

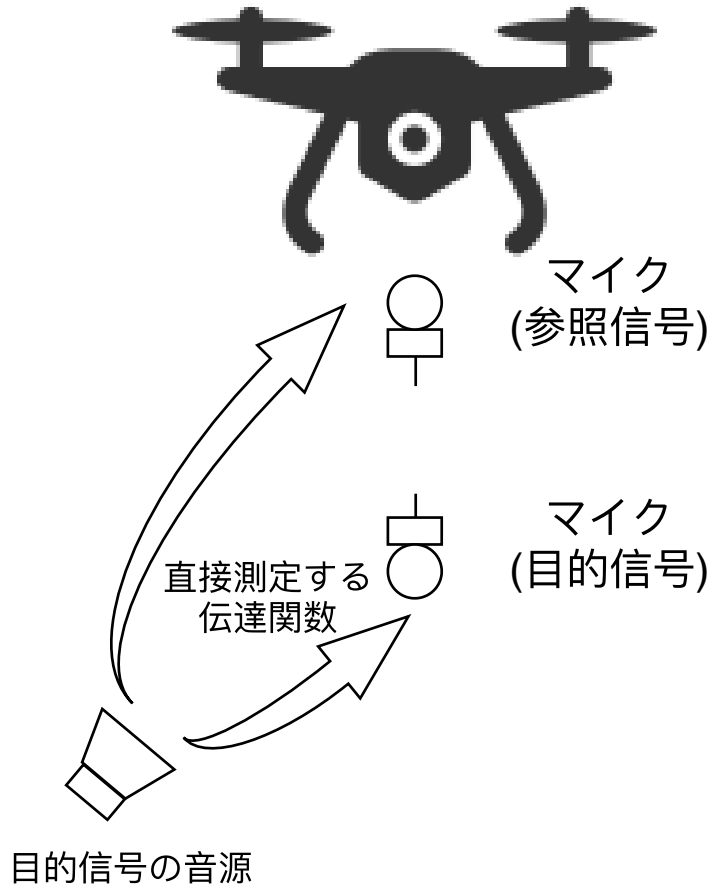

図 6.2.2 音声信号の音源から 2 つのマイクまでの伝達関数の測定

(3) ドローンの駆動音に対する 2 つのマイクの擬似的な伝達関数の計算

2 つのマイク間の疑似伝達関数を計算するためにはそれぞれのマイクで駆動音を収音する必要がある. 収音は 5.1 節の構成を 2ch に拡張して行った.

疑似伝達関数は 2 つの駆動音をフーリエ変換し, 除算したものを逆フーリエ変換することで得られる. この処理は付録 1.1.2-6 のプログラムを実行して行った.

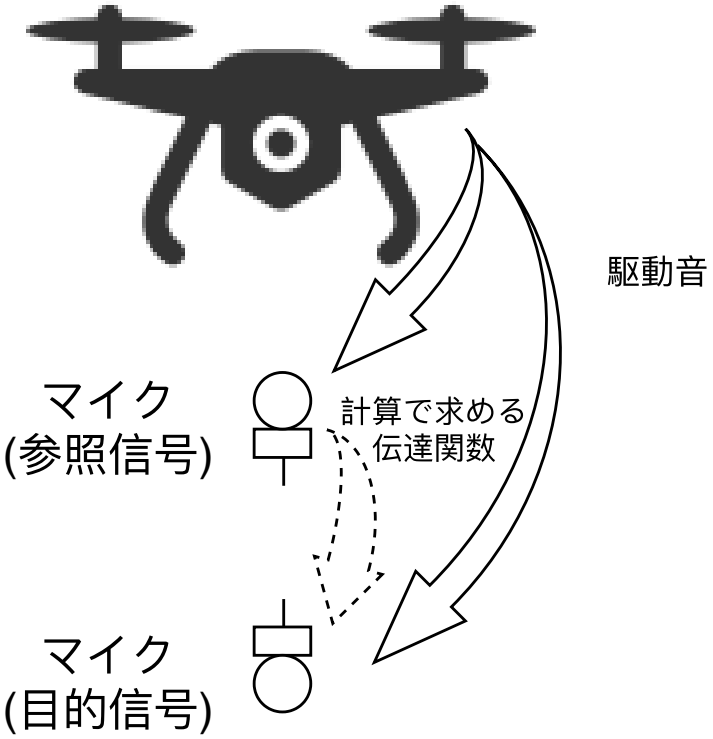

図 6.2.3 ドローンの駆動音に対する 2 つのマイクの擬似的な伝達関数の計算

(4) 駆動音と音声信号それぞれで参照信号と目的信号用の伝達関数を畳み込み

(2), (3) で各伝達関数を求めたので, 駆動音・音声信号をそれぞれ付録 1.2.1-8 のプログラムを実行して伝達関数と畳込み, 参照信号と目的信号を生成した.

(5) 指定した SN 比で駆動音と音声信号を合成

付録 1.2.1-5 のプログラムを実行し, -40dB から 0dB まで 5dB 刻みで駆動音と音声信号を合成した.

(6)　合成した参照信号と目的信号で ADF を実行

5.2 節と同様に合成した参照信号と目的信号に対して, ADF の各アルゴリズムにおける最適なステップサイズを求め, 十分な時間フィルタリングを実行する.

6.3 実験結果

SN 比を-40dB, -20dB, 0dB, フィルタ長を 4, 64, 256 とした場合の各アルゴリズムの収束特性を図 6.3.1 に示す.

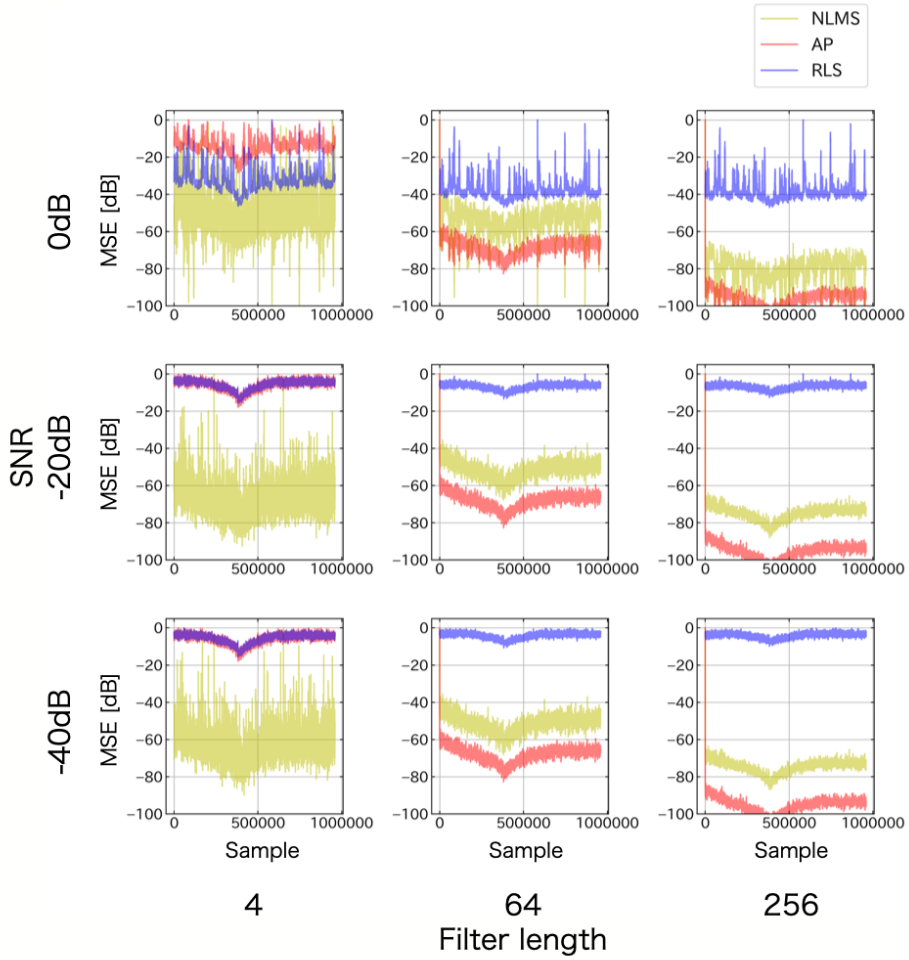

図 6.3.1 種々の SN 比とフィルタにおける各アルゴリズムの収束特性

6.4　考察

図 6.3.1 より, 音声が波形として見られるのは AP アルゴリズムの SN 比 0dB, フィルタ長 4 タップのものと SN 比 0dB の RLS アルゴリズムのみであった. 実際の運用では SN 比が 0dB よりも悪化することが予想されるため, 適切に駆動音を低減し, 音声を抽出することは難しいと思われる.

図 6.3.1 の波形に音声帯域以外の高周波が存在すると仮定し, 収束特性に対してカットオフ 8kHz, フィルタ長 512 タップの FIR ローパスフィルタを適用した. これにより得られた収束特性を図 6.4.1 に示す.

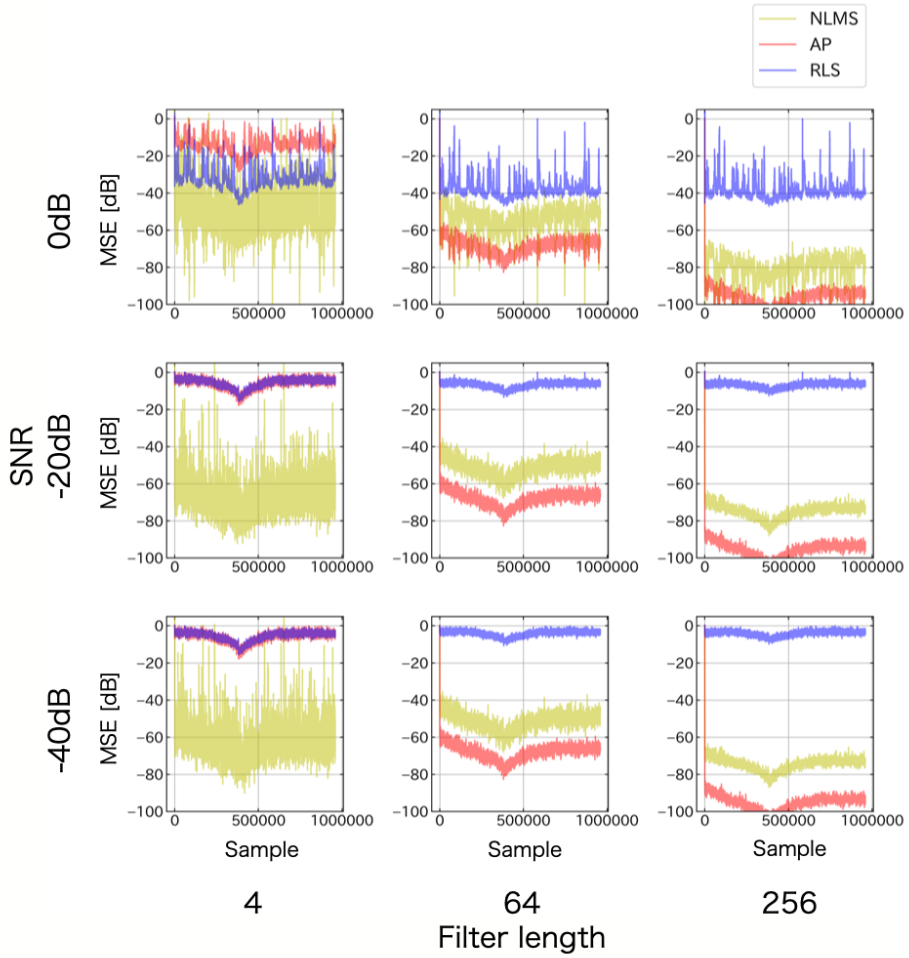

図 6.4.1 種々の SN 比とフィルタ長における各アルゴリズムの収束特性（8kHz 以下）

図 6.4.1 から分かる通り, 周波数帯域を 8kHz 以下に限定しても収束特性がほとん

ど変化しないことがわかる．したがって, 波形に現れたのは 8kHz 以下の雑音であると推測される．

NLMS フィルタが有効な結果を得られなかった理由について考察する．本実験で使用したマイクは無指向性である．また, マイクの配置は参照信号と目的信号の収音を模擬した簡易的なものであった．ここで, 6.2 節 (5) で合成した参照信号と目的信号を比較する．参照信号と目的信号を図 6.4.2 に示す．

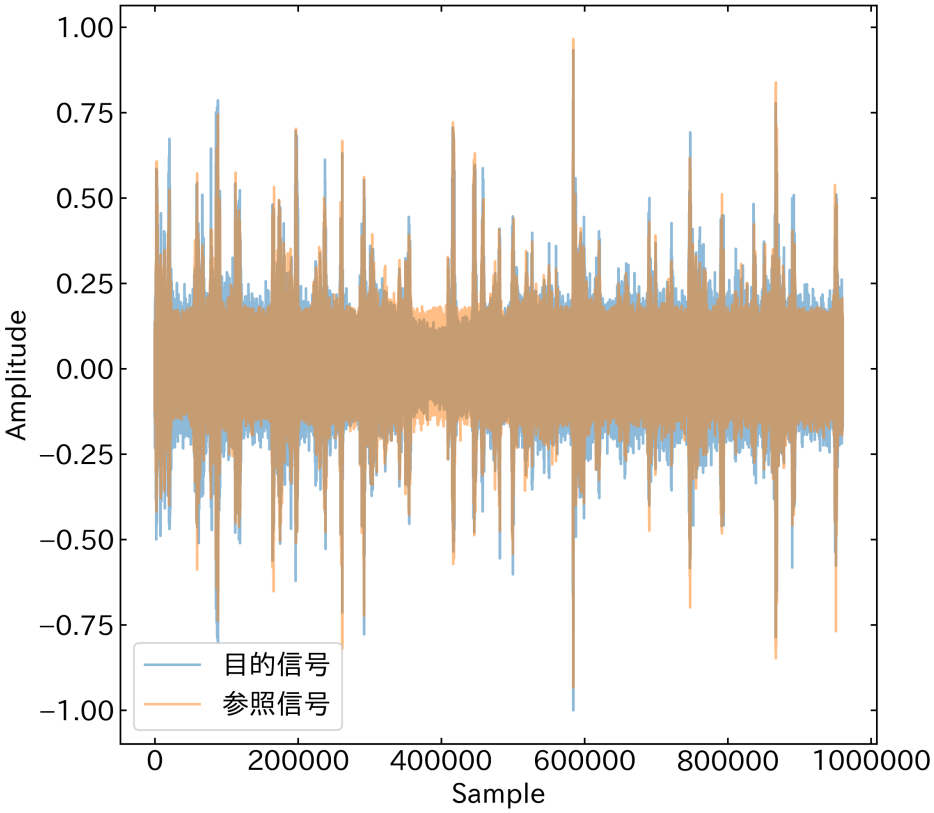

図 6.4.2 作成した参照信号と目的信号の比較

図 6.4.2 より, 参照信号と目的信号に大きな差が無いことがわかる．一般に ADF が十分なフィルタリング性能を発揮するためには, 参照信号に目的信号が入り込まないことが条件となる．前述の通り, 本実験では簡易的なシステムを使用したためこのような結果が得られたと推察される．したがって, 十分な指向性を有したマイクを使用し, 適切な配置で収音することが今後の課題となる．

図 6.3.1 より雑音低減の効果が最も優れた結果となったのは RLS アルゴリズムで有ることがわかる．しかしながら, 5.3 節の予備実験で確認したように, リアルタイム処理の観点から見ると RLS は実装に適していないため, 本研究で検討した構成での雑音低減の実現可能性は低いと思われる．

第7章　結論

本研究では, ドローンの応用分野の 1 つとして音声収録に着目し, 搭載されたマイクと小型の計算機を使用して収録した音声信号からドローンの駆動音を低減する手法について検討した.

駆動音の低減法を検討するにあたり, ドローン・バイノーラルマイクの組み立て, Raspberry Pi のセットアップなどハードウェアの準備を行った. Raspberry Pi には AD 変換が搭載されていないため, 拡張ボードとして PumpkinPi を使用し, 初期設定を行った.

ソフトウェアに関しては, まず Python を使用して, 静的な FIR フィルタ・適応アルゴリズムの評価を行った. 次に ADF のライブラリを Go 言語で自作し, 公開した. また, Python・Go 言語を使用して波形表示や音声編集用のソフトウェアを制作した.

次にドローンの駆動音に対する各適応アルゴリズムの有効性を検証するために, 駆動音のサンプル収音を行い, ADF の収束特性を試験した. 結果的に NLMS・AP アルゴリズムに比べて RLS アルゴリズムの収束誤差は 20dB 小さいが, 収束速度が遅く収束速度が 4167ms とリアルタイム処理には向かないことが判明した.

制作した ADF ライブラリのベンチマークを取ると, Raspberry Pi を計算媒体とした場合, 一番高速な NLMS アルゴリズムでも計算速度が遅く, アクティブノイズコントロールの実装は難しいことが判明した.

最後に, 実際に入力される音響信号を模擬し, ADF による雑音低減の効果を検討した. 音声の抽出に成功したのは SN 比 0dB の RLS アルゴリズムと一部のフィルタ長の AP アルゴリズムのみであった.

以上より本研究で検討した Raspberry Pi を有した構成での雑音低減の実現可能性は低いと思われる.

参考文献

[1] 浅野太. 音のアレイ信号処理. コロナ社, February 2011.

[2] A.H. Sayed. *Adaptibe Filters*. John Wiley & Sons, October 2011.

[3] Lynxmotion. Lynxmotionuav quadrinonano user-guidev1.1. `http://www.lynxmotion.com/images/document/PDF/LynxmotionUAV-QuadrinoNano-UserGuideV1.1.pdf`. (Accessed on 01/15/2020).

[4] CQ 出版. トランジスタ技術 2017 年 1 月号. `https://toragi.cqpub.co.jp/tabid/829/Default.aspx`. (Accessed on 01/16/2020).

[5] Matous Cejnek. Padasip - open source library for adaptive signal processing in language python. *Studentská tvůrčí činnost 2017*, Apr 2017. (Accessed on 01/17/2020).

[6] Tetsu Takizawa. tetsuzawa/go-adflib: go-adflib is designed to simplify adaptive signal processing tasks with golang. `https://github.com/tetsuzawa/go-adflib`. (Accessed on 01/23/2020).

[7] GoDoc. go-adflib - godoc. `https://godoc.org/github.com/tetsuzawa/go-adflib`. (Accessed on 01/17/2020).

[8] SoX. Sox - sound exchange — homepage. `http://sox.sourceforge.net/`. (Accessed on 01/19/2020).

[9] PortAudio. Portaudio - an open-source cross-platform audio api. `http://www.portaudio.com/`. (Accessed on 01/19/2020).

[10] Audacity. Audacity ® — free, open source, cross-platform audio software for multi-track recording and editing. `https://www.audacityteam.org/`. (Accessed on 01/19/2020).

[11] S.Boll. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, April 1979.

謝辞

本研究を進めるにあたり、御指導を頂いた苫小牧工業高等専門学校電気電子工学科准教授工藤彰洋博士に心より感謝します.

また本研究は、文部科学省・平成 29 年度宇宙航空科学技術推進委託費・宇宙航空人材育成プログラム、「超小型衛星開発を通した高専ネットワーク型宇宙人材育成」（研究代表者徳山工業高等専門学校 北村健太郎）の支援を受けて実施しました.

