情報学群実験第 3C/3i 実験レポート 第 1 回

音波の特性と聴覚特性に関する実験

1250373　溝口 洸熙*
Group 10

May 7th, 2023

*高知工科大学 情報学群 情報セキュリティシステム研究室

概要

音は，計算機上で行列を用いて表現される．基本となる正弦波の振幅や位相と，音の関係について実験する．さらに，周波数の異なる音の加算合成で起こるうなり現象，統計的な雑音である白色ガウス雑音の実装も行う．

また，音は，さまざまな周期や振幅を持つ正弦波の重ね合わせで再現されていることを踏まえて，周波数成分を取り出して解析する周波数解析し，周波数解析の結果を用いて周波数を編集し波形を再構成する．

最後にヒトの聴覚特性を再現する．ヒトの聴覚特性である音脈分凝，連続聴効果またミッシングファンダメンタル波刺激を起こす音波を計算機上で作成し，その効果の再現する．効果の起こる条件や考えられる原因を考察する．

目次

図目次

表目次

ソースコード

第 1 章

音の工学的特徴

1.1　周波数の異なる純音の生成

1.1.1　実験の目的

我々は音の「高い」「低い」をどのようにして認識しているのだろうか．音が高いまたは低いと知覚するためには何かと比較するはずだがその比較の指標は何だろうか．この実験ではまず，行列上に正弦波を生成する．そして周波数の変化に対して正弦波グラフおよび音の違いを実験を通して確認し，考察する．

1.1.2　実験の方法と考え方

■実験に用いる装置　このレポート内すべての実験には MathWorks®社の MATLAB®を用いて，表 1-1 の環境下で実験する．

表 1-1: 実験環境

実験機	MacBook Air 2022（Apple 社）`MLY13J/A`
プロセッサ	Apple Silicon M2　8 コア CPU，8 コア GPU
メモリ	8GB
MATLAB®	R2023a - academic use (Update1 9.14.02239454) 64-bit (maci64) March 30, 2023

また，このレポートないすべての実験では MATLAB®でプロットしたグラフを出力するために，`exportgraphics` 関数を用いる（src.1-1）．

src. 1-1: グラフ出力

```
exportgraphics(figurename,'path/figure_name.pdf','ContentType','vecto');
```

■正弦波について　時刻 t に対して周波数 f の正弦波は，式 (1.1) で得られる．

$$y = \sin(2\pi f t) \tag{1.1}$$

時間軸データ t を 1 行 N 列のベクトルに代入する．時間軸データの作成について，サンプリング周波数 `Fs` に対して m 秒間の正弦波を生成するためには，src.1-2 のように `0` から `Fs` まのベクトルに対して，各要素をサンプリング周波数で割ると時間軸テーブルを作成できる．

src. 1-2: 時間軸作成と正弦波の作成

```
t = (0 : m*(Fs-1)) /Fs;
y = sin(2*pi*f*t);
```

t の各要素 t_n に対して三角関数 $\sin(2\pi f t_n)$ を演算し，ベクトル y の要素 y_n に代入する．したがって y も 1 行 N 列のベクトルになる．生成した正弦波を `plot` 関数を用いて y を t の関数として描画する．

このように，ただ一つの正弦波からなるような音を**純音**という [1, p.1]．また，サンプリング周波数を `Fs` とし，データ列 `y` を再生するためには `sound` 関数を用いる．

■実験内容　この実験では，周波数を $f_1 = 440\text{Hz}$，$f_2 = 660\text{Hz}$ の 2 種類を用いてそれぞれ正弦波 y_1，y_2 を生成する．生成した y_1，y_2 に対して，t を横軸に取りグラフを作成し，サンプリング周波数を `Fs` $= 16\text{kHz}$ として再生する．グラフを描画するときの横軸を時間としたとき，便宜的*に縦軸を `amplitude` とする．1.1 章のソースコードは，src.A-1. ►p.23

1.1.3　実験の結果

各正弦波のグラフを図 1-1 に示す．音を聴き比べた結果，f_2 の周波数を用いた正弦波は f_1 を用いた正弦波に比べて音が高かった．具体的には f_1 が A4 の音*†であるのに対して，f_2 の音は完全 5 度大きい E5 の音‡であった．さらに，聴音確認・目視確認では音の大きさ，音色，振幅や波形の変化は確認でなかった．

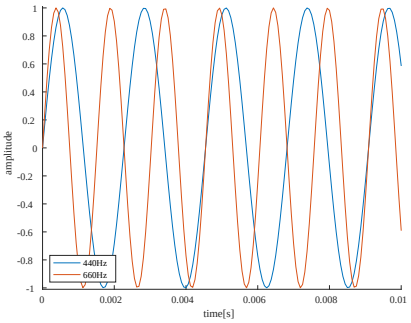


図 1-1: 周波数が異なる正弦波のグラフ

1.1.4　考察

周波数 (f) とは 1 秒間の振動回数であり，これが音の高さを決める．実験結果より周波数が大きい，つまり 1 秒間により多く振動すれば音は高くなることが分かった．また，1 回振動あたりの時間を周期 (T) と言い，周期と周波数は反比例の関係であり，式 (1.2) となる．

$$f = \frac{1}{T} \qquad \big(\text{無論}\quad T \neq 0,\quad f \neq 0\big) \tag{1.2}$$