付 録A　付録

A.1　Pythonで制作したプログラム

A.1.1　モジュール

ソースコード 1.1.1-1 plot_tools.py

```
1   # coding: utf-8
2
3   import matplotlib.pyplot as plt
4   import numpy as np
5   from matplotlib.colors import Normalize
6
7   # diagram display settings
8   plt.rcParams['font.family'] = 'IPAPGothic'
9   plt.rcParams['font.size'] = 16
10  plt.rcParams['xtick.direction'] = 'in'
11  plt.rcParams['ytick.direction'] = 'in'
12  plt.rcParams['xtick.top'] = True
13  plt.rcParams['ytick.right'] = True
14  plt.rcParams['xtick.major.width'] = 1.0
15  plt.rcParams['ytick.major.width'] = 1.0
16  plt.rcParams['axes.linewidth'] = 1.0
17  plt.rcParams['figure.figsize'] = (8, 7)
18  plt.rcParams['figure.dpi'] = 100
19  plt.rcParams['figure.subplot.hspace'] = 0.3
20  plt.rcParams['figure.subplot.wspace'] = 0.3
21
22
23  class PlotTools(object):
24      """ PlotTools - plotting utility class for research
25      """
26
27      def __init__(self, y, fs=44100, fft_N=1024, stft_N=256, **
            kwargs):
28
29          if "start_pos" in kwargs:
30              self.start_sec = kwargs["start_sec"]
31          else:
```

```
32                 self.start_sec = 0
33
34             #
35             self.y = np.array(y) # data
36             self.fs = fs # Sampling frequency
37             self.dt = 1 / fs # Sampling interval
38             # Start position to analyse
39             self.start_pos = int(self.start_sec / self.dt)
40
41             self.fft_N = fft_N # FFT length
42             self.stft_N = stft_N # STFT length
43             self.freq_list = np.fft.fftfreq(fft_N, d=self.dt) # FFT
                    frequency list
44
45             if "window" in kwargs:
46                 window_name = kwargs["window"]
47             else:
48                 window_name = "hamming"
49
50             self.fft_window = define_window_function(name=
                   window_name, N=self.fft_N)
51             self.stft_window = define_window_function(name=
                   window_name, N=self.stft_N)
52
53             self.Y = np.fft.fft(self.fft_window * self.y)
54             self.samples = np.arange(self.start_pos, fft_N + self.
                   start_pos)
55             self.t = np.arange(self.start_sec, (fft_N + self.
                   start_pos) * self.dt, self.dt)
56
57             y_abs = np.abs(y)
58             # Complement 0 or less to mean to prevent from
                   divergence
59             self.y_abs = completion_0(y_abs) # Absolute value of y
60             self.y_gain = 20.0 * np.log10(y_abs) # Gain of y
61
62             # Spectrum
63             self.amp_spectrum = np.abs(self.Y) / fft_N * 2
64             self.amp_spectrum[0] = self.amp_spectrum[0] / 2
65             self.gain_spectrum = 20 * np.log10(self.amp_spectrum /
                   np.max(self.amp_spectrum))
66             self.phase_spectrum = np.rad2deg(np.angle(self.Y))
67             self.power_spectrum = self.amp_spectrum ** 2
68             self.power_gain_spectrum = 10 * np.log10(self.
                   power_spectrum / np.max(self.power_spectrum))
69
```

```
70              self.acf = np.real(np.fft.ifft(self.power_spectrum / np
                    .amax(self.power_spectrum)))
71              self.acf = self.acf / np.amax(self.acf)
72
73          def plot_y_time(self):
74              """plot_y_time - y vs time [sec]
75              """
76              fig = plt.figure()
77              ax1 = fig.add_subplot(111)
78              ax1.plot(self.t, self.y, "-", markersize=1)
79              ax1.axis([self.start_sec, (self.fft_N + self.start_pos)
                     *
80                        self.dt, np.amin(self.y) * (-1.1), np.amax(
                            self.y) * 1.1])
81              ax1.set_xlabel("Time [sec]")
82              ax1.set_ylabel("Amplitude")
83              ax1.set_title("Amplitude - Time")
84              plt.show()
85
86          def plot_y_sample(self):
87              """plot_y_sample - y vs sample number
88              """
89              fig = plt.figure()
90              ax1 = fig.add_subplot(111)
91              ax1.plot(self.samples, self.y, "-", markersize=1)
92              ax1.axis([self.start_pos, self.fft_N + self.start_pos,
93                        np.amin(self.y) * 1.1, np.amax(self.y) *
                            1.1])
94              ax1.set_xlabel("Sample")
95              ax1.set_ylabel("Amplitude")
96              ax1.set_title("Amplitude - Sample")
97              plt.show()
98
99          def plot_freq_analysis_log(self):
100             """ plot_freq_analysis_log - 1. gain vs frequency. 2.
                    phase vs frequency
101             """
102             fig = plt.figure()
103             ax1 = fig.add_subplot(211)
104             ax1.set_xscale('log')
105             ax1.axis([10, self.fs / 2, np.amin(self.gain_spectrum),
106                       np.amax(self.gain_spectrum) + 10])
107             ax1.plot(self.freq_list, self.gain_spectrum, '-',
                    markersize=1)
108             ax1.set_xlabel("Frequency [Hz]")
109             ax1.set_ylabel("Amplitude [dB]")
110             ax1.set_title("Amplitude spectrum")
```

```
111
112             ax2 = fig.add_subplot(212)
113             ax2.set_xscale('log')
114             ax2.axis([10, self.fs / 2, -180, 180])
115             ax2.set_yticks(np.linspace(-180, 180, 9))
116             ax2.plot(self.freq_list, self.phase_spectrum, '-',
                    markersize=1)
117             ax2.set_xlabel("Frequency [Hz]")
118             ax2.set_ylabel("Phase [deg]")
119             ax2.set_title("Phase spectrum")
120             plt.show()
121
122         def plot_freq_analysis(self):
123             """ plot_freq_analysis - 1. amplitude vs time [sec]. 2.
                     amplitude vs frequency
124             """
125             fig = plt.figure()
126             ax1 = fig.add_subplot(211)
127             ax1.plot(self.t, self.y, "-", markersize=1)
128             ax1.axis([self.start_sec, (self.fft_N + self.start_pos)
                     *
129                   self.dt, np.amin(self.y) * 1.2, np.amax(self.
                        y) * 1.2])
130             ax1.set_xlabel("Time [sec]")
131             ax1.set_ylabel("Amplitude")
132             ax1.set_title("Amplitude - Time")
133
134             ax2 = fig.add_subplot(212)
135             ax2.axis([10, self.fs / 2, np.amin(self.amp_spectrum)
136                   * 0.9, np.amax(self.amp_spectrum) * 1.1])
137             ax2.plot(self.freq_list, self.amp_spectrum, '-',
                    markersize=1)
138             ax2.set_xlabel("Frequency [Hz]")
139             ax2.set_ylabel("Amplitude")
140             ax2.set_title("Amplitude - Frequency")
141             plt.show()
142
143         def plot_power_gain_spectrum(self):
144             """ plot_power_gain_spectrum - power vs frequency
145             """
146             fig = plt.figure()
147             ax1 = fig.add_subplot(111)
148             ax1.set_xscale('log')
149             ax1.axis([10, self.fs / 2, np.amin(self.
                    power_gain_spectrum),
150                   np.amax(self.power_gain_spectrum) + 10])
```

```
151             ax1.plot(self.freq_list, self.gain_spectrum, '-',
                    markersize=1)
152             ax1.set_xlabel("Frequency [Hz]")
153             ax1.set_ylabel("power")
154             ax1.set_title("Power spectrum")
155             plt.show()
156
157         def plot_acf(self):
158             """ plot_power_gain_spectrum - auto correlation
                    function
159             """
160             fig = plt.figure()
161             ax1 = fig.add_subplot(111)
162             ax1.axis([-self.fft_N / 10, self.fft_N / 2, np.amin(
                    self.acf) * 1.1,
163                       np.amax(self.acf) * 1.1])
164             ax1.plot(list(range(self.fft_N)), self.acf, '-',
                    markersize=1)
165             ax1.set_xlabel("sample number")
166             ax1.set_ylabel("Correlation")
167             ax1.set_title("Auto correlation Function")
168             plt.show()
169
170         def plot_spectrogram_acf(self):
171             """ plot_power_gain_spectrum - 1. spectrogram. 2. short
                     time auto correlation function
172             """
173             # The degree of frame overlap when the window is
                    shifted
174             overlap = int(self.stft_N / 2)
175             # Length of wav
176             frame_length = len(self.y)
177             # Time per wav_file
178             time_of_file = frame_length * self.dt
179
180             # Define execute time
181             start = overlap * self.dt
182             stop = time_of_file
183             step = (self.stft_N - overlap) * self.dt
184             time_ruler = np.arange(start, stop, step)
185
186             # Definition initialization in transposition state
187             spec = np.zeros([len(time_ruler), 1 + int(self.stft_N /
                     2)])
188             st_acf = np.zeros([len(time_ruler), 1 + int(self.stft_N
                     / 2)])
189             pos = 0
```

```
190
191             for fft_index in range(len(time_ruler)):
192                 # Frame cut out
193                 frame = self.y[pos:pos + self.stft_N]
194                 # Frame cut out determination
195                 if len(frame) == self.stft_N:
196                     # Multiply window function
197                     windowed_data = self.stft_window * frame
198                     # FFT for only real demention
199                     fft_result = np.fft.rfft(windowed_data)
200                     fft_result = np.abs(fft_result)
201                     fft_result = fft_result / np.amax(fft_result)
202                     power_spectrum = np.abs(fft_result) ** 2
203                     acf = np.real(np.fft.ifft(
204                         power_spectrum / np.amax(power_spectrum)))
205                     acf = acf / np.amax(acf)
206                     # Find power spectrum
207                     fft_data = 10 * np.log10(np.abs(fft_result) **
                            2)
208                     # Assign to spec
209                     for i in range(len(spec[fft_index])):
210                         spec[fft_index][-i - 1] = fft_data[i]
211                         st_acf[fft_index][-i - 1] = acf[i]
212
213                     # Shift the window and execute the next frame.
214                     pos += (self.stft_N - overlap)
215
216             # ============ plot =============
217             fig = plt.figure()
218             ax1 = fig.add_subplot(111)
219             im = ax1.imshow(spec.T, extent=[0, time_of_file,
220                                             0, self.fs / 2],
221                             aspect="auto",
222                             cmap="inferno",
223                             interpolation="nearest",
224                             norm=Normalize(vmin=-80, vmax=0))
225             ax1.set_xlabel("Time[sec]")
226             ax1.set_ylabel("Frequency[Hz]")
227             ax1.set_title("Spectrogram")
228             pp1 = fig.colorbar(im, ax=ax1, orientation="vertical")
229             plt.tight_layout()
230             # pp1.set_clim(-80, 0)
231             pp1.set_label("power")
232             plt.show()
233
234             # ============ plot =============
235             fig = plt.figure()
```

```
236             ax2 = fig.add_subplot(111)
237             im2 = ax2.imshow(st_acf.T, extent=[0, time_of_file,
238                                                0, self.fs / 2],
239                              aspect="auto",
240                              cmap="inferno",
241                              interpolation="nearest",
242                              norm=Normalize(vmin=0, vmax=1.00))
243
244             ax2.set_xlabel("time[sec]")
245             ax2.set_ylabel("frequency[Hz]")
246             ax2.set_title("Short Time Auto correlation Function")
247
248             pp2 = fig.colorbar(im2, ax=ax2, orientation="vertical")
249             plt.tight_layout()
250             pp2.set_label("power")
251
252             plt.show()
253
254         def plot_all(self):
255             """ plot_all - execute all of plotting functions
256             """
257             self.plot_y_time()
258             self.plot_y_sample()
259             self.plot_freq_analysis()
260             self.plot_freq_analysis_log()
261             self.plot_power_gain_spectrum()
262             self.plot_acf()
263             self.plot_spectrogram_acf()
264
265
266     def completion_0(data_array):
267         """completion_0 - intermediate value completion
268         """
269         under_0list = np.where(data_array <= 0)
270         for under_0 in under_0list:
271             try:
272                 data_array[under_0] = 1e-8
273                 # 抜け値など，平均を取りたい場合コメントアウトを外す
274                 # data_array[under_0] = (
275                 # data_array[under_0-1] + data_array[under_0+1]) /
                        2
276             except IndexError as identifier:
277                 print(identifier)
278                 data_array[under_0] = 1e-8
279             return data_array
280
281
```

```
282     def define_window_function(N, name):
283         """define_window_function - defines window function
284         """
285         if name == "kaiser":
286             # TODO kaiser_param kakikaeru
287             # kaiser_param = input("Parameter of Kaiser Window : ")
288             kaiser_param = 5
289         else:
290             kaiser_param = 5
291 
292         windows_dic = {
293             "rectangular": np.ones(shape=N),
294             "hamming": np.hamming(M=N),
295             "hanning": np.hanning(M=N),
296             "bartlett": np.bartlett(M=N),
297             "blackman": np.blackman(M=N),
298             "kaiser": np.kaiser(M=N, beta=kaiser_param),
299         }
300 
301         if name in windows_dic:
302             return windows_dic[name]
303         else:
304             raise WindowNameNotFoundError
305 
306 
307     class WindowNameNotFoundError(Exception):
308         """Window name not found
309         """
310         print("Window Not Found")
```

ソースコード 1.1.1-2 wave_handler.py

```
1 
2 # coding:utf-8
3 import wave
4 
5 import numpy as np
6 
7 
8 class WaveHandler(object):
9 
10     def __init__(self, filename=None, **kwargs):
11         if filename:
12             self.wave_read(filename)
13         else:
14             self.ch = 1
15             self.width = 2
16             self.fs = 48000
```

```
17              self.chunk_size = 1024
18
19      def wave_read(self, filename):
20          # open wave file
21          wf = wave.open(filename, 'r')
22
23          # wave ファイルが持つ性質を取得
24          self.filename = filename
25          self.ch = wf.getnchannels()
26          self.width = wf.getsampwidth()
27          self.fs = wf.getframerate()
28          self.params = wf.getparams()
29          chunk_size = wf.getnframes()
30          # load wave data
31          amp = (2 ** 8) ** self.width / 2
32          data = wf.readframes(chunk_size) # バイナリ読み込み
33          # data = np.frombuffer(data, 'int16') # int に変換
34          data = np.frombuffer(data, dtype="int16") # int に変換
35          data = data / amp # 振幅正規化
36          self.chunk_size = chunk_size
37          self.data = data
38          # self.data = data[::self.ch] # date を 1ch に限定
39          wf.close() # 結果表示
40
41          print("分析対象ファイル: ", self.filename)
42          print("チャンクサイズ: ", self.chunk_size)
43          print("サンプルサイズのバイト数: ", self.width)
44          print("チャンネル数: ", self.ch)
45          print("wav ファイルのサンプリング周波数 :", self.fs)
46          print("パラメータ : ", self.params)
47          print("wav ファイルのデータ個数 :", len(self.data))
48
49      def wave_write(self, filename, data_array):
50          ww = wave.open(filename, 'w')
51          ww.setnchannels(self.ch)
52          ww.setsampwidth(self.width)
53          ww.setframerate(self.fs)
54          amp = (2 ** 8) ** self.width / 2
55          data_array = data_array / np.max(data_array)
56          write_array = np.array(data_array * amp, dtype=np.int16)
57          ww.writeframes(write_array)
58          ww.close()
```

ソースコード 1.1.1-3 decorators.py

```
1  import time
2  from functools import wraps
3
```

```
4
5  def stop_watch(func):
6      @wraps(func)
7      def wrapper(*args, **kwargs):
8          print("################ START ################")
9          start = time.time()
10         result = func(*args, **kwargs)
11         elapsed_time = time.time() - start
12         print("################ END ################")
13         elapsed_time = round(elapsed_time, 5)
14         print(
15             f"{elapsed_time}[sec] elapsed to execute the function:{
                   func.__name__}"
16         )
17         return result
18
19     return wrapper
20
21
22 def print_info(func):
23     @wraps(func)
24     def wrapper(*args, **kwargs):
25         print("func:", func.__name__)
26         print("args:", args)
27         print("kwargs:", kwargs)
28         result = func(*args, **kwargs)
29         print("result:", result)
30         return result
31
32     return wrapper
```

A.1.2 ツール

ソースコード 1.1.2-4 calc_convolution_wav.py

```
1  #! /usr/bin/env python
2  # coding: utf-8
3
4  import argparse
5  import pathlib
6
7  import numpy as np
8  import soundfile as sf
9
10
```

```
11 def main():
12     description = "This script calculates the convolution of two
           wav files."
13     usage = f"Usage: python {__file__} [-m full|valid|same] name1.
           wav name2.wav /path/to/output_name.wav"
14
15     # initial setting for command line arguments
16     parser = argparse.ArgumentParser(usage=usage, description=
           description)
17
18     parser.add_argument('input_paths',
19                         nargs="*",
20                         type=str,
21                         help='paths where the wav file is located.')
22
23     parser.add_argument('-o', '--output_path',
24                         nargs='?',
25                         default="conv.wav",
26                         type=str,
27                         help='Output file path where you want to
                               locate wav file. (default: current
                               directory)',
28                         )
29
30     parser.add_argument('-m', '--mode',
31                         action='store',
32                         nargs='?',
33                         default="full",
34                         type=str,
35                         choices=['full', 'valid', 'same'],
36                         help='Convolution type',
37                         )
38
39     # parse command line arguments
40     args = parser.parse_args()
41
42     input_paths = args.input_paths
43     output_path = pathlib.Path(args.output_path)
44     mode = args.mode
45
46     # validation of command line arguments
47     if len(input_paths) != 2:
48         print(f"Error: number of arguments is invalid. got:{len(
               input_paths)}")
49         print(parser.usage)
50         exit(1)
51
```

```
52      if mode != "full" and mode != "valid" and mode != "same":
53          print(f"Error: mode of convolution is invalid. got:{mode}")
54          print(parser.usage)
55
56          exit(1)
57
58      input_path_1 = input_paths[0]
59      input_path_2 = input_paths[1]
60      output_path = output_path
61
62      output_path = pathlib.Path(output_path)
63
64      # read audio file
65      data_1, fs_1 = sf.read(input_path_1)
66      data_2, fs_2 = sf.read(input_path_2)
67
68      # convolve
69      print("working...")
70      wav_out = np.convolve(data_1, data_2, mode=mode)
71
72      # write audio file
73      sf.write(file=str(output_path.stem + ".wav"), data=wav_out,
            samplerate=48000, endian="LITTLE",
74              format="WAV", subtype="PCM_16")
75
76      print(f"output files are saved at: {output_path}")
77
78
79  if __name__ == '__main__':
80      main()
```

ソースコード 1.1.2-5 calc_stereo2mono_LR.py

```
1   # coding:utf-8
2   import sys
3
4   import soundfile as sf
5
6
7   def main():
8       print("start")
9
10      filename = sys.argv[1]
11      filename_L = sys.argv[2]
12      filename_R = sys.argv[3]
13      print("input: ", filename)
14      print("output L: ", filename_L)
15      print("output R: ", filename_R)
```

```
16
17      data, fs = sf.read(filename)
18      mono_L = data[:, 0]
19      mono_R = data[:, 1]
20
21      sf.write(file=filename_L, data=mono_L, samplerate=48000, endian
            ="LITTLE", format="WAV", subtype="PCM_16")
22      sf.write(file=filename_R, data=mono_R, samplerate=48000, endian
            ="LITTLE", format="WAV", subtype="PCM_16")
23
24      print("done")
25
26
27  if __name__ == '__main__':
28      main()
```

ソースコード 1.1.2-6 calc_pseudo_ir_between_wav_files.py

```
1  #! /usr/bin/env python
2  # coding: utf-8
3
4  import sys
5
6  import numpy as np
7  import soundfile as sf
8
9
10 def main():
11     l_path = sys.argv[1]
12     r_path = sys.argv[2]
13     out_path = sys.argv[3]
14
15     data_l, fs_l = sf.read(l_path)
16     data_r, fs_r = sf.read(r_path)
17
18     F_l = np.fft.fft(data_l)
19     F_r = np.fft.fft(data_r)
20
21     F_pseudo_ir = F_l / F_r
22     f_pseude_ir = np.fft.ifft(F_pseudo_ir)
23     f_pseude_ir_real = np.real(f_pseude_ir)
24
25     sig = f_pseude_ir_real
26     sf.write(file=out_path, data=sig, samplerate=48000, endian="
           LITTLE", format="WAV", subtype="PCM_16")
27
28
29 if __name__ == '__main__':
```

```
30      main()
```