図 1-1 より，周波数が大きい正弦波は周期が短く，逆もまた確認できる．周波数のみを変更したので，振幅や正弦波の波形変化はない．

また，周波数が f_1 正弦波の音と f_2 正弦波の音はそれぞれ純音だがこれらの間にも関係がある．数学的には $f_1 : f_2 = 2 : 3$ の簡単な整数比になっている．

> ヒトの耳は，2 つの音を同時に聞いた場合，その 2 つの音の周波数が簡単になっているほど協和して聴こえる．（略）
> 周波数比が整数比になっている音程を純正音程とよぶ．　[1, p.46 - p.47]

つまり，周波数 f_1 正弦波と周波数 f_2 正弦波の加算合成波は協和してきこえる．

1.2　振幅・位相の確認

1.2.1　実験の目的

音の大小は何によって決まるのだろうか．音の大小に関わる波の振幅や，波を構成するうえで重要な初期位相を変化させ，変化の前後での音の違いを聞き取り，ヒトの耳に初期位相の変化や振幅の変化がどのように感じるか実験を通して考察する．

1.2.2　実験の方法と考え方

時刻 t に対して周波数 f の正弦波は式 (1.1) で得られ，その初期位相（*initial phase*）を ϕ とすると，その正弦波は式 (1.3) となる．

$$y = \sin(2\pi f t + \phi) \tag{1.3}$$

また，式 (1.1) の振幅を A 倍して得られる正弦波は式 (1.4) となる．

$$y = A\sin(2\pi f t) \tag{1.4}$$

*振幅は振動の中心からの距離であり，正の値である．ゆえに負の値が存在するデータ対して `amplitude`（振幅）と記すのは本来不適切である．
*以降，音階はドイツ語音階で記す．
†イタリア語音階で「ラ」
‡イタリア語音階で「ミ」

■実験内容　振幅，周期，音の変化を確認する．周波数 $f = 440$Hz，サンプリング周波数を 16kHz とする．確認方法として，左から純音，右から実験変化させた音を再生するようにし（src.1-3），音の変化を確認する．実験内容は，表 1-2 に示す．1.2 章のソースコードは，src.A-2. ▶p.23

src. 1-3: 左右から別の音を出力

```
Fs = 16000; % サンプリング周波数
y1 = 音声データ1; % N行1列（左）
y2 = 音声データ2; % N行1列（右）
y = [y1 y2]; % N行2列　列結合を行う
sound(y, Fs);
```

1.2.3　実験の結果

聴音確認の結果を表 1-2，図 1-2 および図 1-3 に示す．振幅の変化に比例した音量変化を確認できた．初期位相の変化による，音の変化は確認できなかった．

表 1-2: 振幅・位相の確認 実験結果

実験対象	振幅（基準倍）	初期位相	生成される正弦波	純音との比較
純音	基準	基準	$y_0 = \sin(2\pi ft)$	—
振幅	0.5	0	$y_1 = 0.5 \times \sin(2\pi ft)$	音量が基準の 1/2 程度に聞こえた．
	0.25	0	$y_2 = 0.25 \times \sin(2\pi ft)$	音量が基準の 1/4 程度に聞こえた．
初期位相	1	$+\frac{\pi}{2}$	$y_3 = \sin(2\pi ft + \frac{\pi}{2})$	違いは分からなかった．
	1	$+\pi$	$y_4 = \sin(2\pi ft + \pi)$	違いは分からなかった．

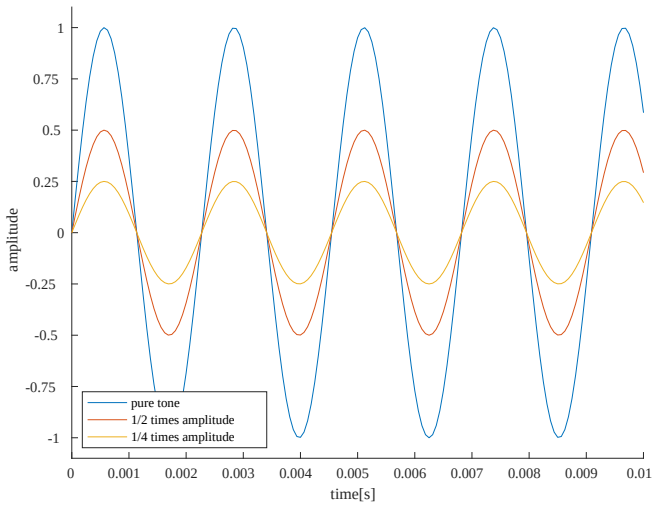


図 1-2: 振幅・位相の確認 実験結果（振幅の変化）

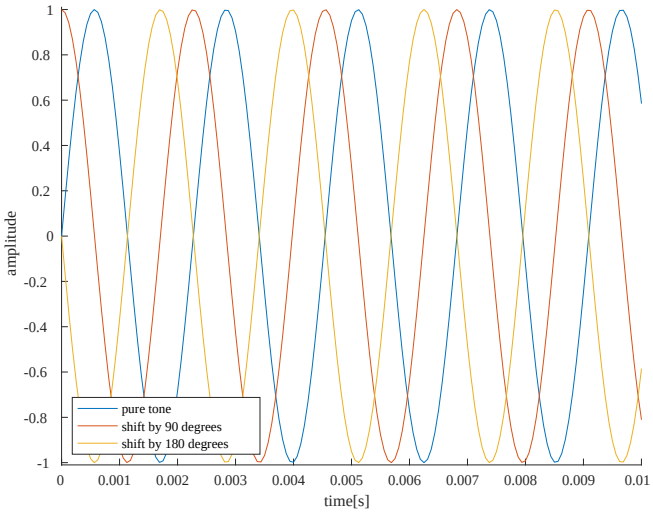


図 1-3: 振幅・位相の確認 実験結果（初期位相の変化）

1.2.4　考察

■振幅の変化について　振幅と，音量は比例することが確認できた．図 1-2 より，各正弦波において周期や初期位相が等しいので，縦軸が 0 になる時刻や最大値を迎える時刻は等しいく，振幅のみが異なる．振幅の変化だけでは，音色や音程は変わらないだろう．

■初期位相の変化について　初期位相の変化による音の変化は確認できなかった．図 1-3 より，振幅や周期は等しい．初期位相が異なるため最小値や最大値を迎える時刻が初期位相分異なるが，その変化を知覚できなかった．振幅や周波数の変化がないので，音量や音色の変化はない．

1.3　うなり

1.3.1　実験の目的

ティンパニやギターのチューニングを行うとき，音叉やチューナーから音を出して，異なる互いに周波数であれば気付きチューニングする．それぞれ異なる周波数の違いによるうなりの発生やその原因を数学的観点から考察する．

1.3.2　実験の方法と考え方

うなりとは，異なる周波数が干渉するとき起こる現象である．異なる周波数どうしの正弦波が干渉することで，強め合いの場所と弱め合いの場所が周期的に現れることで発生する．たとえば周波数 f_1 の正弦波に対して周波数 $f_2(\neq f_1)$ の正弦波を干渉させると，式 (1.5) より 1 秒間に N 回のうなりを聞くことができる．

$$N = \left|f_1 - f_2\right| \tag{1.5}$$

f_1 と f_2 の差が大きければ，当然うなり（N）の回数は多くなり，f_1 と f_2 の差が小さければうなりの回数は少なくなる．

本実験では，サンプリング周波数を 16kHz，提示時間 4 秒，周波数が $f_1 = 440$Hz の純音 y_1，$f_2 = 441$Hz の純音 y_2 を加算合成 $\left(y_1 + y_2\right)$ した波形を生成し再生する．1.3 章のソースコードは，src.A-3.
▶p.24

1.3.3　実験の結果

うなりは 4 回聞こえた．また，時間軸に対して加算合成したデータを描画すると図 1-4 となった．

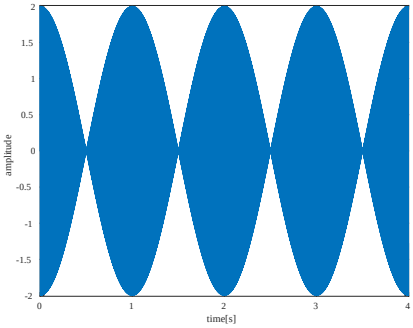


図 1-4: うなり 実験結果

1.3.4　考察

式 (1.5) に従って，$\left|f_1 - f_2\right| = \left|440 - 441\right| = 1$ より 1 秒あたり 1 回のうなりが聞こえるはずなので，この実験結果は論理的に正しい．加算合成波図 1-4 の振幅に着目しても，周波数 f_1，f_2 それぞれの正弦波の振幅が 1 であることを考えると，その加算合成波の振幅が 2 であることも論理的に正しい．さらに，この加算合成波は $t = 0$ のとき最大値を迎えており，これは初期位相が同一の正弦波の加算合成であることを表している．

■うなりの仕組み　異なる周波数 f_1，f_2 の正弦波を加算合成すると，式 (1.6) の和積の公式より，式 (1.7) となる．

$$\sin\theta_1 + \sin\theta_2 = 2\sin\left(\frac{\theta_1 + \theta_2}{2}\right)\cos\left(\frac{\theta_1 - \theta_2}{2}\right) \tag{1.6}$$

$$\sin(2\pi f_1 t) + \sin(2\pi f_2 t) = \underbrace{2\cos\left(2\pi\frac{f_1 - f_2}{2}t\right)}_{\to \mathtt{A}} \times \underbrace{\sin\left(2\pi\frac{f_1 + f_2}{2}t\right)}_{\to \mathtt{B}} \tag{1.7}$$