ソースコード 1.1.2-7 csv_to_wav_each_column.py

```
1  #! /usr/bin/env python
2  # coding: utf-8
3
4  import pathlib
5  import sys
6
7  import numpy as np
8  import pandas as pd
9  import soundfile as sf
10
11
12 def main():
13     input_path = sys.argv[1]
14     output_path = sys.argv[2]
15
16     df = pd.read_csv(input_path)
17     output_path = pathlib.Path(output_path)
18
19     arr = np.array(df)
20
21     for i, data in enumerate(arr.T):
22         sf.write(file=str(output_path.stem + f"_col{i}.wav"), data=
               data, samplerate=48000, endian="LITTLE",
23                  format="WAV", subtype="PCM_16")
24
25     print(f"output files are saved at: {output_path}")
26
27
28 if __name__ == '__main__':
29     main()
```

ソースコード 1.1.2-8 calc_subtracted_wav.py

```
1  # coding: utf-8
2
3
4  import sys
5
6  import librosa
7  import numpy as np
8  import soundfile as sf
9
10
11 def main():
```

```
12      input_file_path = sys.argv[1]
13      noise_file_path = sys.argv[2]
14      output_file_path = sys.argv[3]
15
16      print('input file:', input_file_path)
17      data, data_fs = librosa.load(input_file_path, sr=None, mono=
            True)
18      data_st = librosa.stft(data)
19      data_st_abs = np.abs(data_st)
20      angle = np.angle(data_st)
21      b = np.exp(1.0j * angle)
22
23      print('noise file:', noise_file_path)
24      noise_data, noise_fs = librosa.load(noise_file_path, sr=None,
            mono=True)
25      noise_data_st = librosa.stft(noise_data)
26      noise_data_st_abs = np.abs(noise_data_st)
27      mean_noise_abs = np.mean(noise_data_st_abs, axis=1)
28
29      subtracted_data = data_st_abs - mean_noise_abs.reshape(
30          (mean_noise_abs.shape[0], 1)) # reshape for broadcast to
                subtract
31      subtracted_data_phase = subtracted_data * b # apply phase
            information
32      y = librosa.istft(subtracted_data_phase) # back to time domain
            signal
33
34      # save as a wav file
35      sf.write(file=str(output_file_path), data=y, samplerate=data_fs
            , endian="LITTLE", format="WAV", subtype="PCM_16")
36      print('output file:', output_file_path)
37
38
39  if __name__ == '__main__':
40      main()
```

ソースコード 1.1.2-9 generate_white_noise_as_wav.py

```
1   #!/usr/bin/env python3
2   # coding: utf-8
3
4   import argparse
5   import pathlib
6
7   import numpy as np
8   import soundfile as sf
9
10
```

```
def main():
    parser = argparse.ArgumentParser(description="This script makes white noise to designated path.")

    parser.add_argument('duration',
                        action='store',
                        type=float,
                        help='The length of noise.')

    parser.add_argument('-d', '--dst_path',
                        action='store',
                        nargs='?',
                        const="/tmp",
                        default=".",
                        type=str,
                        help='Directory path where you want to locate output files. (default: current directory)')

    args = parser.parse_args()
    duration = args.duration
    output_dir = pathlib.Path(args.dst_path)
    output_name = pathlib.Path.joinpath(output_dir, f"white_noise_{duration}s.wav")

    sig = np.random.rand(int(duration * 48000))

    sf.write(file=str(output_name), data=sig, samplerate=48000, endian="LITTLE", format="WAV", subtype="PCM_16")


if __name__ == '__main__':
    main()
```

ソースコード 1.1.2-10 plot_animation_from_csv.py

```
# encoding: utf-8

import argparse
import pathlib

import matplotlib.pyplot as plt
import pandas as pd
from matplotlib import animation

plt.rcParams['font.family'] = 'IPAPGothic'
plt.rcParams['font.size'] = 11
plt.rcParams['xtick.direction'] = 'in'
```

```
13 plt.rcParams['ytick.direction'] = 'in'
14 plt.rcParams['xtick.top'] = True
15 plt.rcParams['ytick.right'] = True
16 plt.rcParams['xtick.major.width'] = 1.0
17 plt.rcParams['ytick.major.width'] = 1.0
18 plt.rcParams['axes.linewidth'] = 1.0
19 plt.rcParams['figure.figsize'] = (8, 7)
20 plt.rcParams['figure.dpi'] = 100
21 plt.rcParams['figure.subplot.hspace'] = 0.3
22 plt.rcParams['figure.subplot.wspace'] = 0.3
23
24
25 def main():
26     description = "This script plots graph from a csv file with 3
          columns."
27     parser = argparse.ArgumentParser(description=description)
28
29     parser.add_argument('csv_path',
30                         action='store',
31                         const=None,
32                         default=None,
33                         type=str,
34                         help='Directory path where the csv file is
                               located.',
35                         metavar=None)
36
37     parser.add_argument('-d', '--dst_path',
38                         action='store',
39                         nargs='?',
40                         const="/tmp",
41                         default=".",
42                         type=str,
43                         help='Directory path where you want to
                               locate png files. (default: current
                               directory)',
44                         metavar=None)
45
46     parser.add_argument('-s', '--samples',
47                         action='store',
48                         nargs='?',
49                         # const="/tmp",
50                         default=50,
51                         type=int,
52                         help='Samples to draw.',
53                         metavar=None)
54
55     parser.add_argument('-i', '--interval',
```

```
56                          action='store',
57                          nargs='?',
58                          # const="/tmp",
59                          default=100,
60                          type=int,
61                          help='Interval to draw.',
62                          metavar=None)
63
64      args = parser.parse_args()
65
66      input_path = args.csv_path
67      input_path = pathlib.Path(input_path)
68
69      df = pd.read_csv(input_path, header=None)
70      print("analize file name: ", input_path)
71
72      d, y, e = df[0], df[1], df[2]
73
74      output_dir = pathlib.Path(args.dst_path)
75      output_name = pathlib.Path(input_path.name).with_suffix(".gif")
76      # output_name = pathlib.Path(input_path.name).with_suffix(".mp4
            ")
77      output_path = pathlib.Path.joinpath(output_dir, output_name)
78
79      samples = args.samples
80      interval = args.interval
81
82      fig, (ax1, ax2) = plt.subplots(2, 1)
83      ax1.set_xlabel("iteration n")
84      ax1.set_ylim((-1.6, 1.6))
85      ax2.set_xlabel("iteration n")
86      ax2.set_ylim((-0.5, 0.5))
87
88      ani = animation.FuncAnimation(fig, update, fargs=(d[:samples],
            y[:samples], e[:samples], ax1, ax2),
89                                    interval=interval, frames=int(
                                          samples / 2))
90      ani.save(output_path, writer='imagemagick')
91      # ani.save(output_path, writer='ffmpeg')
92
93      print("\nfilterd data plot is saved at: ", output_path, "\n")
94
95
96  def update(i, d, y, e, ax1, ax2):
97      if i == 0:
98          ax1.legend(loc='upper right')
99          ax1.set_title("Desired value, Filter output and Filter
```

```
            error")
100     else:
101         plt.cla()
102     i = i * 2
103     ax1.plot(d[0:i], color="b", alpha=0.7, label="desired value d(
            n)")
104     ax1.plot(y[0:i], color="r", alpha=0.7, label="filter output y(
            n)")
105
106     ax2.plot(e[0:i], color="y", alpha=1.0, label="filter error e(n
            )")
107
108
109 if __name__ == '__main__':
110     main()
```

ソースコード 1.1.2-11 plot_from_csv.py

```
1  # encoding: utf-8
2
3  import argparse
4  import pathlib
5
6  import matplotlib
7  import numpy as np
8  import pandas as pd
9
10 matplotlib.use('Agg')
11 import matplotlib.pyplot as plt
12
13 """
14 Note that the modules (numpy, maplotlib, wave, scipy) are properly
       installed on your environment.
15
16 Plot wave, spectrum, save them as pdf and png at same directory.
17
18 Example:
19     python calc_wave_analysis.py IR_test.wav
20 """
21
22 plt.rcParams['font.family'] = 'IPAPGothic'
23 plt.rcParams['xtick.direction'] = 'in'
24 plt.rcParams['ytick.direction'] = 'in'
25 plt.rcParams['xtick.top'] = True
26 plt.rcParams['ytick.right'] = True
27 plt.rcParams['xtick.major.width'] = 1.0
28 plt.rcParams['ytick.major.width'] = 1.0
29 plt.rcParams['font.size'] = 11
```

```
30 plt.rcParams['axes.linewidth'] = 1.0
31 plt.rcParams['figure.figsize'] = (8, 7)
32 plt.rcParams['figure.dpi'] = 300
33 plt.rcParams['figure.subplot.hspace'] = 0.3
34 plt.rcParams['figure.subplot.wspace'] = 0.3
35
36
37 def main():
38     parser = argparse.ArgumentParser(description="This script plots
           graph from a csv file with 3 columns.")
39
40     parser.add_argument('csv_path',
41                         action='store',
42                         type=str,
43                         help='Directory path where the csv file is
                               located.',
44                         metavar=None)
45
46     parser.add_argument('-d', '--dst_path',
47                         action='store',
48                         nargs='?',
49                         default=".",
50                         type=str,
51                         help='Directory path where you want to
                               locate png files. (default: current
                               directory)',
52                         metavar=None)
53
54     parser.add_argument('-l', '--log',
55                         action='store_true',
56                         help='Use y-axis logarithmic display.')
57
58     args = parser.parse_args()
59
60     input_name = args.csv_path
61     input_name = pathlib.Path(input_name)
62
63     is_logarithm = args.log
64
65     df = pd.read_csv(input_name, header=None)
66     print("analize file name: ", input_name)
67
68     d, y, e = df[0], df[1], df[2]
69
70     fig, (ax1, ax2) = plt.subplots(2, 1)
71     if is_logarithm:
72         ax1.set_yscale("log")
```

```
73          ax2.set_yscale("log")
74          d /= np.max(d)
75          y /= np.max(d)
76          e /= np.max(e)
77
78      ax1.plot(d, "b--", alpha=0.5, label="desired signal d(n)")
79      ax1.plot(y, "r-", alpha=0.5, label="output y(n)")
80      ax1.legend()
81      ax2.plot(e, "y-", alpha=1.0, label="error e(n)")
82      plt.grid()
83      ax2.legend()
84      plt.title('ADF Output')
85
86      output_dir = pathlib.Path(args.dst_path)
87      output_name = pathlib.Path(input_name.name).with_suffix(".png")
88      output_path = pathlib.Path.joinpath(output_dir, output_name)
89      plt.savefig(output_path)
90      print("\nfilterd data plot is saved at: ", output_path, "\n")
91
92
93  if __name__ == '__main__':
94      main()
```

ソースコード 1.1.2-12 plot_multiwave.py

```
1  #!/usr/bin/env python
2  # coding: utf-8
3
4  # Usage:
5  # python3 plot_multiwave.py foo.wav
6  # then you can see the wave's abstruct
7
8  import signal
9  import sys
10
11 import matplotlib
12 matplotlib.use('Agg')
13 import matplotlib.pyplot as plt
14 import soundfile as sf
15
16 signal.signal(signal.SIGINT, signal.SIG_DFL)
17
18 plt.rcParams['font.family'] = 'IPAPGothic'
19 plt.rcParams['font.size'] = 11
20 plt.rcParams['xtick.direction'] = 'in'
21 plt.rcParams['ytick.direction'] = 'in'
22 plt.rcParams['xtick.top'] = True
23 plt.rcParams['ytick.right'] = True
```

```
plt.rcParams['xtick.major.width'] = 1.0
plt.rcParams['ytick.major.width'] = 1.0
plt.rcParams['axes.linewidth'] = 1.0
plt.rcParams['figure.figsize'] = (8, 7)
plt.rcParams['figure.dpi'] = 300
plt.rcParams['figure.subplot.hspace'] = 0.3
plt.rcParams['figure.subplot.wspace'] = 0.3


def main():
    args = sys.argv

    if len(args) < 2:
        print("error : please pass the wave file argument")
        print("Usage: python3 plot_muiltiwave.py foo.wav")
        sys.exit()

    for i in range(len(args) - 1):
        data, sr = sf.read(args[i + 1])

        plt.plot(data, alpha=0.5, label=args[i + 1])
        plt.legend()
        plt.xlabel("Iteration")
        plt.grid(True)

    plt.show()


if __name__ == '__main__':
    main()
```

ソースコード 1.1.2-13 plot_spectrogram_librosa.py

```
#!/usr/bin/env python
# encoding: utf-8

import argparse
import pathlib

import librosa
import librosa.display
import matplotlib
import numpy as np
import pandas as pd

matplotlib.use('Agg')
import matplotlib.pyplot as plt

```

```
16 plt.rcParams['font.family'] = 'IPAPGothic'
17 plt.rcParams['font.size'] = 16
18 plt.rcParams['xtick.direction'] = 'in'
19 plt.rcParams['ytick.direction'] = 'in'
20 plt.rcParams['xtick.top'] = True
21 plt.rcParams['ytick.right'] = True
22 plt.rcParams['xtick.major.width'] = 1.0
23 plt.rcParams['ytick.major.width'] = 1.0
24 plt.rcParams['axes.linewidth'] = 1.0
25 plt.rcParams['figure.figsize'] = (8, 7)
26 plt.rcParams['figure.dpi'] = 300
27 plt.rcParams['figure.subplot.hspace'] = 0.3
28 plt.rcParams['figure.subplot.wspace'] = 0.3
29
30
31 def main():
32     description = "This script plots spectrogram from csv or wav
          file."
33     parser = argparse.ArgumentParser(description=description)
34
35     parser.add_argument('input_path',
36                         action='store',
37                         type=str,
38                         help='Directory path where the input file is
                                located.',
39                         metavar=None)
40
41     parser.add_argument('-d', '--dst_path',
42                         action='store',
43                         nargs='?',
44                         const="/tmp",
45                         default=".",
46                         type=str,
47                         help='Directory path where you want to
                               locate img files. (default: current
                               directory)',
48                         metavar=None)
49
50     parser.add_argument('-l', '--log',
51                         action='store_true',
52                         help='Use y-axis logarithmic display.')
53
54     args = parser.parse_args()
55
56     input_path = pathlib.Path(args.input_path)
57     output_dir = pathlib.Path(args.dst_path)
58     is_logarithm = args.log
```

```
59
60      sr = 48000
61      data = []
62      if input_path.suffix == ".wav":
63          data, sr = librosa.load(str(input_path), sr=None)
64      elif input_path.suffix == ".csv":
65          df = pd.read_csv(str(input_path), header=None)
66          data = np.array(df[0], dtype=np.float)
67      else:
68          parser.usage()
69          exit(1)
70
71      d = np.abs(librosa.stft(data))
72      log_D = librosa.amplitude_to_db(d, ref=np.max)
73
74      plt.figure(figsize=(8, 7))
75      if is_logarithm:
76          librosa.display.specshow(log_D, sr=sr, x_axis='time',
                y_axis='log')
77          librosa.display.specshow(log_D, sr=sr, x_axis='time',
                y_axis='log')
78      else:
79          librosa.display.specshow(log_D, sr=sr, x_axis='s', y_axis='
                linear')
80
81      plt.set_cmap("inferno")
82      plt.xlabel("Time [sec]")
83      plt.ylabel("Frequency [Hz]")
84      plt.title('Spectrogram')
85      plt.colorbar(format='%2.0f dB')
86      plt.tight_layout()
87      plt.show()
88
89      output_name = pathlib.Path(input_path.name).with_suffix(".png")
90      output_path = pathlib.Path.joinpath(output_dir, output_name)
91
92      plt.savefig(str(output_path))
93
94      print(f"\nimage is saved at: {output_path}\n")
95
96
97  if __name__ == '__main__':
98      main()
```

A.2 Go言語で制作したプログラム

A.2.1 ツール

ソースコード 1.2.1-1 ディレクトリ構成

```
.
├── converter.go
├── filehandler.go
├── calculator.go
└── cmd
    ├── calc_noise_mix
    │   └── main.go
    ├── convolve_wav
    │   └── main.go
    ├── convolve_wav_coef
    │   └── main.go
    ├── convolve_wav_fast
    │   └── main.go
    ├── csv_to_wav
    │   └── main.go
    ├── drone_json_generator
    │   └── main.go
    ├── multirecord
    │   ├── main.go
    │   ├── multirecord.go
    │   ├── util.go
    │   └── wav_to_DSB.go
    └── wav_to_DSB
        ├── main.go
        ├── util.go
        └── wtod.go
```

ソースコード 1.2.1-2 converter.go

```
package goresearch

import (
    "math"

    "gonum.org/v1/gonum/floats"
)

func abs(x int) int {
    if x < 0 {
        return -1 * x
    }
```

```
13      return x
14  }
15
16  func AbsFloat64s(fs []float64) []float64 {
17      fsAbs := make([]float64, len(fs))
18      for i, v := range fs {
19          fsAbs[i] = math.Abs(v)
20      }
21      return fsAbs
22  }
23
24  func AbsInts(is []int) []int {
25      isAbs := make([]int, len(is))
26      for i, v := range is {
27          isAbs[i] = abs(v)
28      }
29      return isAbs
30  }
31
32  func NormToMaxInt16(data []float64) []float64 {
33
34      maxAmp := floats.Max(AbsFloat64s(data))
35      if maxAmp > math.MaxInt16+1 {
36          reductionRate := math.MaxInt16 / maxAmp
37          for i, _ := range data {
38              data[i] *= reductionRate
39          }
40      }
41      return data
42  }
43
44  func Int16sToInts(i16s []int16) []int {
45      var is = make([]int, len(i16s))
46      for i, v := range i16s {
47          is[i] = int(v)
48      }
49      return is
50  }
51
52  func Float64sToInts(fs []float64) []int {
53      is := make([]int, len(fs))
54      for i, s := range fs {
55          is[i] = int(s)
56      }
57      return is
58  }
59
```

```
func IntsToFloat64s(is []int) []float64 {
    fs := make([]float64, len(is))
    for i, s := range is {
        fs[i] = float64(s)
    }
    return fs
}
func Float64sToComplex128s(fs []float64) []complex128 {
    cs := make([]complex128, len(fs))
    for i, f := range fs {
        cs[i] = complex(f, 0)
    }
    return cs
}

func Complex128sToFloat64s(cs []complex128) []float64 {
    fs := make([]float64, len(cs))
    for i, c := range cs {
        fs[i] = real(c)
    }
    return fs
}
```

ソースコード 1.2.1-3 calculator.go

```
package goresearch

import (
        "errors"
        "fmt"
        "math"

        "github.com/mjibson/go-dsp/fft"
)

func CalcAdjustedRMS(cleanRMS float64, snr float64) (noiseRMS
    float64) {
        a := snr / 20
        noiseRMS = cleanRMS / (math.Pow(10, a))
        return noiseRMS
}

func CalcRMS(amp []float64) float64 {
        var sum float64
        for _, v := range amp {
                sum += v * v
        }
        return math.Sqrt(sum / float64(len(amp)))
```

```
}

func CalcMSE(a []float64, b []float64) (float64, error) {
        if b == nil {
                b = make([]float64, len(a))
        } else if len(a) != len(b) {
                return 0, errors.New("length of a and b must agree
                    ")
        }

        var sum float64
        for i := 0; i < len(a); i++ {
                sum += (a[i] - b[i]) * (a[i] - b[i])
        }

        return sum / float64(len(a)), nil
}

func Convolve(xs, ys []float64) []float64 {
        var convLen, sumLen = len(xs), len(ys)
        if convLen > sumLen {
                ys = append(ys, make([]float64, convLen-sumLen)...)
        } else {
                convLen, sumLen = sumLen, convLen
                xs = append(xs, make([]float64, convLen-sumLen)...)
        }
        var rs = make([]float64, convLen)
        var nodeSum float64
        var i, j int
        for i = 0; i < convLen; i++ {
                for j = 0; j < sumLen; j++ {
                        if i-j < 0 {
                                continue
                        }
                        nodeSum += xs[i-j] * ys[j]
                }
                rs[i] = nodeSum
                nodeSum = 0
        }
        return rs
}

// FastConvolve - Linear fast convolution
func FastConvolve(xs, ys []float64) []float64 {
        L := len(xs)
        N := len(ys)
        M := N + L - 1
```

```
69
70          // zero padding
71          xsz := append(xs, make([]float64, M-L)...)
72          ysz := append(ys, make([]float64, M-N)...)
73
74          var rs = make([]float64, M)
75          var Rs = make([]complex128, M)
76
77          fmt.Printf("calcurating fft...\n")
78
79          Xs := fft.FFT(Float64sToComplex128s(xsz))
80          Ys := fft.FFT(Float64sToComplex128s(ysz))
81
82          for i := 0; i < M; i++ {
83                  // progress
84                  fmt.Printf("calucurating convolution... %d%%\r", (i
                        +1)*100/M)
85                  Rs[i] = Xs[i] * Ys[i]
86          }
87          fmt.Printf("\ncalcurating ifft...\n")
88
89          rs = Complex128sToFloat64s(fft.IFFT(Rs))
90
91          return rs
92  }
93
94  func LinSpace(start, end float64, n int) []float64 {
95          res := make([]float64, n)
96          if n == 1 {
97                  res[0] = end
98                  return res
99          }
100         delta := (end - start) / (float64(n) - 1)
101         for i := 0; i < n; i++ {
102                 res[i] = start + (delta * float64(i))
103         }
104         return res
105 }
```