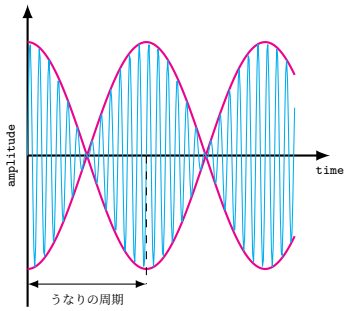


図 1-5: うなりの仕組み

$f_1 - f_2 \ll f_1 + f_2$ より，式 (1.7) の A を，B の振幅として考えると，図 1-5 のようになる．1 回のうなりは，A の半周期なので，$f_1 - f_2$ が，1 秒あたりのうなりの回数になる．これは f_1，f_2 を入れ替えても成り立つので，式 (1.5) が成り立つ．

1.4　フーリエ級数展開

1.4.1　実験の目的

フーリエ変換とは，自然界に現れるさまざまな曲線を三角関数に変換する方法である．フーリエ変換のうち基礎的なものはフーリエ級数展開と呼ばれ，周期的な三角関数の和で表す．フーリエ級数展開は周期関数のみに適用できるのに対して，

フーリエ変換は非周期関数にも適用できる [2, p.410].

矩形波は周期関数だが，正弦波ではない．これをプログラム上でどのように再現するのか．フーリエ級数展開を用いた矩形波の描画や，フーリエ級数展開をプログラム上で再現し，理想的な矩形波との比較する．

1.4.2　実験の方法と考え方

■フーリエ級数展開　周期 T の任意の周期信号 $f(t)$ に対して，それはより短い周期 $T/n(n = 1, 2, \dots)$ を持つ正弦波の重ね合わせで表すことができる．具体的には式 (1.8) の $N = \infty$ で表すことができ，これをフーリエ級数展開という [3, p.18 - p.19]．特に式 (1.8) の $N = 1$ の時を基本周波数と呼び，$k > 1$ のときの周波数を持つ正弦波を高調波と呼ぶ．

$$f(t) = \frac{1}{2}a_0 + \sum_{k=1}^{N}\left(a_k \cos(k\omega_0 t) + b_k \sin(k\omega_0 t)\right) \qquad \omega_0 = \frac{2\pi}{T} \tag{1.8}$$

■矩形波　矩形波は周期関数である．その周期を T とすると，関数 f は時刻 t に対して，フーリエ級数展開すると式 (1.9) になる．式 (1.9) の周期を $T = 1/2$ で定義し，$N = 50$ として出力した．

$$f(t) = \sum_{k=1}^{N} \frac{1}{2k-1} \sin\left(2\pi(2k-1)ft\right) \tag{1.9}$$

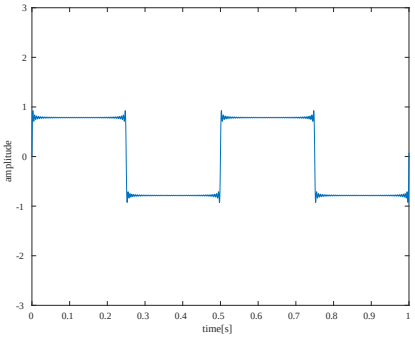


図 1-6: 矩形波 ($N = 50$)

■実験内容　周期 $T = 1/2$ として，式 (1.9) の N の値を $N = 1, 5, 25$ と増やしたときに次第に波形が矩形波に収束していくことを確認する．計算機で和 $\left(\sum\right)$ の演算方法を src.1-4 に示す．1.4 章のソースコードは，src.A-4. ▶p.25

src. 1-4: 和の演算

```
y = 0;
for k=1:50
    y = y + 1/(2*k-1) * sin(2*pi*(2*k-1)*f*t); % 矩形波の例
end
```

1.4.3　実験の結果

図 1-7 の通り，N を大きくすると，矩形波に収束することがわかった．また，基本周波数と矩形波の周期も目測では一致している．

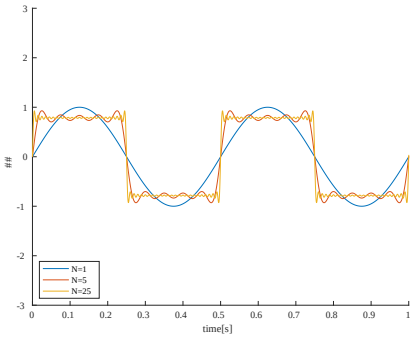


図 1-7: フーリエ級数展開 実験結果

1.4.4　考察

計算機を用いて N の値を無限大で出力は不可能である．図 1-6 のように $N = 50$ として出力すると，理想的な矩形波に近付くが完全ではない．

ここで N の値を大きくすると，$T/2, T$ あたりで尖ったもの（ひげ）が確認される．フーリエ級数展開では不連続で誤差が大きい．このことをギブス現象という [3, p.34].

1.5　白色ガウス雑音

1.5.1　実験の目的

ガウス雑音（*gaussian noise*）は以下のように定義されている．

> Gaussian noise represents statistical noise having the probability density function equal to that of the normal distribution. [4]

つまり，「正規分布の確率関数と等しい確率密度関数を持つ統計的ノイズ」を表す．一方，白色雑音（*white noise*）は，周波数成分を均等に含み，パワースペクトルが一定である不規則な波のことである [5]．白色性は，白色雑音が満たす性質である．

また，白色性を持つガウス雑音を白色ガウス雑音と呼ぶ．今回の実験では白色ガウス雑音を計算機上で構成し，その信号振幅のヒストグラムを作成する．そのヒストグラムがガウス分布（正規分布）の概形に従っているかを確認する．

1.5.2　実験の方法と考え方

■MATLAB®関数の説明　標準正規分布から乱数の行列を取り出すには，`randn` 関数を用いる．src.1-5［1-2 行目］で，乱数生成機の初期化を行い n 行 m 列の乱数を取り出す．また，src.1-5［3-5 行目］で，データ列 `X` に対して，ヒストグラムを `num` 段階で分割し表示する．

src. 1-5: 乱数生成・ヒストグラム

```
1 rng(0, 'twister');
2 X = randn(n,m);
3 num = 100;
4 [h, c] = hist(X, num);
5 plot(c,h);
```

■ガウス分布と標準正規分布　ガウス分布とは，平均値・中央値・最頻値が一致するという特徴を持つ確率分布．その確率変数を x としたときの確率密度関数 f は式 (1.10) のように与えらえる．標準正規分布は，平均 $\mu = 0$，標準偏差 $\sigma^2 = 1$ のガウス分布である．その標準正規分布の確率密度関数 f_N は式 (1.11) となる．

src. 1-6: 平均と標準偏差の変更

```
X = randn(n,m); % 乱数生成
X = a.*X + b; % 演算
```

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu^2)}{2\sigma^2}\right) \qquad \left(\mu：平均 \quad \sigma^2：分散 \quad -\infty < x < \infty\right) \tag{1.10}$$

$$f_N(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \qquad \left(-\infty < x < \infty\right) \tag{1.11}$$

> 確率変数 x に対して，その平均が ${\mu_x}$ で，分散が ${\sigma_x}^2$ のとき，$y = ax + b$　(aとbは定数) で定義される確率変数 y の平均は ${\mu_y} = a{\mu_x} + b$ で，分散は ${\sigma_y}^2 = a^2{\sigma_x}^2$. [6]

したがって，標準正規分布に従って出力されたデータ列の平均を `a`，標準偏差を `b` に変更したい場合には，1 行 m 列データ列 `X` に対して `X` の各要素と `a` の積をとり，データ列 `X` と `b` の和をとる（src.1-6）．今回は，サンプリング周波数を 16kHz，平均 $\mu = 0$，標準偏差 $\sigma^2 = 0.2$，提示時間 1 秒のガウス雑音を生成する．つまり，16k 個の乱数を作成し，それを振幅として音波を形成する．白色性は，`randn` 関数で保証されているものとする．1.5 章のソースコードは，src.A-5. ▶p.26

1.5.3　実験の結果

ガウス雑音の波形を図 1-8，ヒストグラムを図 1-9 に示す．聴音した結果，「ザー」といういわゆる雑音が聞こえた．

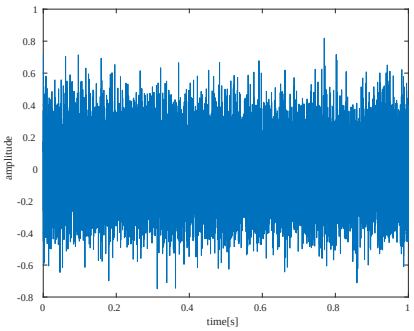


図 1-8: ガウス雑音の波形

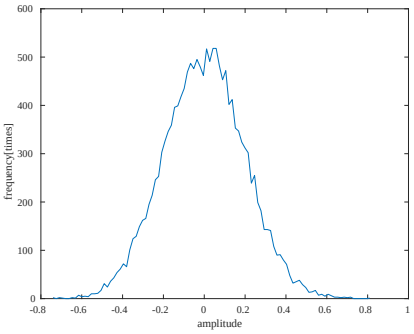


図 1-9: ヒストグラム

1.5.4　考察

聴音確認による，ガウス雑音の特徴はとらえることができなかった．ヒストグラム図 1-9 を確認すると，平均 $\mu = 0$ のガウス分布の概形に従っているだろう．今回はサンプリング周波数を 16kHz で音波を形成したが，理論上 `Fs` $= \infty$ で，ガウス分布の概形を得るであろう．

第 2 章

周波数解析

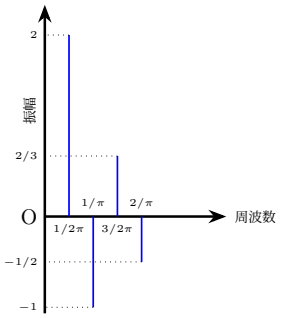
2.1　フーリエ変換と周波数解析

2.1.1　実験の目的

任意の周期関数は式 (1.8) の $N = \infty$ で表せる．また，任意の関数に対して離散フーリエ変換*すると「周波数に対する振幅」を得ることができる．波形の振幅と周波数をグラフに描画したものを振幅スペクトル†と呼ぶ（図 2-1）．さらに，周波数に対してフーリエ変換した絶対値の 2 乗をとったグラフをパワースペクトルと呼ぶ．