ソースコード 1.2.1-4 filehandler.go

```
1  package goresearch
2
3  import (
4          "bufio"
5          "fmt"
6          "os"
7          "path/filepath"
```

```
8         "strconv"
9         "strings"
10
11        "github.com/go-audio/audio"
12        "github.com/go-audio/wav"
13 )
14
15 func ReadDataFromCSV(inputPath string) (ds []float64, ys []float64,
       es []float64) {
16        fr, err := os.Open(inputPath)
17        check(err)
18        sc := bufio.NewScanner(fr)
19        var ss []string
20        var d float64
21        var y float64
22        var e float64
23        for sc.Scan() {
24                ss = strings.Split(sc.Text(), ",")
25                d, err = strconv.ParseFloat(ss[0], 64)
26                check(err)
27                ds = append(ds, d)
28                y, err = strconv.ParseFloat(ss[1], 64)
29                check(err)
30                ys = append(ys, y)
31                e, err = strconv.ParseFloat(ss[2], 64)
32                check(err)
33                es = append(es, e)
34        }
35        return ds, ys, es
36 }
37
38 func ReadDataFromCSVOne(inputPath string, column int) (fs []float64
      ) {
39        fr, err := os.Open(inputPath)
40        check(err)
41        sc := bufio.NewScanner(fr)
42        var ss []string
43        var f float64
44        for sc.Scan() {
45                ss = strings.Split(sc.Text(), ",")
46                f, err = strconv.ParseFloat(ss[column], 64)
47                check(err)
48                fs = append(fs, f)
49        }
50        return fs
51 }
52
```

```
53 func ReadCoefFromCSV(inputPath string) (ws []float64) {
54         fr, err := os.Open(inputPath)
55         check(err)
56         sc := bufio.NewScanner(fr)
57         var ss []string
58         var w float64
59         for sc.Scan() {
60                 ss = strings.Split(sc.Text(), ",")
61                 w, err = strconv.ParseFloat(ss[0], 64)
62                 check(err)
63                 ws = append(ws, w)
64         }
65         return ws
66 }
67
68 func ReadDataFromWav(name string) []int {
69         f, err := os.Open(name)
70         check(err)
71         defer f.Close()
72         wavFile := wav.NewDecoder(f)
73         check(err)
74
75         wavFile.ReadInfo()
76         ch := int(wavFile.NumChans)
77         //byteRate := int(w.BitDepth/8) * ch
78         //bps := byteRate / ch
79         fs := int(wavFile.SampleRate)
80         fmt.Println("ch", ch, "fs", fs)
81
82         buf, err := wavFile.FullPCMBuffer()
83         check(err)
84         fmt.Printf("SourceBitDepth: %v\n", buf.SourceBitDepth)
85
86         return buf.Data
87 }
88
89 func SaveDataAsWav(data []float64, dataDir string, name string) {
90         outputPath := filepath.Join(dataDir, name+".wav")
91         fw, err := os.Create(outputPath)
92         check(err)
93
94         const (
95                 SampleRate = 48000
96                 BitsPerSample = 16
97                 NumChannels = 1
98                 PCM = 1
99         )
```

```
100
101         w1 := wav.NewEncoder(fw, SampleRate, BitsPerSample,
                NumChannels, PCM)
102         aBuf := new(audio.IntBuffer)
103         aBuf.Format = &audio.Format{
104                 NumChannels: NumChannels,
105                 SampleRate: SampleRate,
106         }
107         aBuf.SourceBitDepth = BitsPerSample
108
109         aBuf.Data = Float64sToInts(data)
110         err = w1.Write(aBuf)
111         check(err)
112
113         err = w1.Close()
114         check(err)
115
116         err = fw.Close()
117         check(err)
118
119         fmt.Printf("\nwav file saved at: %v\n", outputPath)
120 }
121
122 func SaveDataAsCSV(d, y, e, mse []float64, dataDir string, testName
        string) {
123         n := len(d)
124         fw, err := os.Create(filepath.Join(dataDir, testName+".csv
                "))
125         check(err)
126         writer := bufio.NewWriter(fw)
127         for i := 0; i < n; i++ {
128                 if i >= len(mse) {
129                         _, err = fmt.Fprintf(writer, "%g,%g,%g\n",
                                d[i], y[i], e[i])
130                         check(err)
131                         continue
132                 }
133                 _, err = fmt.Fprintf(writer, "%g,%g,%g,%g\n", d[i
                        ], y[i], e[i], mse[i])
134                 check(err)
135         }
136         err = writer.Flush()
137         check(err)
138         err = fw.Close()
139         check(err)
140 }
141
```

```
142 func SaveDataAsCSVOne(fs []float64, dataDir string, testName string
    ) {
143         n := len(fs)
144         fw, err := os.Create(filepath.Join(dataDir, testName+".csv
                ""))
145         check(err)
146         writer := bufio.NewWriter(fw)
147         for i := 0; i < n; i++ {
148                 _, err = fmt.Fprintf(writer, "%g\n", fs[i])
149                 check(err)
150         }
151         err = writer.Flush()
152         check(err)
153         err = fw.Close()
154         check(err)
155
156         fmt.Printf("output file is saved at: %v\n", testName+".csv
                ")
157 }
158
159 func SplitPathAndExt(path string) (string, string) {
160         return filepath.Join(filepath.Dir(filepath.Clean(path)),
                filepath.Base(path[:len(path)-len(filepath.Ext(path
                ))])), filepath.Ext(path)
161 }
162
163 func check(err error) {
164         if err != nil {
165                 panic(err)
166         }
167 }
```

ソースコード 1.2.1-5 cmd/calc_noise_mix/main.go

```
1  package main
2
3  import (
4      "flag"
5      "fmt"
6      "log"
7      "math"
8      "math/rand"
9      "os"
10     "path/filepath"
11     "strconv"
12
13     "github.com/go-audio/audio"
14     "github.com/go-audio/wav"
```

```
15      "gonum.org/v1/gonum/floats"
16
17      "github.com/tetsuzawa/research-tools/goresearch"
18  )
19
20  func main() {
21
22      log.SetFlags(log.LstdFlags | log.Lshortfile)
23
24      var (
25          cleanFilepath string
26          noiseFilepath string
27          outputDir string
28          snrStart float64
29          snrEnd float64
30          snrDiv int
31      )
32
33      flag.StringVar(&cleanFilepath, "clean", "/path/to/clean_file.
            wav", "designate clean file path")
34      flag.StringVar(&noiseFilepath, "noise", "/path/to/noise_file.
            wav", "designate noise file path")
35      flag.StringVar(&outputDir, "output", "/path/to/dir/", "
            designate ouput directory")
36      flag.Float64Var(&snrStart, "start", -40, "designate start value
             of S/N Rate")
37      flag.Float64Var(&snrEnd, "end", 40, "designate end value of S/
            N Rate")
38      flag.IntVar(&snrDiv, "div", 19, "designate number of divisions
            ")
39
40      flag.Parse()
41
42      if cleanFilepath == "/path/to/clean_file.wav" ||
43          noiseFilepath == "/path/to/noise_file.wav" ||
44          outputDir == "/path/to/dir/" {
45          flag.Usage()
46          os.Exit(1)
47      }
48      fmt.Println("clean file path:", cleanFilepath)
49      fmt.Println("noise file path:", noiseFilepath)
50      fmt.Println("ouput directory:", outputDir)
51      fmt.Println("start value of S/N Rate:", snrStart)
52      fmt.Println("end value of S/N Rate:", snrEnd)
53      fmt.Println("number of divisions:", snrDiv)
54
55      f1, err := os.Open(cleanFilepath)
```

```
56      check(err)
57      w1 := wav.NewDecoder(f1)
58
59      f2, err := os.Open(noiseFilepath)
60      check(err)
61      w2 := wav.NewDecoder(f2)
62
63      w1.ReadInfo()
64      w2.ReadInfo()
65      ch1 := int(w1.NumChans)
66      ch2 := int(w2.NumChans)
67      bitDepth1 := int(w1.BitDepth)
68      bitDepth2 := int(w2.BitDepth)
69      bps1 := bitDepth1 / 8
70      bps2 := bitDepth2 / 8
71      fs1 := int(w1.SampleRate)
72      fs2 := int(w2.SampleRate)
73
74      buf1, err := w1.FullPCMBuffer()
75      check(err)
76      buf2, err := w2.FullPCMBuffer()
77      check(err)
78
79      err = f1.Close()
80      check(err)
81      err = f2.Close()
82      check(err)
83
84      if ch1 != ch2 ||
85          bitDepth1 != bitDepth2 ||
86          bps1 != bps2 ||
87          fs1 != fs2 {
88          err = fmt.Errorf("format of wav files are not agree")
89          panic(err)
90      }
91
92      cleanAMP := goresearch.IntsToFloat64s(buf1.Data)
93      noiseAMP := goresearch.IntsToFloat64s(buf2.Data)
94
95      cleanRMS := goresearch.CalcRMS(cleanAMP)
96
97      var start int
98      var cutNoiseAmp []float64
99      if len(cleanAMP) == len(noiseAMP) {
100         start = 0
101         cutNoiseAmp = noiseAMP[start : start+len(cleanAMP)]
102     } else if len(cleanAMP) > len(noiseAMP) {
```

```
103         start = rand.Intn(len(cleanAMP) - len(noiseAMP))
104         cleanAMP = cleanAMP[start : start+len(cleanAMP)]
105         cutNoiseAmp = noiseAMP
106     } else {
107         start = rand.Intn(len(noiseAMP) - len(cleanAMP))
108         cutNoiseAmp = noiseAMP[start : start+len(cleanAMP)]
109     }
110     noiseRMS := goresearch.CalcRMS(cutNoiseAmp)
111     snrList := goresearch.LinSpace(snrStart, snrEnd, snrDiv)
112 
113     var (
114         adjustedNoiseAmp = make([]float64, len(cutNoiseAmp))
115         mixedAmp = make([]float64, len(cutNoiseAmp))
116         fw *os.File
117         ww *wav.Encoder
118         wBuf = new(audio.IntBuffer)
119         outputName string
120         outputPath string
121     )
122     wBuf.Format = &audio.Format{
123         NumChannels: ch1,
124         SampleRate: fs1,
125     }
126     wBuf.SourceBitDepth = bitDepth1
127     for _, snr := range snrList {
128 
129         adjustedNoiseRMS := goresearch.CalcAdjustedRMS(cleanRMS,
                snr)
130 
131         for i, v := range cutNoiseAmp {
132             adjustedNoiseAmp[i] = v * (adjustedNoiseRMS / noiseRMS)
133             mixedAmp[i] = cleanAMP[i] + adjustedNoiseAmp[i]
134         }
135         maxAmp := floats.Max(goresearch.AbsFloat64s(mixedAmp))
136         if maxAmp > math.MaxInt16+1 {
137             reductionRate := math.MaxInt16 / maxAmp
138             for i, _ := range cutNoiseAmp {
139                 mixedAmp[i] *= reductionRate
140             }
141         }
142 
143         wBuf.Data = goresearch.Float64sToInts(mixedAmp)
144 
145         outputName, _ = goresearch.SplitPathAndExt(cleanFilepath)
146         outputPath = filepath.Join(outputDir, filepath.Base(
                outputName)+"_snr"+strconv.Itoa(int(snr))+".wav")
147         fw, err = os.Create(outputPath)
```

```
148         check(err)
149         ww = wav.NewEncoder(fw, fs1, bitDepth1, ch1, 1)
150         err = ww.Write(wBuf)
151         check(err)
152         err = ww.Close()
153         check(err)
154 
155     }
156     fmt.Printf("\nSuccessfully created following SN Rate files!!
            SNR: %v\n", snrList)
157 }
158 
159 func check(err error) {
160     if err != nil {
161         panic(err)
162     }
163 }
```

ソースコード 1.2.1-6 cmd/convolve_wav/main.go

```
1  package main
2  
3  import (
4          "flag"
5          "fmt"
6          "log"
7          "math"
8          "os"
9  
10         "github.com/go-audio/audio"
11         "github.com/go-audio/wav"
12         "gonum.org/v1/gonum/floats"
13 
14         "github.com/tetsuzawa/research-tools/goresearch"
15 )
16 
17 func main() {
18 
19         log.SetFlags(log.LstdFlags | log.Lshortfile)
20 
21         var (
22                 inputFilePath1 string
23                 inputFilePath2 string
24                 outputFilePath string
25         )
26 
27         flag.StringVar(&inputFilePath1, "x", "/path/to/name.wav", "
               designate input wav file path")
```

```
28          flag.StringVar(&inputFilePath2, "y", "/path/to/name.wav", "
                designate input wav file path")
29          flag.StringVar(&outputFilePath, "o", "/path/to/name.wav", "
                designate output wav file path")
30
31          flag.Parse()
32
33          if inputFilePath1 == "/path/to/name.wav" ||
34                  inputFilePath2 == "/path/to/name.wav" ||
35                  outputFilePath == "/path/to/name.wav" {
36                  flag.Usage()
37                  os.Exit(1)
38          }
39
40          fmt.Println("wav_1 file path:", inputFilePath1)
41          fmt.Println("wav_2 file path:", inputFilePath2)
42          fmt.Println("output file path:", outputFilePath)
43
44          f1, err := os.Open(inputFilePath1)
45          check(err)
46          w1 := wav.NewDecoder(f1)
47
48          f2, err := os.Open(inputFilePath2)
49          check(err)
50          w2 := wav.NewDecoder(f2)
51
52          w1.ReadInfo()
53          w2.ReadInfo()
54          ch1 := int(w1.NumChans)
55          ch2 := int(w2.NumChans)
56          bitDepth1 := int(w1.BitDepth)
57          bitDepth2 := int(w2.BitDepth)
58          bps1 := bitDepth1 / 8
59          bps2 := bitDepth2 / 8
60          fs1 := int(w1.SampleRate)
61          fs2 := int(w2.SampleRate)
62
63          buf1, err := w1.FullPCMBuffer()
64          check(err)
65          buf2, err := w2.FullPCMBuffer()
66          check(err)
67
68          err = f1.Close()
69          check(err)
70          err = f2.Close()
71          check(err)
72
```

```
73          if ch1 != ch2 ||
74                  bitDepth1 != bitDepth2 ||
75                  bps1 != bps2 ||
76                  fs1 != fs2 {
77                  err = fmt.Errorf("format of wav files are not agree
                        ")
78                  panic(err)
79          }
80
81          amp1 := goresearch.IntsToFloat64s(buf1.Data)
82          amp2 := goresearch.IntsToFloat64s(buf2.Data)
83
84          ampOut := goresearch.Convolve(amp1, amp2)
85
86          var (
87                  fw *os.File
88                  ww *wav.Encoder
89                  wBuf = new(audio.IntBuffer)
90                  outputFileName string
91                  outputPath_1 string
92          )
93
94          wBuf.Format = &audio.Format{
95                  NumChannels: ch1,
96                  SampleRate: fs1,
97          }
98          wBuf.SourceBitDepth = bitDepth1
99
100         maxAmp := floats.Max(goresearch.AbsFloat64s(ampOut))
101         if maxAmp > math.MaxInt16+1 {
102                 reductionRate := math.MaxInt16 / maxAmp
103                 for i, _ := range ampOut {
104                         ampOut[i] *= reductionRate
105                 }
106         }
107
108         wBuf.Data = goresearch.Float64sToInts(ampOut)
109
110         outputFileName, _ = goresearch.SplitPathAndExt(
                outputFilePath)
111
112         outputPath_1 = outputFileName + ".wav"
113         fw, err = os.Create(outputPath_1)
114         check(err)
115         ww = wav.NewEncoder(fw, fs1, bitDepth1, ch1, 1)
116         err = ww.Write(wBuf)
117         check(err)
```

```
118         err = ww.Close()
119         check(err)
120
121         fmt.Printf("\nSuccessfully calcurated convolution! \n")
122 }
123
124 func check(err error) {
125         if err != nil {
126                 panic(err)
127         }
128 }
```

ソースコード 1.2.1-7 cmd/convolve_wav_fast/main.go

```
1  package main
2
3  import (
4          "flag"
5          "fmt"
6          "os"
7          "path/filepath"
8
9          "github.com/tetsuzawa/research-tools/goresearch"
10 )
11
12 func main() {
13         var (
14                 wavPath string
15                 coefPath string
16                 dataDir string
17         )
18
19         flag.StringVar(&wavPath, "wav", "", "wav path")
20         flag.StringVar(&coefPath, "coef", "", "coefficients path (.
               wav or .csv)")
21         flag.StringVar(&dataDir, "dir", "./", "save dir")
22
23         flag.Parse()
24
25         if wavPath == "" {
26                 fmt.Printf("please specify wav path\n\n")
27                 flag.Usage()
28                 os.Exit(1)
29         }
30
31         if coefPath == "" {
32                 fmt.Printf("please specify coef path\n\n")
33                 flag.Usage()
```

```
34                 os.Exit(1)
35         }
36
37         name, ext := goresearch.SplitPathAndExt(wavPath)
38         if ext != ".wav" {
39                 fmt.Printf("please specify wav path\n\n")
40                 flag.Usage()
41                 os.Exit(1)
42         }
43
44         fmt.Println("wavPath:", wavPath)
45         fmt.Println("coefPath:", coefPath)
46         fmt.Println("dataDir:", dataDir)
47
48         wData := goresearch.ReadDataFromWav(wavPath)
49
50         _, cExt := goresearch.SplitPathAndExt(coefPath)
51         var cData []float64
52         switch cExt {
53         case ".wav":
54                 cData = goresearch.IntsToFloat64s(goresearch.
                       ReadDataFromWav(coefPath))
55         case ".csv":
56                 cData = goresearch.ReadCoefFromCSV(coefPath)
57         default:
58                 fmt.Printf("file type is not valid. coefficients
                       file name:%v", coefPath)
59                 os.Exit(1)
60         }
61
62         wDataF := goresearch.IntsToFloat64s(wData)
63         convData := goresearch.FastConvolve(wDataF, cData)
64
65         outputName := filepath.Base(name) + "_convoluted"
66
67         data := goresearch.NormToMaxInt16(convData)
68
69         goresearch.SaveDataAsWav(data, dataDir, outputName)
70
71 }
```