今回の実験では，純音と矩形波に対して周波数解析し，純音では解析結果より純音の周波数をより多く含んでいるか確認する．また矩形波では高調波の存在を確認し，振幅が式 (1.9) 通りになっているか確かめる．

正弦波	振幅	周波数
$2\sin(t)$	2	$1/2\pi$
$-\sin(2t)$	-1	$1/\pi$
$\frac{2}{3}\sin(3t)$	$2/3$	$3/2\pi$
$-\frac{1}{2}\sin(4t)$	$-1/2$	$2/\pi$
$\vdots$		



$$f(t) = \sum_{k=1}^{N} \frac{1}{2k-1} \sin\left(2\pi(2k-1)ft\right) \quad (1.9)$$

図 2-1: 周波数解析

2.1.2　実験の方法と考え方

■高速フーリエ変換の実装　MATLAB®では高速フーリエ変換‡をする関数（`fft`）がある．データ列 `y` に対して振幅スペクトルを取得する手順は以下の通りである（src.2-1）．

1. 高速フーリエ変換する．
   `fft` 関数は出力として，サンプリング周波数 `Fs` に対して，`[-Fs/2, Fs/2]` 範囲の周波数に対する振幅のデータを得る．
2. 出力データ列 `fft(y)` のデータを整列させる．
   `fft` 関数の出力は，正のデータ・負のデータ（左右）が入れ替わった状態で出力される（図 2-2，図 2-3）．
3. これまでの過程で出力されるデータは複素数データである．絶対値を取るために `abs` 関数を用いる（図 2-4）．
4. グラフとして描画するために，周波数軸を作成する§（src.2-1）．

*デジタル値（離散値）に対してフーリエ変換するときは，離散フーリエ変換と呼ぶ．このレポート常の全実験では，全てのフーリエ変換を離散値に対して行うので，離散フーリエ変換をフーリエ変換と記す．

†振幅スペクトルの縦軸は，便宜上 `amplitude`（振幅）とする．

‡高速フーリエ変換は，離散フーリエ変換を高速に計算する手法

§今回は各要素を 2 乗しないので，縦軸は `power` でなく `|amplitude|` とする．

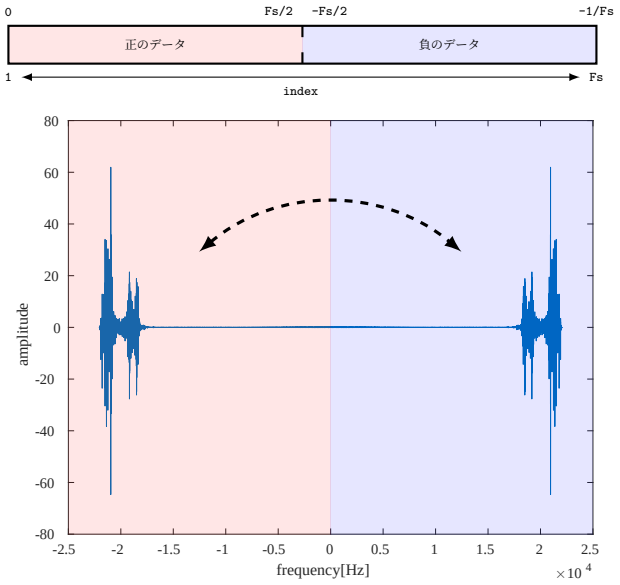


図 2-2: `fft` 直後の出力データ

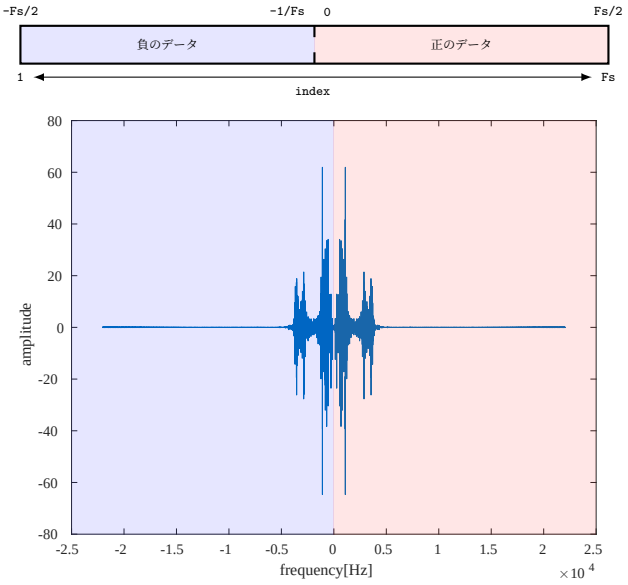


図 2-3: `fftshift` 後の出力データ

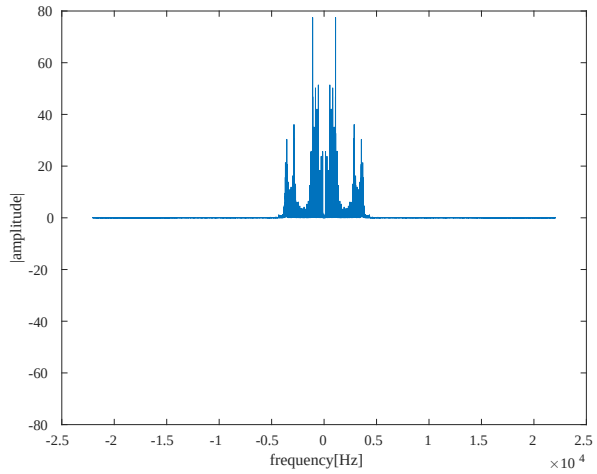


図 2-4: `abs` を適用した後のデータ（グラフ）

src. 2-1: 振幅スペクトルの取得

```
y; % データ列
y_fft = fft(y); % 高速フーリエ変換
y_fftshift = fftshift(y_fft); % 左右交換
y_abs = fft_y; % 絶対値
ln = length(y_abs);
% 周波数テーブルの作成
freq = [-Fs/2 : Fs/ln : Fs/2 - Fs/ln];
plot(freq, y_abs); % グラフ描画
```

周波数テーブルの作成について，得られたデータ数 `ln` に対して，周波数テーブルの長さ `Fs` に収める必要があるので，ステップ幅 `Fs/ln` となる`-Fs/2` から `Fs/2` の配列を作成する．

■実験の内容　今回の実験では，純音と矩形波の周波数解析する．サンプリング周波数を 8192Hz と設定する．

純音　440Hz の純音で実験する．

- 純音から 1024 点取り出す．
  データ列の先頭から 1024 個のデータを選択する方法で行う．
- 振幅スペクトルを 0Hz から 1kHz まで描画する．
- 440Hz にピークが来ているか確認する．

矩形波　128 点値 0 が続き，128 点値 1 が続く波形を 4 回繰り返す矩形波（図 2-5）．

- 振幅スペクトルを 0Hz から 200Hz まで描画する．
- 高調波が発生しているか，その振幅が式 (1.9) 通りになっているか確認する．

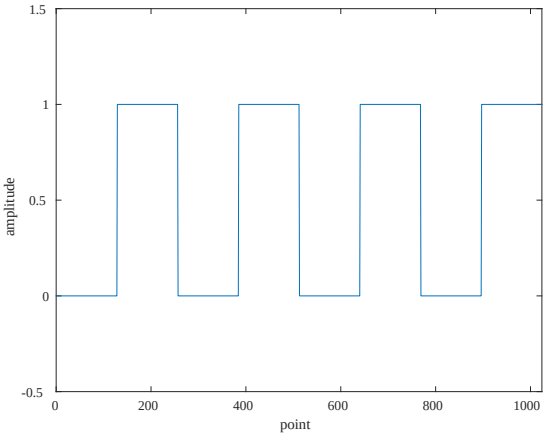


図 2-5: 作成した矩形波

2.1 章のソースコードは，src.B-1，src.B-2.
▶p.26　▶p.27

2.1.3　実験の結果

出力された結果に対して，特出する点の座標を表示させる．図 2-6 について，周波数 440Hz の振幅は最大であることが分かった．また，図 2-7 より図中にある点の高調波が確認された．

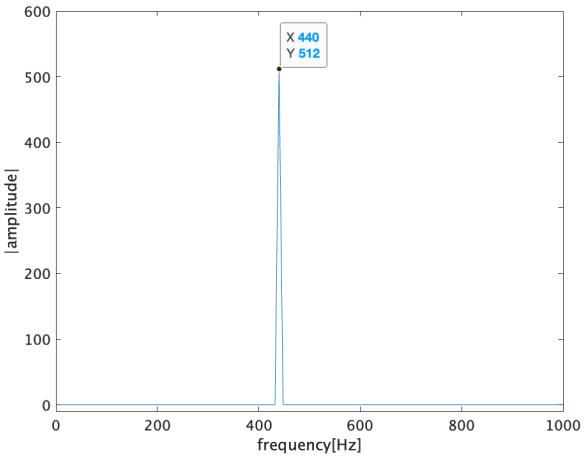


図 2-6: 純音の振幅スペクトル

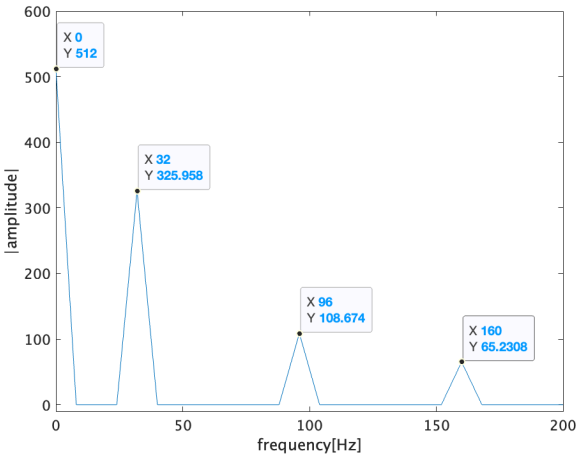


図 2-7: 矩形波の振幅スペクトル

2.1.4　考察

■純音　実験結果より，純音の周波数 440Hz が一番多く，ほかの高調波は確認できなかった．

■矩形波　図 2-7 より高調波の存在は確認できる．矩形波のフーリエ級数展開は式 (1.9) である．$k = 1, 2, 3$ のときの正弦波と，周波数 X，振幅 Y を式 (2.1) に示す．