ソースコード 1.2.1-8 cmd/convolve_wav_coef/main.go

```
1 package main
2
3 import (
4         "flag"
5         "fmt"
```

```
6         "os"
7         "path/filepath"
8
9         "github.com/tetsuzawa/research-tools/goresearch"
10 )
11
12 func main() {
13         var (
14                 wavPath string
15                 coefPath string
16                 dataDir string
17         )
18
19         flag.StringVar(&wavPath, "wav", "", "wav path")
20         flag.StringVar(&coefPath, "coef", "", "coefficients path (.
               wav or .csv)")
21         flag.StringVar(&dataDir, "dir", "./", "save dir")
22
23         flag.Parse()
24
25         if wavPath == "" {
26                 fmt.Printf("please specify wav path\n\n")
27                 flag.Usage()
28                 os.Exit(1)
29         }
30
31         if coefPath == "" {
32                 fmt.Printf("please specify coef path\n\n")
33                 flag.Usage()
34                 os.Exit(1)
35         }
36
37         name, ext := goresearch.SplitPathAndExt(wavPath)
38         if ext != ".wav" {
39                 fmt.Printf("please specify wav path\n\n")
40                 flag.Usage()
41                 os.Exit(1)
42         }
43
44         fmt.Println("wavPath:", wavPath)
45         fmt.Println("coefPath:", coefPath)
46         fmt.Println("dataDir:", dataDir)
47
48         wData := goresearch.ReadDataFromWav(wavPath)
49
50         _, cExt := goresearch.SplitPathAndExt(coefPath)
51         var cData []float64
```

```
52          switch cExt {
53          case ".wav":
54                  cData = goresearch.IntsToFloat64s(goresearch.
                        ReadDataFromWav(coefPath))
55          case ".csv":
56                  cData = goresearch.ReadCoefFromCSV(coefPath)
57          default:
58                  fmt.Printf("file type is not valid. coefficients
                        file name:%v", coefPath)
59                  os.Exit(1)
60          }
61
62          wDataF := goresearch.IntsToFloat64s(wData)
63          convData := goresearch.FastConvolve(wDataF, cData)
64
65          outputName := filepath.Base(name) + "_convoluted"
66
67          data := goresearch.NormToMaxInt16(convData)
68
69          goresearch.SaveDataAsWav(data, dataDir, outputName)
70
71  }
```

ソースコード 1.2.1-9 cmd/csv_to_wav/main.go

```
1   package main
2
3   import (
4       "flag"
5       "fmt"
6       "os"
7       "path/filepath"
8
9       "github.com/tetsuzawa/research-tools/goresearch"
10  )
11
12  func main() {
13
14      var (
15          csvPath string
16          dataDir string
17      )
18
19      flag.StringVar(&csvPath, "path", "", "csv path")
20      flag.StringVar(&dataDir, "dir", "./", "save dir")
21
22      flag.Parse()
23
```

```
24          if csvPath == "" {
25              fmt.Printf("please specify csv path\n\n")
26              flag.Usage()
27              os.Exit(1)
28          }
29
30          name, ext := goresearch.SplitPathAndExt(csvPath)
31          if ext != ".csv" {
32              fmt.Printf("please specify csv path\n\n")
33              flag.Usage()
34              os.Exit(1)
35          }
36
37          fmt.Println("csvPath:", csvPath)
38          fmt.Println("dataDir:", dataDir)
39
40          _, _, e := goresearch.ReadDataFromCSV(csvPath)
41
42          outputName := filepath.Base(name)
43
44          data := goresearch.NormToMaxInt16(e)
45
46          goresearch.SaveDataAsWav(data, dataDir, outputName)
47      }
```

ソースコード 1.2.1-10 cmd/drone_json_generator/main.go

```
1  package main
2
3  import (
4      "encoding/json"
5      "flag"
6      "fmt"
7      "os"
8      "path/filepath"
9  )
10
11 type ADFConfig struct {
12     WavName string `json:"wav_name"`
13     AdfName string `json:"adf_name"`
14     Mu float64 `json:"mu"`
15     L int `json:"l"`
16     Order int `json:"order"`
17 }
18
19 func main() {
20     var (
21         wavName string
```

```
22          adfName string
23          L int
24          Mu float64
25          order int
26          dataDir string
27      )
28
29      flag.StringVar(&wavName, "wav", "../wavfiles/dr_static_20.wav",
            "wav name")
30      flag.StringVar(&adfName, "adf", "NLMS", "algorithm")
31      flag.Float64Var(&Mu, "mu", 1.0, "mu")
32      flag.IntVar(&L, "L", 256, "L")
33      flag.IntVar(&order, "order", 8, "order")
34      flag.StringVar(&dataDir, "dir", "./", "save dir")
35
36      flag.Parse()
37
38      fmt.Println("wavName:", wavName)
39      fmt.Println("adfName:", adfName)
40      fmt.Println("mu:", Mu)
41      fmt.Println("L:", L)
42      fmt.Println("order:", order)
43      fmt.Println("dataDir:", dataDir)
44
45      var testName string
46      applicationName := "static"
47
48      switch adfName {
49      case "LMS":
50          testName = fmt.Sprintf("%v_%v_L-%v", adfName,
                applicationName, L)
51      case "NLMS":
52          testName = fmt.Sprintf("%v_%v_L-%v", adfName,
                applicationName, L)
53      case "AP":
54          testName = fmt.Sprintf("%v_%v_L-%v_order-%v", adfName,
                applicationName, L, order)
55      case "RLS":
56          testName = fmt.Sprintf("%v_%v_L-%v", adfName,
                applicationName, L)
57      default:
58          err := fmt.Errorf("\nadfName is not valid:%v\n", adfName)
59          fmt.Println(err)
60          fmt.Printf("Failed!\n")
61          os.Exit(1)
62      }
63      fmt.Printf("testName: %v\n", testName)
```

```
64
65      var adf = &ADFConfig{
66          WavName: wavName,
67          AdfName: adfName,
68          Mu: Mu,
69          L: L,
70          Order: order,
71      }
72
73      outadfJSON, err := json.Marshal(adf)
74      check(err)
75      fw, err := os.Create(filepath.Join(dataDir, testName+".json"))
76      check(err)
77      defer fw.Close()
78      _, err = fw.Write(outadfJSON)
79      check(err)
80
81      fmt.Printf("json file saved at :%v\n", filepath.Join(dataDir,
            testName+".json"))
82
83  }
84
85  func check(err error) {
86      if err != nil {
87          panic(err)
88      }
89  }
```

ソースコード 1.2.1-11 cmd/calc_mse_csv/main.go

```
1   package main
2
3   import (
4           "flag"
5           "fmt"
6           "gonum.org/v1/gonum/floats"
7           "math"
8           "path/filepath"
9
10          "github.com/tetsuzawa/research-tools/goresearch"
11  )
12
13  func main() {
14          var tap int
15          flag.IntVar(&tap, "tap", 256, "mse taps")
16
17          flag.Parse()
18
```

```
19          fmt.Println("mse taps:", tap)
20
21          inputPath := flag.Arg(0)
22          fmt.Println("inputPath:", inputPath)
23
24          dataDir := flag.Arg(1)
25          fmt.Println("dataDir:", dataDir)
26
27          ds, ys, es := goresearch.ReadDataFromCSV(inputPath)
28
29          var mse = make([]float64, len(es)-tap)
30          var v float64
31          var err error
32          for i := 0; i < len(es)-tap; i++ {
33
34                  fmt.Printf("working... %d%%\r", (i+1)*100/(len(es
                        )-tap))
35
36                  v, err = goresearch.CalcMSE(es[i:i+tap], nil)
37                  check(err)
38                  mse[i] = 20 * math.Log10(v)
39          }
40          floats.AddConst(-1*floats.Max(mse), mse)
41
42          name, _ := goresearch.SplitPathAndExt(inputPath)
43          outputName := filepath.Base(name) + "_mse"
44
45          goresearch.SaveDataAsCSV(ds, ys, es, mse, dataDir,
                outputName)
46
47  }
48
49  func check(err error) {
50          if err != nil {
51                  panic(err)
52          }
53  }
```

ソースコード 1.2.1-12 cmd/multirecord/main.go

```
1       /*
2       Contents: multi channel record.
3           This program works as typical recording app with multi
                channels.
4           Output file format is .wav.
5           Please run ‘multirecord --help‘ for details.
6       Usage: multirecord (-c ch -r rate -b bits -o /path/to/out.wav)
             sec
```

```
7     Author: Tetsu Takizawa
8     E-mail: tt15219@tomakomai.kosen-ac.jp
9     LastUpdate: 2019/11/18
10    DateCreated : 2019/11/18
11    */
12    package main
13
14    import (
15        "os"
16
17        "github.com/urfave/cli"
18    )
19
20    func main() {
21        app := cli.NewApp()
22        defer app.Run(os.Args)
23        app.CustomAppHelpTemplate = HelpTemplate
24
25        app.Name = "multirecord"
26        app.Usage = `This app records sounds with multi channels
              and save as .wav file or .DSB files if --DSB is
              specified.`
27        app.Version = "0.1.2"
28
29        app.Action = multiRecord
30
31        app.Flags = []cli.Flag{
32            cli.IntFlag{
33                Name: "channel, c",
34                Value: 1,
35                Usage: "input number of channels",
36            },
37            cli.IntFlag{
38                Name: "bits, b",
39                Value: 16,
40                Usage: "number of bits per sample",
41            },
42            cli.IntFlag{
43                Name: "rate, r",
44                Value: 48000,
45                Usage: "number of sample rate",
46            },
47            cli.StringFlag{
48                Name: "outpath, o",
49                Value: "out_multirecord.wav",
50                Usage: "specify output path",
51            },
```

```
52              cli.BoolFlag{
53                  Name: "params, p",
54                  Usage: "trace import statements",
55              },
56              cli.BoolFlag{
57                  Name: "DSB, D",
58                  Usage: "make .DSB files",
59              },
60          }
61      }
```

ソースコード 1.2.1-13 cmd/multirecord/multirecord.go

```
1  /*
2  Contents: multi channel record.
3          This program works as typical recording app with multi
               channels.
4          Output file format is .wav.
5          Please run ‘multirecord --help‘ for details.
6  Usage: multirecord (-c ch -r rate -b bits -o /path/to/out.wav) sec
7  Author: Tetsu Takizawa
8  E-mail: tt15219@tomakomai.kosen-ac.jp
9  LastUpdate: 2019/11/18
10 DateCreated : 2019/11/18
11 */
12 package main
13
14 import (
15         "fmt"
16         "os"
17         "strconv"
18         "time"
19
20         "github.com/go-audio/audio"
21         "github.com/go-audio/wav"
22         "github.com/gordonklaus/portaudio"
23         "github.com/pkg/errors"
24         "github.com/urfave/cli"
25
26         "github.com/tetsuzawa/research-tools/goresearch"
27 )
28
29 const (
30         FramesPerBuffer = 1024
31         PCM = 1
32 )
33
34 var (
```

```
35          w1 *wav.Encoder
36          RecordSeconds float64
37          NumChannels int
38          SampleRate int
39          BitsPerSample int
40          NumSamplesToWrite int
41          NumWritten int
42          aBuf = new(audio.IntBuffer)
43          err error
44  )
45
46  func multiRecord(ctx *cli.Context) error {
47          // ************* check argument *************
48          if ctx.Args().Get(0) == "" {
49                  return cli.NewExitError(‘too few arguments. need
                        recording duration
50  Usage: multirecord (-c ch -r rate -b bits -o /path/to/out.wav)
        duration
51  ‘, 2)
52          }
53
54          // ************* validate ext *************
55          name, ext := goresearch.SplitPathAndExt(ctx.String("o"))
56          if ext != ".wav" && ext != "" {
57                  return cli.NewExitError(‘incorrect file format.
                        multirecord saves audio as .wav file.
58  Usage: multirecord -o /path/to/file.wav 5.0‘, 2)
59          }
60
61          // ************* validate parameter *************
62          RecordSeconds, err = strconv.ParseFloat(ctx.Args().Get(0),
                64)
63          if err != nil {
64                  err = errors.Wrap(err, "error occurred while
                        converting arg of recording time from string to
                        float64")
65                  return cli.NewExitError(err, 5)
66          }
67          NumChannels = ctx.Int("c")
68          SampleRate = ctx.Int("r")
69          BitsPerSample := ctx.Int("b")
70          if BitsPerSample != 16 {
71                  return cli.NewExitError(‘sorry, this app is only for
                         16 bits per sample for now‘, 99)
72          }
73          NumSamplesToWrite = int(RecordSeconds * float64(SampleRate
                ))
```

```
74
75          // ************* create output file *************
76          f1, err := os.Create(name + ".wav")
77          if err != nil {
78                  err = errors.Wrap(err, "internal error: error
                        occurred while creating output file")
79                  return cli.NewExitError(err, 5)
80          }
81          defer f1.Close()
82
83          w1 = wav.NewEncoder(f1, SampleRate, BitsPerSample,
                    NumChannels, PCM)
84          aBuf.Format = &audio.Format{
85                  NumChannels: NumChannels,
86                  SampleRate: SampleRate,
87          }
88          aBuf.SourceBitDepth = BitsPerSample
89
90          // ************* initialize portaudio*************
91          err = portaudio.Initialize()
92          defer portaudio.Terminate()
93          if err != nil {
94                  err = errors.Wrap(err, "internal error: error
                        occurred while initializing portaudio")
95                  return cli.NewExitError(err, 5)
96          }
97
98          h, err := portaudio.DefaultHostApi()
99          if err != nil {
100                 err = errors.Wrap(err, "internal error: error
                        occurred while searching host API")
101                 return cli.NewExitError(err, 5)
102         }
103         paParam := portaudio.LowLatencyParameters(h.
                    DefaultInputDevice, h.DefaultOutputDevice)
104         paParam.SampleRate = float64(SampleRate)
105         paParam.Input.Channels = NumChannels
106         paParam.Output.Channels = 1
107         paParam.FramesPerBuffer = FramesPerBuffer
108
109         stream, err := portaudio.OpenStream(paParam, callback)
110         if err != nil {
111                 err = errors.Wrap(err, "internal error: error
                        occurred while opening stream on portaudio")
112                 return cli.NewExitError(err, 5)
113         }
114         defer stream.Close()
```

```
115
116         // ************* print parameter *************
117         fmt.Printf("\nOutput File: \t‘%s‘\n", name+".wav")
118         fmt.Printf("Channels: \t%d \n", NumChannels)
119         fmt.Printf("Sample Rate: \t%d\n", SampleRate)
120         fmt.Printf("Precision: \t%d-bits\n", BitsPerSample)
121         fmt.Printf("Duration: \t%f [sec] = %d samples\n",
                RecordSeconds, NumSamplesToWrite)
122         fmt.Printf("Encoding: \t%d-bits Signed Integer PCM\n\n",
                BitsPerSample)
123
124         if ctx.Bool("p") {
125                 fmt.Printf("\nFrames Per Buffer\n", paParam.
                        FramesPerBuffer)
126
127                 fmt.Printf("\nInput Device Parameters\n")
128                 fmt.Printf("Input Device Name\t\t\t%v\n", paParam.
                        Input.Device.Name)
129                 fmt.Printf("Input Device MaxInputChannels\t\t%v\n",
                         paParam.Input.Device.MaxInputChannels)
130                 fmt.Printf("Input Device DefaultSampleRate\t\t%v\n
                        ", paParam.Input.Device.DefaultSampleRate)
131                 fmt.Printf("Input Device HostApi\t\t\t%v\n",
                        paParam.Input.Device.HostApi)
132                 fmt.Printf("Input Device DefaultLowInputLatency\t%v\
                        n", paParam.Input.Device.DefaultLowInputLatency)
133                 fmt.Printf("Input Device DefaultHighInputLatency\t%v
                        \n", paParam.Input.Device.
                        DefaultHighInputLatency)
134
135                 fmt.Printf("\n\nOutput params\n")
136                 fmt.Printf("Output Device Name\t\t\t%v\n", paParam.
                        Output.Device.Name)
137                 fmt.Printf("Output Device MaxOutputChannels\t\t%v\n
                        ", paParam.Output.Device.MaxOutputChannels)
138                 fmt.Printf("Output Device DefaultSampleRate\t\t%v\n
                        ", paParam.Output.Device.DefaultSampleRate)
139                 fmt.Printf("Output Device HostApi\t\t\t%v\n",
                        paParam.Output.Device.HostApi)
140                 fmt.Printf("Output Device DefaultLowOutputLatency\t%
                        v\n", paParam.Output.Device.
                        DefaultLowOutputLatency)
141                 fmt.Printf("Output Device DefaultHighOutputLatency\t
                        %v\n", paParam.Output.Device.
                        DefaultHighOutputLatency)
142         }
143
```

```
144         // ************* record audio *************
145         err = stream.Start()
146         if err != nil {
147                 err = errors.Wrap(err, "internal error: error
                        occurred while starting stream")
148                 return cli.NewExitError(err, 5)
149         }
150
151         fmt.Printf("\nrecording...\n")
152
153         st := time.Now()
154         for time.Since(st).Seconds() < RecordSeconds {
155                 fmt.Printf("%.1f[sec] : %.1f[sec]\r", time.Since(st
                        ).Seconds(), RecordSeconds)
156         }
157         err = stream.Stop()
158         if err != nil {
159                 err = errors.Wrap(err, "internal error: error
                        occurred while stopping stream")
160                 return cli.NewExitError(err, 5)
161         }
162         err = w1.Close()
163         check(err)
164
165         fmt.Printf("\n\nSuccessfully recorded!!\n")
166
167         if ctx.Bool("D") {
168                 err = wavToDSB(ctx, name)
169                 if err != nil {
170                         err = errors.Wrap(err, "error occurred while
                                converting .wav file to .DSB files")
171                 }
172         }
173
174         return nil
175 }
176
177 // ************* callback function *************
178 func callback(inBuf, outBuf []int16) {
179         if NumWritten+FramesPerBuffer > NumSamplesToWrite {
180                 numWrite := NumSamplesToWrite - NumWritten
181                 NumWritten += numWrite
182                 aBuf.Data = goresearch.Int16sToInts(inBuf[:numWrite
                        ])
183                 err = w1.Write(aBuf)
184                 check(err)
185                 return
```

```
186         }
187         NumWritten += len(inBuf) / NumChannels
188         aBuf.Data = goresearch.Int16sToInts(inBuf)
189         err = w1.Write(aBuf)
190         check(err)
191 }
```

ソースコード 1.2.1-14 cmd/multirecord/util.go

```
1  package main
2
3
4  func check(err error) {
5          if err != nil {
6                  panic(err)
7          }
8  }
9
10 var HelpTemplate = `NAME:
11    {{.Name}}{{if .Usage}} - {{.Usage}}{{end}}
12
13 USAGE:
14    {{if .UsageText}}{{.UsageText}}{{else}}{{.HelpName}} {{if .
          VisibleFlags}}[options]{{end}} {{if .ArgsUsage}}{{.ArgsUsage
          }}{{else}}[arguments...]{{end}}{{end}}{{if .Version}}{{if
          not .HideVersion}}
15
16 VERSION:
17    {{.Version}}{{end}}{{end}}{{if .Description}}
18
19 DESCRIPTION:
20    {{.Description}}{{end}}{{if len .Authors}}
21
22 AUTHOR{{with $length := len .Authors}}{{if ne 1 $length}}S{{end
      }}{{end}}:
23    {{range $index, $author := .Authors}}{{if $index}}
24    {{end}}{{$author}}{{end}}{{end}}{{if .VisibleCommands}}
25
26 OPTIONS:
27    {{range $index, $option := .VisibleFlags}}{{if $index}}
28    {{end}}{{$option}}{{end}}{{end}}`
```

ソースコード 1.2.1-15 cmd/multirecord/wav_to_DSB.go

```
1  /*
2  Contents: multi channel record.
3          This program works as typical recording app with multi
               channels.
```

```
4         Output file format is .wav.
5         Please run ‘multirecord --help‘ for details.
6 Usage: multirecord (-c ch -r rate -b bits -o /path/to/out.wav) sec
7 Author: Tetsu Takizawa
8 E-mail: tt15219@tomakomai.kosen-ac.jp
9 LastUpdate: 2019/11/18
10 DateCreated : 2019/11/18
11 */
12 package main
13
14 import (
15         "bytes"
16         "encoding/binary"
17         "fmt"
18         "os"
19
20         "github.com/go-audio/wav"
21         "github.com/pkg/errors"
22         "github.com/urfave/cli"
23 )
24
25 func wavToDSB(ctx *cli.Context, name string) error {
26
27         // ************** create output file *************
28         f, err := os.Open(name + ".wav")
29         if err != nil {
30                 err = errors.Wrap(err, "internal error: error
                      occurred while creating output file")
31                 return cli.NewExitError(err, 5)
32         }
33         defer f.Close()
34         w := wav.NewDecoder(f)
35         if err != nil {
36                 err = errors.Wrap(err, "error occurred while
                      processing .wav file")
37                 return cli.NewExitError(err, 3)
38         }
39
40         // ************** validate parameter *************
41         w.ReadInfo()
42         ch := int(w.NumChans)
43         byteRate := int(w.BitDepth/8) * ch
44         bps := byteRate / ch
45         fs := int(w.SampleRate)
46
47         if fs != 48000 || bps != 2 || w.WavAudioFormat != 1 {
48                 errMsg := fmt.Sprintf(‘audio format error: wav
```

```
                format must be as follows.
49 sample rate: want 48000 Hz, got %v Hz
50 sampling bit rate: want 16 bits per sample, got %v bits per sample
51 audio format: want 1 (PCM), got %v‘, fs, w.BitDepth, w.
       WavAudioFormat)
52              return cli.NewExitError(errMsg, 3)
53      }
54
55      aBuf, err := w.FullPCMBuffer()
56      if err != nil {
57              err = errors.Wrap(err, "error occurred while
                   processing .wav file")
58              return cli.NewExitError(err, 3)
59      }
60
61      if aBuf.SourceBitDepth != 16 {
62              err = errors.New("sampling bit rate is incorrect.
                   need 16 bits per sample")
63              err = errors.Wrap(err, "error occurred while
                   processing .wav file")
64              return cli.NewExitError(err, 3)
65      }
66
67      iter := aBuf.NumFrames()
68      wBuf := make([]byte, iter*bps)
69
70      // ************** split channel and write to .DSB files
           **************
71      var fw *os.File
72      for j := 0; j < ch; j++ {
73              if ch == 1 {
74                      fw, err = os.Create(fmt.Sprintf("%s.DSB",
                           name))
75              } else {
76                      fw, err = os.Create(fmt.Sprintf("%s_ch%d.DSB
                           ", name, j+1))
77              }
78              if err != nil {
79                      return cli.NewExitError(err, 3)
80              }
81              defer fw.Close()
82
83              for i := 0; i < iter; i++ {
84                      fmt.Printf("converting... %d%%\r", (i
                           +1)*100/iter)
85                      b := new(bytes.Buffer)
86                      //err = binary.Write(b, binary.LittleEndian,
```

```
                                value[ch*i+j])
87                          err = binary.Write(b, binary.LittleEndian,
                                int16(aBuf.Data[ch*i+j]))
88                          if err != nil {
89                                  err = errors.Wrap(err, "internal
                                        error: error occurred while
                                        writing data to buffer")
90                                  return cli.NewExitError(err, 5)
91                          }
92                          copy(wBuf[bps*i:bps*(i+1)], b.Bytes())
93                  }
94                  _, err = fw.Write(wBuf)
95                  if err != nil {
96                          err = errors.Wrap(err, "error occurred while
                                writing data to .DSB file")
97                          return cli.NewExitError(err, 3)
98                  }
99
100         }
101         if ch == 1 {
102                 fmt.Printf("\n\n%d file created as `%s.DSB`\n", ch,
                        name)
103         } else {
104                 fmt.Printf("\n\n%d files created as `%s_chX.DSB`\n
                        ", ch, name)
105         }
106
107         fmt.Printf("\n")
108         fmt.Println("end!!")
109
110         return nil
111 }
```

ソースコード 1.2.1-16 cmd/wav_to_DSB/main.go

```
1  /*
2  Contents: wav to DSB converter.
3          This program converts .wav file to .DSB files.
4          Please run `wav_to_DSB --help` for details.
5  Usage: wav_to_DSB (-o /path/to/out.DSB) /path/to/file.wav
6  Author: Tetsu Takizawa
7  E-mail: tt15219@tomakomai.kosen-ac.jp
8  LastUpdate: 2019/11/16
9  DateCreated : 2019/11/16
10 */
11 package main
12
13 import (
```

```
14          "os"
15
16          "github.com/pkg/errors"
17          "github.com/urfave/cli"
18
19          "github.com/tetsuzawa/research-tools/goresearch"
20  )
21
22  func main() {
23          app := cli.NewApp()
24          defer app.Run(os.Args)
25          app.CustomAppHelpTemplate = HelpTemplate
26
27          app.Name = "wav_to_DSB"
28          app.Usage = `This app converts .wav file to .DSB file.`
29          app.Version = "0.1.0"
30
31          app.Action = action
32
33          app.Flags = []cli.Flag{
34                  cli.StringFlag{
35                          Name: "outpath, o",
36                          Usage: "specify output path like as /path/to
                                /file",
37                  },
38          }
39  }
40
41  func action(ctx *cli.Context) error {
42          if ctx.Args().Get(0) == "" {
43                  return cli.NewExitError("too few arguments. need
                        input file path. \nUsage: wav-to-DSB-multi /path
                        /to/file.wav", 2)
44          }
45
46          fileName := ctx.Args().Get(0)
47          name, ext := goresearch.SplitPathAndExt(fileName)
48
49          if ext != ".wav" {
50                  return cli.NewExitError("incorrect file format. need
                         .wav file. \nUsage: wav-to-DSB-multi /path/to/
                        file.wav", 2)
51          }
52
53          if ctx.String("o") != "" {
54                  argName := ctx.String("o")
55                  name, _ = goresearch.SplitPathAndExt(argName)
```

```
56          }
57
58          f, err := os.Open(fileName)
59          if err != nil {
60                  err = errors.Wrap(err, "error occurred while
                        opening input file")
61                  return cli.NewExitError("no such a file", 2)
62          }
63          defer f.Close()
64
65          return wavToDSB(ctx, f, name)
66  }
```

ソースコード 1.2.1-17 cmd/wav_to_DSB/util.go

```
1   package main
2
3   var HelpTemplate = `NAME:
4       {{.Name}}{{if .Usage}} - {{.Usage}}{{end}}
5
6   USAGE:
7       {{if .UsageText}}{{.UsageText}}{{else}}{{.HelpName}} {{if .
            VisibleFlags}}[options]{{end}} {{if .ArgsUsage}}{{.
            ArgsUsage}}{{else}}[arguments...]{{end}}{{end}}{{if .
            Version}}{{if not .HideVersion}}
8
9   VERSION:
10      {{.Version}}{{end}}{{end}}{{if .Description}}
11
12  DESCRIPTION:
13      {{.Description}}{{end}}{{if len .Authors}}
14
15  AUTHOR{{with $length := len .Authors}}{{if ne 1 $length}}S{{end
        }}{{end}}:
16      {{range $index, $author := .Authors}}{{if $index}}
17      {{end}}{{$author}}{{end}}{{end}}{{if .VisibleCommands}}
18
19  OPTIONS:
20      {{range $index, $option := .VisibleFlags}}{{if $index}}
21      {{end}}{{$option}}{{end}}{{end}}`
```

ソースコード 1.2.1-18 cmd/wav_to_DSB/wtod.go

```
1   package main
2
3   import (
4           "bytes"
5           "encoding/binary"
```

```
6         "fmt"
7         "github.com/go-audio/wav"
8         "os"
9
10        "github.com/pkg/errors"
11        "github.com/urfave/cli"
12 )
13
14 var err error
15
16 func wavToDSB(ctx *cli.Context, f *os.File, name string) error {
17        //w, err := wav.NewReader(f)
18        w := wav.NewDecoder(f)
19        if err != nil {
20                err = errors.Wrap(err, "error occurred while
                      processing .wav file")
21                return cli.NewExitError(err, 3)
22        }
23
24        w.ReadInfo()
25        ch := int(w.NumChans)
26        byteRate := int(w.BitDepth/8) * ch
27        bps := byteRate / ch
28        fs := int(w.SampleRate)
29
30        if fs != 48000 || bps != 2 || w.WavAudioFormat != 1 {
31                errMsg := fmt.Sprintf(‘audio format error: wav
                      format must be as follows.
32 sample rate: want 48000 Hz, got %v Hz
33 sampling bit rate: want 16 bits per sample, got %v bits per sample
34 audio format: want 1 (PCM), got %v‘, fs, w.BitDepth, w.
       WavAudioFormat)
35                return cli.NewExitError(errMsg, 3)
36        }
37
38        //data, err := w.ReadSamples(int(w.GetSubChunkSize()) /
              byteRate * ch)
39        aBuf, err := w.FullPCMBuffer()
40        if err != nil {
41                err = errors.Wrap(err, "error occurred while
                      processing .wav file")
42                return cli.NewExitError(err, 3)
43        }
44
45        //value, ok := data.([]int16)
46        if aBuf.SourceBitDepth != 16 {
47                err = errors.New("sampling bit rate is incorrect.
```

```
                need 16 bits per sample")
48              err = errors.Wrap(err, "error occurred while
                    processing .wav file")
49              return cli.NewExitError(err, 3)
50      }
51
52      //iter := len(value) / ch
53      iter := aBuf.NumFrames() / ch
54      wBuf := make([]byte, iter*bps)
55
56      var fw *os.File
57      for j := 0; j < ch; j++ {
58              if ch == 1 {
59                      fw, err = os.Create(fmt.Sprintf("%s.DSB",
                            name))
60              } else {
61                      fw, err = os.Create(fmt.Sprintf("%s_ch%d.DSB
                            ", name, j+1))
62              }
63              if err != nil {
64                      return cli.NewExitError(err, 3)
65              }
66              defer fw.Close()
67
68              for i := 0; i < iter; i++ {
69                      fmt.Printf("working... %d%%\r", (i+1)*100/
                            iter)
70                      b := new(bytes.Buffer)
71                      //err = binary.Write(b, binary.LittleEndian,
                             value[ch*i+j])
72                      err = binary.Write(b, binary.LittleEndian,
                            int16(aBuf.Data[ch*i+j]))
73                      if err != nil {
74                              err = errors.Wrap(err, "internal
                                    error: error occurred while
                                    writing data to buffer")
75                              return cli.NewExitError(err, 5)
76                      }
77                      copy(wBuf[bps*i:bps*(i+1)], b.Bytes())
78              }
79              _, err = fw.Write(wBuf)
80              if err != nil {
81                      err = errors.Wrap(err, "error occurred while
                             writing data to .DSB file")
82                      return cli.NewExitError(err, 3)
83              }
84
```

```
85          }
86          if ch == 1 {
87                  fmt.Printf("\n\n%d file created as %s.DSB\n", ch,
                        name)
88          } else {
89                  fmt.Printf("\n\n%d files created as %s_chX.DSB\n",
                        ch, name)
90          }
91
92          fmt.Printf("\n")
93          fmt.Println("end!!")
94
95          return nil
96  }
```

A.2.2 ADF ライブラリ

ソースコード 1.2.2-19 ディレクトリ構成

```
1  .
2  ├──── adf
3  │   ├──── ap.go
4  │   ├──── base.go
5  │   ├──── lms.go
6  │   ├──── nlms.go
7  │   └──── rls.go
8  ├──── fdadf
9  │   ├──── base.go
10 │   └──── fblms.go
11 └──── misc
12     └──── misc.go
13
14 # test コードや LISENCE ファイルなどは省略
```

ソースコード 1.2.2-20 ap.go

```
1  package adf
2
3  import (
4      "errors"
5      "gonum.org/v1/gonum/mat"
6  )
7
8  //FiltAP is base struct for AP filter.
9  //Use NewFiltAP to make instance.
10 type FiltAP struct {
```

```
11      filtBase
12      order int
13      eps float64
14      wHist [][]float64
15      xMem *mat.Dense
16      dMem *mat.Dense
17      yMem *mat.Dense
18      eMem *mat.Dense
19      epsIDE *mat.Dense
20      ide *mat.Dense
21  }
22
23  //NewFiltAP is constructor of AP filter.
24  //This func initialize filter length ‘n‘, update step size ‘mu‘,
        projection order ‘order‘ and filter weight ‘w‘.
25  func NewFiltAP(n int, mu float64, order int, eps float64, w []
        float64)(AdaptiveFilter, error) {
26      var err error
27      p := new(FiltAP)
28      p.kind = "AP filter"
29      p.n = n
30      p.muMin = 0
31      p.muMax = 1000
32      p.mu, err = p.checkFloatParam(mu, p.muMin, p.muMax, "mu")
33      if err != nil {
34          return nil, err
35      }
36      p.order = order
37      p.eps, err = p.checkFloatParam(eps, 0, 1000, "eps")
38      if err != nil {
39          return nil, err
40      }
41      err = p.initWeights(w, n)
42      if err != nil {
43          return nil, err
44      }
45      p.xMem = mat.NewDense(n, order, nil)
46      p.dMem = mat.NewDense(1, order, nil)
47
48      elmNum := order * order
49
50      //make diagonal matrix
51      diaMat := make([]float64, elmNum)
52      for i := 0; i < order; i++ {
53          diaMat[i*(order+1)] = eps
54      }
55      p.epsIDE = mat.NewDense(order, order, diaMat)
```

```
56
57      for i := 0; i < order; i++ {
58          diaMat[i*(order+1)] = 1
59      }
60      p.ide = mat.NewDense(order, order, diaMat)
61
62      p.yMem = mat.NewDense(order, 1, nil)
63      p.eMem = mat.NewDense(1, order, nil)
64
65      return p, nil
66  }
67
68  //Adapt calculates the error ‘e‘ between desired value ‘d‘ and
        estimated value ‘y‘,
69  //and update filter weights according to error ‘e‘.
70  func (af *FiltAP) Adapt(d float64, x []float64) {
71      xr, _ := af.xMem.Dims()
72      xCol := make([]float64, xr)
73      dr, _ := af.dMem.Dims()
74      dCol := make([]float64, dr)
75      // create input matrix and target vector
76      // shift column
77      for i := af.order - 1; i > 0; i-- {
78          mat.Col(xCol, i-1, af.xMem)
79          af.xMem.SetCol(i, xCol)
80          mat.Col(dCol, i-1, af.dMem)
81          af.dMem.SetCol(i, dCol)
82      }
83      af.xMem.SetCol(0, x)
84      af.dMem.Set(0, 0, d)
85
86      // estimate output and error
87      af.yMem.Mul(af.xMem.T(), af.w.T())
88      af.eMem.Sub(af.dMem, af.yMem.T())
89
90      // update
91      dw1 := mat.NewDense(af.order, af.order, nil)
92      dw1.Mul(af.xMem.T(), af.xMem)
93      dw1.Add(dw1, af.epsIDE)
94      dw2 := mat.NewDense(af.order, af.order, nil)
95      err := dw2.Solve(dw1, af.ide)
96      if err != nil {
97          panic(err)
98      }
99      dw3 := mat.NewDense(1, af.order, nil)
100     dw3.Mul(af.eMem, dw2)
101     dw := mat.NewDense(1, af.n, nil)
```

```
102     dw.Mul(dw3, af.xMem.T())
103     dw.Scale(af.mu, dw)
104     af.w.Add(af.w, dw)
105 }
106
107 //Run calculates the errors ‘e‘ between desired values ‘d‘ and
        estimated values ‘y‘ in a row,
108 //while updating filter weights according to error ‘e‘.
109 func (af *FiltAP) Run(d []float64, x [][]float64) (y []float64, e
        []float64, wHist [][]float64, err error) {
110     //measure the data and check if the dimension agree
111     N := len(x)
112     if len(d) != N {
113         return nil, nil, nil, errors.New("the length of slice d and
                x must agree")
114     }
115     af.n = len(x[0])
116     af.wHist = make([][]float64, N)
117     for i := 0; i < N; i++ {
118         af.wHist[i] = make([]float64, af.n)
119     }
120
121     y = make([]float64, N)
122     e = make([]float64, N)
123     w := af.w.RawRowView(0)
124
125     xr, _ := af.xMem.Dims()
126     xCol := make([]float64, xr)
127     dr, _ := af.dMem.Dims()
128     dCol := make([]float64, dr)
129
130     //adaptation loop
131     for i := 0; i < N; i++ {
132         copy(af.wHist[i], w)
133
134         // create input matrix and target vector
135         // shift column
136         for i := af.order - 1; i > 0; i-- {
137             mat.Col(xCol, i-1, af.xMem)
138             af.xMem.SetCol(i, xCol)
139             mat.Col(dCol, i-1, af.dMem)
140             af.dMem.SetCol(i, dCol)
141         }
142         af.xMem.SetCol(0, x[i])
143         af.dMem.Set(0, 0, d[i])
144
145         // estimate output and error
```

```
146         // same as af.yMem.Mul(af.xMem, af.w.T()).T()
147         af.yMem.Mul(af.xMem.T(), af.w.T())
148         af.eMem.Sub(af.dMem, af.yMem.T())
149         y[i] = af.yMem.At(0, 0)
150         e[i] = af.eMem.At(0, 0)
151 
152         // update
153         dw1 := mat.NewDense(af.order, af.order, nil)
154         dw1.Mul(af.xMem.T(), af.xMem)
155         dw1.Add(dw1, af.epsIDE)
156         dw2 := mat.NewDense(af.order, af.order, nil)
157         err := dw2.Solve(dw1, af.ide)
158         if err != nil {
159             return nil, nil, nil, err
160         }
161         dw3 := mat.NewDense(1, af.order, nil)
162         dw3.Mul(af.eMem, dw2)
163         dw := mat.NewDense(af.n, 1, nil)
164         dw.Mul(af.xMem, dw3.T())
165         dw.Scale(af.mu, dw)
166         af.w.Add(af.w, dw.T())
167     }
168     wHist = af.wHist
169     return y, e, wHist, nil
170 }
171 
172 func (af *FiltAP) clone() AdaptiveFilter {
173     altaf := *af
174     return &altaf
175 }
```

ソースコード 1.2.2-21 base.go

```
1  package adf
2  
3  import (
4          "fmt"
5  
6          "github.com/pkg/errors"
7          "gonum.org/v1/gonum/floats"
8          "gonum.org/v1/gonum/mat"
9  
10         "github.com/tetsuzawa/go-adflib/misc"
11 )
12 
13 // AdaptiveFilter is the basic Adaptive Filter interface type.
14 type AdaptiveFilter interface {
15         initWeights(w []float64, n int) error
```

```
16          //Predict calculates the new estimated value ‘y‘ from input
                slice ‘x‘.
17          Predict(x []float64) (y float64)
18
19          //Adapt calculates the error ‘e‘ between desired value ‘d‘
                and estimated value ‘y‘,
20          //and update filter weights according to error ‘e‘.
21          Adapt(d float64, x []float64)
22
23          //Run calculates the errors ‘e‘ between desired values ‘d‘
                and estimated values ‘y‘ in a row,
24          //while updating filter weights according to error ‘e‘.
25          Run(d []float64, x [][]float64) (y []float64, e []float64,
                wHist [][]float64, err error)
26          checkFloatParam(p, low, high float64, name string) (float64
                , error)
27          checkIntParam(p, low, high int, name string) (int, error)
28
29          //SetStepSize sets the step size of adaptive filter.
30          SetStepSize(mu float64) error
31
32          //GetParams returns the parameters at the time this func is
                called.
33          //parameters contains ‘n‘: filter length, ‘mu‘: filter
                update step size and ‘w‘: filter weights.
34          GetParams() (n int, mu float64, w []float64)
35
36          //GetParams returns the name of ADF.
37          GetKindName() (kind string)
38
39          clone() AdaptiveFilter
40  }
41
42  //Must checks whether err is nil or not. If err in not nil, this
        func causes panic.
43  func Must(af AdaptiveFilter, err error) AdaptiveFilter {
44          if err != nil {
45                  panic(err)
46          }
47          return af
48  }
49
50  //PreTrainedRun use part of the data for few epochs of learning.
51  //The arg ‘d‘ is desired values.
52  //‘x‘ is input matrix. columns are bunch of samples and rows are
        set of samples.
53  //‘nTrain‘ is train to test ratio, typical value is 0.5. (that
```

```
    means 50% of data is used for training).
54 //‘epochs‘ is number of training epochs, typical value is 1. This
    number describes how many times the training will be repeated.
55 func PreTrainedRun(af AdaptiveFilter, d []float64, x [][]float64,
    nTrain float64, epochs int) (y, e []float64, w [][]float64, err
     error) {
56         var nTrainI = int(float64(len(d)) * nTrain)
57         //train
58         for i := 0; i < epochs; i++ {
59                 _, _, _, err = af.Run(d[:nTrainI], x[:nTrainI])
60                 if err != nil {
61                         return nil, nil, nil, err
62                 }
63         }
64         //run
65         y, e, w, err = af.Run(d[:nTrainI], x[:nTrainI])
66         if err != nil {
67                 return nil, nil, nil, err
68         }
69         return y, e, w, nil
70 }
71
72 //ExploreLearning searches the ‘mu‘ with the smallest error value
    from the input matrix ‘x‘ and desired values ‘d‘.
73 //
74 //The arg ‘d‘ is desired value.
75 //
76 //‘x‘ is input matrix.
77 //
78 //‘muStart‘ is starting learning rate.
79 //
80 //‘muEnd‘ is final learning rate.
81 //
82 //‘steps‘ : how many learning rates should be tested between ‘
    muStart‘ and ‘muEnd‘.
83 //
84 //‘nTrain‘ is train to test ratio, typical value is 0.5. (that
    means 50% of data is used for training)
85 //
86 //‘epochs‘ is number of training epochs, typical value is 1. This
    number describes how many times the training will be repeated.
87 //
88 //‘criteria‘ is how should be measured the mean error. Available
    values are "MSE", "MAE" and "RMSE".
89 //
90 //‘target_w‘ is target weights. If the slice is nil, the mean
    error is estimated from prediction error.
```

```
91 //
92 // If an slice is provided, the error between weights and ‘target_w
       ‘ is used.
93 func ExploreLearning(af AdaptiveFilter, d []float64, x [][]float64,
       muStart, muEnd float64, steps int,
94         nTrain float64, epochs int, criteria string, targetW []
              float64) ([]float64, []float64, error) {
95         mus := misc.LinSpace(muStart, muEnd, steps)
96         es := make([]float64, len(mus))
97         zeros := make([]float64, int(float64(len(x))*nTrain))
98         for i, mu := range mus {
99                 //init
100                err := af.initWeights(nil, len(x[0]))
101                if err != nil {
102                        return nil, nil, errors.Wrap(err, "failed to
                             init weights at InitWights()")
103                }
104                err = af.SetStepSize(mu)
105                if err != nil {
106                        return nil, nil, errors.Wrap(err, "failed to
                             set step size at StetStepSize()")
107                }
108                //run
109                _, e, _, err := PreTrainedRun(af, d, x, nTrain,
                      epochs)
110                if err != nil {
111                        return nil, nil, errors.Wrap(err, "failed to
                             pre train at PreTrainedRun()")
112                }
113                es[i], err = misc.GetMeanError(e, zeros, criteria)
114                //fmt.Println(es[i])
115                if err != nil {
116                        return nil, nil, errors.Wrap(err, "failed to
                             get mean error at GetMeanError()")
117                }
118        }
119        return es, mus, nil
120 }
121
122 //FiltBase is base struct for adaptive filter structs.
123 //It puts together some functions used by all adaptive filters.
124 type filtBase struct {
125        kind string
126        n int
127        muMin float64
128        muMax float64
129        mu float64
```

```
130         w *mat.Dense
131 }
132
133 //NewFiltBase is constructor of base adaptive filter only for
        development.
134 func newFiltBase(n int, mu float64, w []float64) (AdaptiveFilter,
        error) {
135         var err error
136         p := new(filtBase)
137         p.kind = "Base filter"
138         p.n = n
139         p.muMin = 0
140         p.muMax = 1000
141         p.mu, err = p.checkFloatParam(mu, p.muMin, p.muMax, "mu")
142         if err != nil {
143                 return nil, err
144         }
145         err = p.initWeights(w, n)
146         if err != nil {
147                 return nil, err
148         }
149         return p, nil
150 }
151
152 //initWeights initialises the adaptive weights of the filter.
153 //The arg ‘w‘ is initial weights of filter.
154 //Typical value is zeros with length ‘n‘.
155 //If ‘w‘ is nil, this func initializes ‘w‘ as zeros.
156 //‘n‘ is size of filter. Note that it is often mistaken for the
        sample length.
157 func (af *filtBase) initWeights(w []float64, n int) error {
158         if n <= 0 {
159                 n = af.n
160         }
161         if w == nil {
162                 w = make([]float64, n)
163         }
164         if len(w) != n {
165                 return fmt.Errorf("the length of slice ‘w‘ and ‘n‘
                        must agree. len(w): %d, n: %d", len(w), n)
166         }
167         af.w = mat.NewDense(1, n, w)
168
169         return nil
170 }
171
172 //Predict calculates the new estimated value ‘y‘ from input slice ‘
```

```
    x‘.
func (af *filtBase) Predict(x []float64) (y float64) {
        y = floats.Dot(af.w.RawRowView(0), x)
        return y
}

//Adapt is just a method to satisfy the interface.
//It is used by overriding.
func (af *filtBase) Adapt(d float64, x []float64) {
        //TODO
}

//Run is just a method to satisfy the interface.
//It is used by overriding.
func (af *filtBase) Run(d []float64, x [][]float64) ([]float64, []
    float64, [][]float64, error) {
        //TODO
        //measure the data and check if the dimension agree
        N := len(x)
        if len(d) != N {
                return nil, nil, nil, errors.New("the length of
                    slice d and x must agree")
        }
        af.n = len(x[0])

        y := make([]float64, N)
        e := make([]float64, N)
        w := make([]float64, af.n)
        wHist := make([][]float64, N)
        //adaptation loop
        for i := 0; i < N; i++ {
                w = af.w.RawRowView(0)
                y[i] = floats.Dot(w, x[i])
                e[i] = d[i] - y[i]
                copy(wHist[i], w)
        }
        return y, e, wHist, nil
}

//checkFloatParam check if the value of the given parameter
//is in the given range and a float.
func (af *filtBase) checkFloatParam(p, low, high float64, name
    string) (float64, error) {
        if low <= p && p <= high {
                return p, nil
        } else {
                err := fmt.Errorf("parameter %v is not in range <%v
```

```
                    , %v>", name, low, high)
216                 return 0, err
217         }
218 }
219
220 //checkIntParam check if the value of the given parameter
221 //is in the given range and a int.
222 func (af *filtBase) checkIntParam(p, low, high int, name string) (
        int, error) {
223         if low <= p && p <= high {
224                 return p, nil
225         } else {
226                 err := fmt.Errorf("parameter %v is not in range <%v
                        , %v>", name, low, high)
227                 return 0, err
228         }
229 }
230
231 //SetStepSize set a update step size mu.
232 func (af *filtBase) SetStepSize(mu float64) error {
233         var err error
234         af.mu, err = af.checkFloatParam(mu, af.muMin, af.muMax, "mu
                ")
235         if err != nil {
236                 return err
237         }
238         return nil
239 }
240
241 //GetParams returns the parameters at the time this func is called.
242 //parameters contains `n`: filter length, `mu`: filter update step
        size and `w`: filter weights.
243 func (af *filtBase) GetParams() (int, float64, []float64) {
244         return af.n, af.mu, af.w.RawRowView(0)
245 }
246
247 //GetParams returns the name of ADF.
248 func (af *filtBase) GetKindName() string {
249         return af.kind
250 }
251
252 func (af *filtBase) clone() AdaptiveFilter {
253         altaf := *af
254         return &altaf
255 }
```

ソースコード 1.2.2-22 lms.go

```
1  package adf
2
3  import (
4          "errors"
5
6          "gonum.org/v1/gonum/floats"
7  )
8
9  //FiltLMS is base struct for LMS filter.
10 //Use NewFiltLMS to make instance.
11 type FiltLMS struct {
12         filtBase
13         wHistory [][]float64
14 }
15
16 //NewFiltLMS is constructor of LMS filter.
17 //This func initialize filter length ‘n‘, update step size ‘mu‘
       and filter weight ‘w‘.
18 func NewFiltLMS(n int, mu float64, w []float64) (AdaptiveFilter,
       error) {
19         var err error
20         p := new(FiltLMS)
21         p.kind = "LMS filter"
22         p.n = n
23         p.muMin = 0
24         p.muMax = 2
25         p.mu, err = p.checkFloatParam(mu, p.muMin, p.muMax, "mu")
26         if err != nil {
27                 return nil, err
28         }
29         err = p.initWeights(w, n)
30         if err != nil {
31                 return nil, err
32         }
33         return p, nil
34 }
35
36 //Adapt calculates the error ‘e‘ between desired value ‘d‘ and
       estimated value ‘y‘,
37 //and update filter weights according to error ‘e‘.
38 func (af *FiltLMS) Adapt(d float64, x []float64) {
39         w := af.w.RawRowView(0)
40         y := floats.Dot(w, x)
41         e := d - y
42         for i := 0; i < len(x); i++ {
43                 w[i] += af.mu * e * x[i]
44         }
```

```
45 }
46
47 //Run calculates the errors ‘e‘ between desired values ‘d‘ and
       estimated values ‘y‘ in a row,
48 //while updating filter weights according to error ‘e‘.
49 func (af *FiltLMS) Run(d []float64, x [][]float64) (y []float64, e
       []float64, wHist [][]float64, err error) {
50         //measure the data and check if the dimension agree
51         N := len(x)
52         if len(d) != N {
53                 return nil, nil, nil, errors.New("the length of
                       slice d and x must agree")
54         }
55         af.n = len(x[0])
56         af.wHistory = make([][]float64, N)
57         for i := 0; i < N; i++ {
58                 af.wHistory[i] = make([]float64, af.n)
59         }
60
61         y = make([]float64, N)
62         e = make([]float64, N)
63         //adaptation loop
64         for i := 0; i < N; i++ {
65                 w := af.w.RawRowView(0)
66                 copy(af.wHistory[i], w)
67                 y[i] = floats.Dot(w, x[i])
68                 e[i] = d[i] - y[i]
69                 for j := 0; j < af.n; j++ {
70                         w[j] += af.mu * e[i] * x[i][j]
71                 }
72         }
73         wHist = af.wHistory
74         return y, e, wHist, nil
75 }
76
77 func (af *FiltLMS) clone() AdaptiveFilter {
78         altaf := *af
79         return &altaf
80 }
```

ソースコード 1.2.2-23 nlms.go

```
1 package adf
2
3 import (
4         "errors"
5
6         "gonum.org/v1/gonum/floats"
```

```
)

//FiltNLMS is base struct for NLMS filter.
//Use NewFiltNLMS to make instance.
type FiltNLMS struct {
        filtBase
        eps float64
        wHistory [][]float64
}

//NewFiltLMS is constructor of LMS filter.
//This func initialize filter length ‘n‘, update step size ‘mu‘ and filter weight ‘w‘.
func NewFiltNLMS(n int, mu float64, eps float64, w []float64) (AdaptiveFilter, error) {
        var err error
        p := new(FiltNLMS)
        p.kind = "NLMS filter"
        p.n = n
        p.muMin = 0
        p.muMax = 2
        p.mu, err = p.checkFloatParam(mu, p.muMin, p.muMax, "mu")
        if err != nil {
                return nil, err
        }
        p.eps, err = p.checkFloatParam(eps, 0, 1, "eps")
        if err != nil {
                return nil, err
        }
        err = p.initWeights(w, n)
        if err != nil {
                return nil, err
        }
        return p, nil
}

//Adapt calculates the error ‘e‘ between desired value ‘d‘ and estimated value ‘y‘,
//and update filter weights according to error ‘e‘.
func (af *FiltNLMS) Adapt(d float64, x []float64) {
        w := af.w.RawRowView(0)
        y := floats.Dot(w, x)
        e := d - y
        nu := af.mu / (af.eps + floats.Dot(x, x))
        for i := 0; i < len(x); i++ {
                w[i] += nu * e * x[i]
        }
```

```
}

//Run calculates the errors ‘e‘ between desired values ‘d‘ and
    estimated values ‘y‘ in a row,
//while updating filter weights according to error ‘e‘.
func (af *FiltNLMS) Run(d []float64, x [][]float64) (y []float64,
    e []float64, wHist [][]float64, err error) {
        //measure the data and check if the dimension agree
        N := len(x)
        if len(d) != N {
                return nil, nil, nil, errors.New("the length of
                    slice d and x must agree")
        }
        af.n = len(x[0])
        af.wHistory = make([][]float64, N)
        for i := 0; i < N; i++ {
                af.wHistory[i] = make([]float64, af.n)
        }

        y = make([]float64, N)
        e = make([]float64, N)
        w := af.w.RawRowView(0)
        //adaptation loop
        for i := 0; i < N; i++ {
                copy(af.wHistory[i], w)
                y[i] = floats.Dot(w, x[i])
                e[i] = d[i] - y[i]
                nu := af.mu / (af.eps + floats.Dot(x[i], x[i]))
                for j := 0; j < af.n; j++ {
                        w[j] += nu * e[i] * x[i][j]
                }
        }
        wHist = af.wHistory
        return y, e, af.wHistory, nil
}

func (af *FiltNLMS) clone() AdaptiveFilter {
        altaf := *af
        return &altaf
}
```

ソースコード 1.2.2-24 rls.go

```
package adf

import (
        "errors"

```

```
6         "gonum.org/v1/gonum/floats"
7         "gonum.org/v1/gonum/mat"
8 )
9
10 //FiltRLS is base struct for RLS filter.
11 //Use NewFiltRLS to make instance.
12 type FiltRLS struct {
13         filtBase
14         wHist [][]float64
15         eps float64
16         rMat *mat.Dense
17 }
18
19 //NewFiltRLS is constructor of RLS filter.
20 //This func initialize filter length ‘n‘, update step size ‘mu‘,
       small enough value ‘eps‘, and filter weight ‘w‘.
21 func NewFiltRLS(n int, mu float64, eps float64, w []float64) (
       AdaptiveFilter, error) {
22         var err error
23         p := new(FiltRLS)
24         p.kind = "RLS filter"
25         p.n = n
26         p.muMin = 0
27         p.muMax = 1
28         p.mu, err = p.checkFloatParam(mu, p.muMin, p.muMax, "mu")
29         if err != nil {
30                 return nil, err
31         }
32         p.eps, err = p.checkFloatParam(mu, 0, 1, "eps")
33         if err != nil {
34                 return nil, err
35         }
36         err = p.initWeights(w, n)
37         if err != nil {
38                 return nil, err
39         }
40         var Rs = make([]float64, n*n)
41         for i := 0; i < n; i++ {
42                 Rs[i*(n+1)] = 1 / eps
43         }
44         p.rMat = mat.NewDense(n, n, Rs)
45         return p, nil
46 }
47
48 //Adapt calculates the error ‘e‘ between desired value ‘d‘ and
       estimated value ‘y‘,
49 //and update filter weights according to error ‘e‘.
```

```
50 func (af *FiltRLS) Adapt(d float64, x []float64) {
51         w := af.w.RawRowView(0)
52         R1 := mat.NewDense(af.n, af.n, nil)
53         var R2 float64
54         xVec := mat.NewDense(1, af.n, nil)
55         aux1 := mat.NewDense(af.n, 1, nil)
56         aux4 := mat.NewDense(1, af.n, nil)
57         var aux2 float64
58         aux3 := mat.NewDense(af.n, af.n, nil)
59         dwT := mat.NewDense(af.n, 1, nil)
60
61         y := floats.Dot(w, x)
62         e := d - y
63
64         xVec.SetRow(0, x)
65         aux1.Mul(af.rMat, xVec.T())
66         aux2 = floats.Dot(mat.Col(nil, 0, aux1), mat.Row(nil, 0,
               xVec))
67         R1 = mat.DenseCopyOf(af.rMat)
68         R1.Scale(aux2, R1)
69         aux4.Mul(xVec, af.rMat)
70
71         R2 = af.mu + mat.Dot(aux4.RowView(0), mat.DenseCopyOf(xVec.
               T()).ColView(0))
72         R1.Scale(1/R2, R1)
73         aux3.Sub(af.rMat, R1)
74         af.rMat.Scale(1/af.mu, aux3)
75         dwT.Mul(af.rMat, xVec.T())
76         dwT.Scale(e, dwT)
77
78         floats.Add(w, mat.Col(nil, 0, dwT))
79 }
80
81 //Run calculates the errors ‘e‘ between desired values ‘d‘ and
       estimated values ‘y‘ in a row,
82 //while updating filter weights according to error ‘e‘.
83 func (af *FiltRLS) Run(d []float64, x [][]float64) (y []float64, e
       []float64, wHist [][]float64, err error) {
84         //measure the data and check if the dimension agree
85         N := len(x)
86         if len(d) != N {
87                 return nil, nil, nil, errors.New("the length of
                       slice d and x must agree")
88         }
89         af.n = len(x[0])
90         af.wHist = make([][]float64, N)
91         for i := 0; i < N; i++ {
```

```
92                  af.wHist[i] = make([]float64, af.n)
93          }
94
95          y = make([]float64, N)
96          e = make([]float64, N)
97          w := af.w.RawRowView(0)
98          R1 := mat.NewDense(af.n, af.n, nil)
99          var R2 float64
100         xVec := mat.NewDense(1, af.n, nil)
101         aux1 := mat.NewDense(af.n, 1, nil)
102         aux4 := mat.NewDense(1, af.n, nil)
103         //aux2 := mat.NewDense(af.n, af.n, nil)
104         var aux2 float64
105         aux3 := mat.NewDense(af.n, af.n, nil)
106         dwT := mat.NewDense(af.n, 1, nil)
107         //adaptation loop
108         for i := 0; i < N; i++ {
109                 copy(af.wHist[i], w)
110                 y[i] = floats.Dot(w, x[i])
111                 e[i] = d[i] - y[i]
112                 xVec.SetRow(0, x[i])
113                 aux1.Mul(af.rMat, xVec.T())
114                 aux2 = floats.Dot(mat.Col(nil, 0, aux1), mat.Row(
                        nil, 0, xVec))
115                 R1 = mat.DenseCopyOf(af.rMat)
116                 R1.Scale(aux2, R1)
117                 //R1.Product(aux1.T(), xVec.T())
118                 aux4.Mul(xVec, af.rMat)
119                 R2 = af.mu + mat.Dot(aux4.RowView(0), mat.
                        DenseCopyOf(xVec.T()).ColView(0))
120                 //for j:=0;j<af.n;j++{
121                 // floats.AddConst(af.rMat.RawRowView(j))
122                 //}
123                 R1.Scale(1/R2, R1)
124                 aux3.Sub(af.rMat, R1)
125                 af.rMat.Scale(1/af.mu, aux3)
126                 dwT.Mul(af.rMat, xVec.T())
127                 dwT.Scale(e[i], dwT)
128                 floats.Add(w, mat.Col(nil, 0, dwT))
129         }
130         wHist = af.wHist
131         return y, e, af.wHist, nil
132 }
133
134 func (af *FiltRLS) clone() AdaptiveFilter {
135         altaf := *af
136         return &altaf
```

```
137 }
```

ソースコード 1.2.2-25 fdadf/base.go

```
1  package fdadf
2
3  import (
4      "fmt"
5
6      "github.com/pkg/errors"
7      "github.com/tetsuzawa/go-adflib/misc"
8      "gonum.org/v1/gonum/mat"
9  )
10
11 // FDAdaptiveFilter is the basic Frequency Domain Adaptive Filter
       interface type.
12 type FDAdaptiveFilter interface {
13     initWeights(w interface{}, n int) error
14
15     //Predict calculates the new estimated value ‘y‘ from input
           slice ‘x‘.
16     Predict(x []float64) (y []float64)
17
18     //Adapt calculates the error ‘e‘ between desired value ‘d‘ and
           estimated value ‘y‘,
19     //and update filter weights according to error ‘e‘.
20     Adapt(d []float64, x []float64)
21
22     //Run calculates the errors ‘e‘ between desired values ‘d‘ and
           estimated values ‘y‘ in a row,
23     //while updating filter weights according to error ‘e‘.
24     Run(d [][]float64, x [][]float64) ([][]float64, [][]float64,
           [][]float64, error)
25     checkFloatParam(p, low, high float64, name string) (float64,
           error)
26     checkIntParam(p, low, high int, name string) (int, error)
27     setStepSize(mu float64)
28
29     //GetParams returns the parameters at the time this func is
           called.
30     //parameters contains ‘n‘: filter length, ‘mu‘: filter update
           step size and ‘w‘: filter weights.
31     GetParams() (int, float64, []float64)
32
33     //GetParams returns the name of FDADF.
34     GetKindName() (kind string)
35 }
36
```

```
37 //Must checks whether err is nil or not. If err in not nil, this
       func causes panic.
38 func Must(af FDAdaptiveFilter, err error) FDAdaptiveFilter {
39     if err != nil {
40         panic(err)
41     }
42     return af
43 }
44
45 //PreTrainedRun use part of the data for few epochs of learning.
46 //The arg ‘d‘ is desired values. rows are
47 //‘x‘ is input matrix. rows are samples and columns are features.
48 //‘nTrain‘ is train to test ratio, typical value is 0.5. (that
       means 50% of data is used for training).
49 //‘epochs‘ is number of training epochs, typical value is 1. This
       number describes how many times the training will be repeated.
50 func PreTrainedRun(af FDAdaptiveFilter, d [][]float64, x [][]
       float64, nTrain float64, epochs int) (y, e [][]float64, w [][]
       float64, err error) {
51     var nTrainI = int(float64(len(d)) * nTrain)
52     //train
53     for i := 0; i < epochs; i++ {
54         _, _, _, err = af.Run(d[:nTrainI], x[:nTrainI])
55         if err != nil {
56             return nil, nil, nil, err
57         }
58     }
59     //run
60     y, e, w, err = af.Run(d[:nTrainI], x[:nTrainI])
61     if err != nil {
62         return nil, nil, nil, err
63     }
64     return y, e, w, nil
65 }
66
67 //ExploreLearning searches the ‘mu‘ with the smallest error value
       from the input matrix ‘x‘ and desired values ‘d‘.
68 //The arg ‘d‘ is desired value.
69 //‘x‘ is input matrix.
70 //‘muStart‘ is starting learning rate.
71 //‘muEnd‘ is final learning rate.
72 //‘steps‘ : how many learning rates should be tested between ‘
       muStart‘ and ‘muEnd‘.
73 //‘nTrain‘ is train to test ratio, typical value is 0.5. (that
       means 50% of data is used for training)
74 //‘epochs‘ is number of training epochs, typical value is 1. This
       number describes how many times the training will be repeated.
```

```
75 //‘criteria‘ is how should be measured the mean error. Available
       values are "MSE", "MAE" and "RMSE".
76 //‘target_w‘ is target weights. If the slice is nil, the mean
       error is estimated from prediction error.
77 // If an slice is provided, the error between weights and ‘target_w
       ‘ is used.
78 func ExploreLearning(af FDAdaptiveFilter, d [][]float64, x [][]
       float64, muStart, muEnd float64, steps int,
79     nTrain float64, epochs int, criteria string, targetW []float64)
           ([]float64, []float64, error) {
80     mus := misc.LinSpace(muStart, muEnd, steps)
81     es := make([]float64, len(mus))
82     zeros := make([]float64, int(float64(len(x))*nTrain))
83     ee := make([]float64, int(float64(len(x))*nTrain))
84     _, _, w := af.GetParams()
85     for i, mu := range mus {
86         //init
87         err := af.initWeights("zeros", len(w))
88         if err != nil {
89             return nil, nil, errors.Wrap(err, "failed to init
                   weights at InitWights()")
90         }
91         af.setStepSize(mu)
92         //run
93         _, e, _, err := PreTrainedRun(af, d, x, nTrain, epochs)
94         if err != nil {
95             return nil, nil, errors.Wrap(err, "failed to pre train
                   at PreTrainedRun()")
96         }
97         for i, sl := range e {
98             ee[i], err = misc.MSE(sl, make([]float64, len(sl)))
99         }
100        if err != nil {
101            return nil, nil, errors.Wrap(err, "failed to find MSE
                   of e at misc.MSE()")
102        }
103        es[i], err = misc.GetMeanError(ee, zeros, criteria)
104        //fmt.Println(es[i])
105        if err != nil {
106            return nil, nil, errors.Wrap(err, "failed to get mean
                   error at GetMeanError()")
107        }
108    }
109    return es, mus, nil
110 }
111
112 //filtBase is base struct for frequency domain adaptive filter
```

```
    structs
//It puts together some functions used by all adaptive filters.
type filtBase struct {
    kind string
    n int
    mu float64
    w *mat.Dense
}

//NewFiltBase is constructor of base frequency domain adaptive
    filter only for development.
func newFiltBase(n int, mu float64, w interface{}) (
    FDAdaptiveFilter, error) {
    var err error
    p := new(filtBase)
    p.n = n
    p.mu, err = p.checkFloatParam(mu, 0, 1000, "mu")
    if err != nil {
        return nil, err
    }
    err = p.initWeights(w, n)
    if err != nil {
        return nil, err
    }
    return p, nil
}

//initWeights initialises the adaptive weights of the filter.
//The arg `w` is initial weights of filter.
// Possible value "random": create random weights with stddev 0.5
    and mean is 0.
// "zeros": create zero value weights.
//`n` is size of filter. Note that it is often mistaken for the
    sample length.
func (af *filtBase) initWeights(w interface{}, n int) error {
    if n <= 0 {
        n = af.n
    }
    switch v := w.(type) {
    case string:
        if v == "random" {
            w := make([]float64, n)
            for i := 0; i < n; i++ {
                w[i] = misc.NewRandn(0.5, 0)
            }
            af.w = mat.NewDense(1, n, w)
        } else if v == "zeros" {
```

```
155             w := make([]float64, n)
156             af.w = mat.NewDense(1, n, w)
157         } else {
158             return errors.New("impossible to understand the w")
159         }
160     case []float64:
161         if len(v) != n {
162             return errors.New("length of w is different from n")
163         }
164         af.w = mat.NewDense(1, n, v)
165     default:
166         return errors.New(‘args w must be "random" or "zeros" or []
                float64{...}‘)
167     }
168     return nil
169 }
170
171 //Predict calculates the new output value ‘y‘ from input array ‘x
        ‘.
172 func (af *filtBase) Predict(x []float64) (y []float64) {
173     //TODO
174     //y = floats.Dot(af.w.RawRowView(0), x)
175     //return y
176     copy(y, x)
177     return
178 }
179
180 //Adapt is just a method to satisfy the interface.
181 //It is used by overriding.
182 func (af *filtBase) Adapt(d []float64, x []float64) {
183     //TODO
184 }
185
186 //Run is just a method to satisfy the interface.
187 //It is used by overriding.
188 func (af *filtBase) Run(d [][]float64, x [][]float64) ([][]float64
        , [][]float64, [][]float64, error) {
189     //TODO
190     return nil, nil, nil, nil
191 }
192
193 //checkFloatParam check if the value of the given parameter
194 //is in the given range and a float.
195 func (af *filtBase) checkFloatParam(p, low, high float64, name
        string) (float64, error) {
196     if low <= p && p <= high {
197         return p, nil
```

```
198     } else {
199         err := fmt.Errorf("parameter %v is not in range <%v, %v>",
                name, low, high)
200         return 0, err
201     }
202 }
203
204 //checkIntParam check if the value of the given parameter
205 //is in the given range and a int.
206 func (af *filtBase) checkIntParam(p, low, high int, name string) (
        int, error) {
207     if low <= p && p <= high {
208         return p, nil
209     } else {
210         err := fmt.Errorf("parameter %v is not in range <%v, %v>",
                name, low, high)
211         return 0, err
212     }
213 }
214
215 //setStepSize set a update step size mu.
216 func (af *filtBase) setStepSize(mu float64) {
217     af.mu = mu
218 }
219
220 //GetParams returns the parameters at the time this func is called.
221 //parameters contains `n`: filter length, `mu`: filter update step
        size and `w`: filter weights.
222 func (af *filtBase) GetParams() (n int, mu float64, w []float64) {
223     return af.n, af.mu, af.w.RawRowView(0)
224 }
225
226 //GetParams returns the kind name of ADF.
227 func (af *filtBase) GetKindName() (kind string) {
228     return af.kind
229 }
```

ソースコード 1.2.2-26 fblms.go

```
1 package fdadf
2
3 import (
4         "math/cmplx"
5
6         "github.com/mjibson/go-dsp/fft"
7         "github.com/pkg/errors"
8         "gonum.org/v1/gonum/mat"
9 )
```

```
10
11 //FiltFBLMS is base struct for FBLMS filter
12 //(Fast Block Least Mean Square filter).
13 //Use NewFiltFBLMS to make instance.
14 type FiltFBLMS struct {
15         filtBase
16         wHistory [][]float64
17         xMem *mat.Dense
18 }
19
20 //NewFiltFBLMS is constructor of FBLMS filter.
21 //This func initialize filter length ‘n‘, update step size ‘mu‘
       and filter weight ‘w‘.
22 func NewFiltFBLMS(n int, mu float64, w interface{}) (
       FDAdaptiveFilter, error) {
23         var err error
24         p := new(FiltFBLMS)
25         p.kind = "FBLMS filter"
26         p.n = n
27         p.mu, err = p.checkFloatParam(mu, 0, 1000, "mu")
28         if err != nil {
29                 return nil, errors.Wrap(err, "Parameter error at
                       checkFloatParam()")
30         }
31         err = p.initWeights(w, 2*n)
32         if err != nil {
33                 return nil, err
34         }
35         p.xMem = mat.NewDense(1, n, make([]float64, n))
36         return p, nil
37 }
38
39 //Adapt calculates the error ‘e‘ between desired value ‘d‘ and
       estimated value ‘y‘,
40 //and update filter weights according to error ‘e‘.
41 func (af *FiltFBLMS) Adapt(d []float64, x []float64) {
42         zeros := make([]float64, af.n)
43         Y := make([]complex128, 2*af.n)
44         y := make([]float64, af.n)
45         e := make([]float64, af.n)
46         EU := make([]complex128, 2*af.n)
47
48         w := af.w.RawRowView(0)
49         // 1 compute the output of the filter for the block kM,
               ..., KM + M -1
50         W := fft.FFT(float64sToComplex128s(append(w[:af.n], zeros
               ...)))
```

```
51          xSet := append(af.xMem.RawRowView(0), x...)
52          U := fft.FFT(float64sToComplex128s(xSet))
53          af.xMem.SetRow(0, x)
54          for i := 0; i < 2*af.n; i++ {
55                  Y[i] = W[i] * U[i]
56          }
57          yc := fft.IFFT(Y)[af.n:]
58          for i := 0; i < af.n; i++ {
59                  y[i] = real(yc[i])
60                  e[i] = d[i] - y[i]
61          }
62
63          // 2 compute the correlation vector
64          aux1 := fft.FFT(float64sToComplex128s(append(zeros, e...)))
65          aux2 := fft.FFT(float64sToComplex128s(xSet))
66          for i := 0; i < 2*af.n; i++ {
67                  EU[i] = aux1[i] * cmplx.Conj(aux2[i])
68          }
69          phi := fft.IFFT(EU)[:af.n]
70
71          // 3 update the parameters of the filter
72          aux1 = fft.FFT(float64sToComplex128s(append(w[:af.n], zeros
                ...)))
73          aux2 = fft.FFT(append(phi, float64sToComplex128s(zeros
                )...))
74          for i := 0; i < 2*af.n; i++ {
75                  W[i] = aux1[i] + complex(af.mu, 0)*aux2[i]
76          }
77          aux3 := fft.IFFT(W)
78          for i := 0; i < 2*af.n; i++ {
79                  w[i] = real(aux3[i])
80          }
81  }
82
83  //Predict calculates the new output value ‘y‘ from input array ‘x
        ‘.
84  func (af *FiltFBLMS) Predict(x []float64) (y []float64) {
85          zeros := make([]float64, af.n)
86          y = make([]float64, af.n)
87          Y := make([]complex128, 2*af.n)
88          W := fft.FFT(float64sToComplex128s(append(af.w.RawRowView
                (0)[:af.n], zeros...)))
89          U := fft.FFT(float64sToComplex128s(append(af.xMem.
                RawRowView(0), x...)))
90          for i := 0; i < 2*af.n; i++ {
91                  Y[i] = W[i] * U[i]
92          }
```

```
93          yc := fft.IFFT(Y)[af.n:]
94          for i := 0; i < af.n; i++ {
95                  y[i] = real(yc[i])
96          }
97          return
98  }
99
100 //Run calculates the errors ‘e‘ between desired values ‘d‘ and
        estimated values ‘y‘ in a row,
101 //while updating filter weights according to error ‘e‘.
102 //The arg ‘x‘: rows are samples sets, columns are input values.
103 func (af *FiltFBLMS) Run(d [][]float64, x [][]float64) ([][]
        float64, [][]float64, [][]float64, error) {
104         //measure the data and check if the dimension agree
105         N := len(x)
106         if len(d) != N {
107                 return nil, nil, nil, errors.New("the length of
                        slice d and x must agree")
108         }
109         af.n = len(x[0])
110         af.wHistory = make([][]float64, N)
111         for i := range af.wHistory {
112                 af.wHistory[i] = make([]float64, af.n)
113         }
114
115         zeros := make([]float64, af.n)
116         Y := make([]complex128, 2*af.n)
117         y := make([][]float64, N)
118         for i := range y {
119                 y[i] = make([]float64, af.n)
120         }
121         e := make([][]float64, N)
122         for i := range e {
123                 e[i] = make([]float64, af.n)
124         }
125
126         EU := make([]complex128, 2*af.n)
127
128         for k := 0; k < N; k++ {
129                 w := af.w.RawRowView(0)
130                 copy(af.wHistory[k], w)
131
132                 // 1 compute the output of the filter for the block
                        kM, ..., KM + M -1
133                 W := fft.FFT(float64sToComplex128s(append(w[:af.n],
                        zeros...)))
134                 xSet := append(af.xMem.RawRowView(0), x[k]...)
```

```
135                 U := fft.FFT(float64sToComplex128s(xSet))
136                 af.xMem.SetRow(0, x[k])
137
138                 for i := 0; i < 2*af.n; i++ {
139                         Y[i] = W[i] * U[i]
140                 }
141                 yc := fft.IFFT(Y)[af.n:]
142                 for i := 0; i < af.n; i++ {
143                         y[k][i] = real(yc[i])
144                         e[k][i] = x[k][i] - y[k][i]
145                 }
146
147                 // 2 compute the correlation vector
148                 aux1 := fft.FFT(float64sToComplex128s(append(zeros,
                        e[k]...)))
149                 aux2 := fft.FFT(float64sToComplex128s(xSet))
150                 for i := 0; i < 2*af.n; i++ {
151                         EU[i] = aux1[i] * cmplx.Conj(aux2[i])
152                 }
153                 phi := fft.IFFT(EU)[:af.n]
154
155                 // 3 update the parameters of the filter
156                 aux1 = fft.FFT(float64sToComplex128s(append(w[:af.n
                        ], zeros...)))
157                 aux2 = fft.FFT(append(phi, float64sToComplex128s(
                        zeros)...))
158                 for i := 0; i < 2*af.n; i++ {
159                         W[i] = aux1[i] + complex(af.mu, 0)*aux2[i]
160                 }
161                 aux3 := fft.IFFT(W)
162                 for i := 0; i < 2*af.n; i++ {
163                         w[i] = real(aux3[i])
164                 }
165         }
166
167         return y, e, af.wHistory, nil
168 }
```

ソースコード 1.2.2-27 misc.go

```
1  package misc
2
3  import (
4          "errors"
5          "math"
6          "math/rand"
7
8          "gonum.org/v1/gonum/floats"
```

```
)

func ElmAbs(fs []float64) []float64 {
	fsAbs := make([]float64, len(fs))
	for i, f := range fs {
		fsAbs[i] = math.Abs(f)
	}
	return fsAbs
}

func LogSE(x1, x2 []float64) ([]float64, error) {
	e, err := GetValidError(x1, x2)
	if err != nil {
		return nil, err
	}
	for i := 0; i < len(e); i++ {
		e[i] = 10 * math.Log10(math.Pow(e[i], 2))
	}
	return e, nil

}

func MAE(x1, x2 []float64) (float64, error) {
	e, err := GetValidError(x1, x2)
	if err != nil {
		return 0, err
	}
	return floats.Sum(ElmAbs(e)) / float64((len(e))), nil
}

func MSE(x1, x2 []float64) (float64, error) {
	e, err := GetValidError(x1, x2)
	if err != nil {
		return 0, err
	}
	return floats.Dot(e, e) / float64(len(e)), nil
}

func RMSE(x1, x2 []float64) (float64, error) {
	e, err := GetValidError(x1, x2)
	if err != nil {
		return 0, err
	}
	return math.Sqrt(floats.Dot(e, e)) / float64(len(e)), nil
}

func GetValidError(x1, x2 []float64) ([]float64, error) {
```

```
56          if len(x1) != len(x2) {
57                  err := errors.New("length of two slices is
                        different")
58                  return nil, err
59          }
60          floats.Sub(x1, x2)
61          e := x1
62          return e, nil
63  }
64
65  func GetMeanError(x1, x2 []float64, fn string) (float64, error) {
66          switch fn {
67          case "MAE":
68                  return MAE(x1, x2)
69          case "MSE":
70                  return MSE(x1, x2)
71          case "RMSE":
72                  return RMSE(x1, x2)
73          default:
74                  err := errors.New(‘The provided error function (fn)
                            is not known.
75                                                                  Use "
                                                                        MAE
                                                                        ",
                                                                         "
                                                                        MSE
                                                                        "
                                                                        or
                                                                         "
                                                                        RMSE
                                                                        "‘)

76                  return 0, err
77          }
78  }
79
80  // NewRandn returns random value. stddev 0.5, mean 0.
81  func NewRandn(stddev, mean float64) float64 {
82          return rand.NormFloat64()*stddev + mean
83  }
84
85  func LinSpace(start, end float64, n int) []float64 {
86          res := make([]float64, n)
87          if n == 1 {
88                  res[0] = end
89                  return res
90          }
```

```
91          delta := (end - start) / (float64(n) - 1)
92          for i := 0; i < n; i++ {
93                  res[i] = start + (delta * float64(i))
94          }
95          return res
96  }
97
98  func Floor(fs [][]float64) []float64 {
99          var fs1d = make([]float64, len(fs)*len(fs[0]))
100         for i, sl := range fs {
101                 for j, v := range sl {
102                         fs1d[len(fs[0])*i+j] = v
103                 }
104         }
105         return fs1d
106 }
107
108 func NewRandSlice(n int) []float64 {
109         rs := make([]float64, n)
110         for i := 0; i < n; i++ {
111                 rs[i] = rand.Float64()
112         }
113         return rs
114 }
115
116 func NewNormRandSlice(n int) []float64 {
117         rs := make([]float64, n)
118         for i := 0; i < n; i++ {
119                 rs[i] = rand.NormFloat64()
120         }
121         return rs
122 }
123
124 // NewRand2dSlice make 2d slice.
125 // the arg n is sample number and m is number of signals.
126 func NewRand2dSlice(n, m int) [][]float64 {
127         rs2 := make([][]float64, m)
128         for j := 0; j < m; j++ {
129                 rs2[j] = NewRandSlice(n)
130         }
131         return rs2
132 }
133
134 // NewRandNorm2dSlice make 2d slice.
135 // the arg n is sample number and m is number of signals.
136 func NewNormRand2dSlice(n, m int) [][]float64 {
137         rs2 := make([][]float64, m)
```

```
138         for j := 0; j < m; j++ {
139                 rs2[j] = NewNormRandSlice(n)
140         }
141         return rs2
142 }
143
144 func Unset(s []float64, i int) []float64 {
145         if i >= len(s) {
146                 return s
147         }
148         return append(s[:i], s[i+1:]...)
149 }
```