$$\begin{aligned} \sin(2\pi(2-1)ft) &= \sin(2\pi ft) && (k=1) \quad (X,Y) = (32, 325.958) \\ \frac{1}{3}\sin(2\pi(4-1)ft) &= \frac{1}{3}\sin(6\pi ft) && (k=2) \quad (X,Y) = (96, 108.6474) \\ \frac{1}{5}\sin(2\pi(6-1)ft) &= \frac{1}{5}\sin(10\pi ft) && (k=3) \quad (X,Y) = (160, 65.2308) \end{aligned} \tag{2.1}$$

$(X,Y) = (0, 512)$ の振幅スペクトルについて，周波数が 0 である正弦波は考えることができないので，$k = 1$ を $(X,Y) = (32, 325.958)$ とすると，$k = 1$ の正弦波に対して，振幅が $108.674/325.958 = 1/3$ 倍，$65.2308/325.958 = 1/5$ 倍になっている．また，周波数についても，$k = 1$ の正弦波に対して，$96/32 = 3$ 倍，$160/32 = 5/3$ 倍になっている．これは式 (2.1) の正弦波と一致する．なぜ周波数軸の 0 対する振幅の存在については分からなかった．

2.2　LPF（ローパスフィルタ）

2.2.1　実験の目的

ローパスフィルタ（以下 LPF）は以下のように説明されている．

> ある周波数（カットオフ周波数）より高い周波数を通さないフィルタは LPF（ローパスフィルタ）と呼ばれる．
>
> [7, p.65]

今回の実験では LPF を作成して，別途作成した音に対してフィルタを適用する．フィルタを適用する前後での振幅スペクトルを比較し，フィルタが機能しているかを確認するとともに，フィルタが適用されているか聴音確認する．

2.2.2　実験の方法と考え方

今回フィルタを適用させる音源は，ドイツ語音階で「C4」から「C5」まで 0.25 秒ずつ音を提示する．提示する音の順序を式 (2.2) に，各音階の周波数を表 2-1 に示す．

$$\mathrm{C4} \rightarrow \mathrm{D4} \rightarrow \mathrm{E4} \rightarrow \mathrm{F4} \rightarrow \mathrm{G4} \rightarrow \mathrm{A4} \rightarrow \mathrm{H4} \rightarrow \mathrm{C5} \tag{2.2}$$

音源に対して，G4 より低い音のみを通過させる LPF を作成しフィルタを適用する．カットオフ周波数を G4 の周波数に設定すると，G4 の周波数より小さいわずかな成分が除去できないため，F4 と G4 の中間に位置する周波数（375Hz）へ設定する．

表 2-1: 音階と周波数

音階	C4	D4	E4	F4	G4	A4	H4	C5
周波数 Hz	261.38	293.67	329.63	349.23	392.00	440.00	493.88	523.23

■LPF の作成と適用　今回はフィルタの行列を作る方法ではなく，周波数テーブルの閉区間 $\left[-375\mathrm{Hz}, 375\mathrm{Hz}\right]$ 以外の振幅を 0 にする方法でフィルタを適用する．（src.2-2）2.2 章のソースコードは，src.B-3. ▶p.27

src. 2-2: フィルターを適用する

```
% データ列yをフーリエ変換後，ffthiftしてabsをとったものを "fft_y" に格納している．
% 周波数テーブルの変数名は "freq".
for k=1:length(fft_y)
    if freq(k) >= 375 || freq(k) <= -375
        fft_y(k) = 0;
    end
end
```

2.2.3　実験の結果

LPF 適用前の音声波形と，LPF 適用後の音声波形をに示す．また，図 2-10 には上図にフィルタ適用前の振幅スペクトル，下図にフィルタ適用後の振幅スペクトルを示している．聴音確認の結果 G4 以降の音は聞こえなかった．

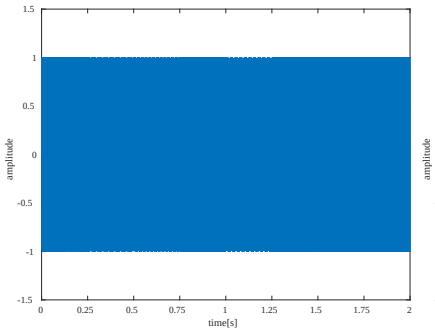


図 2-8: LPF 適用前の波形

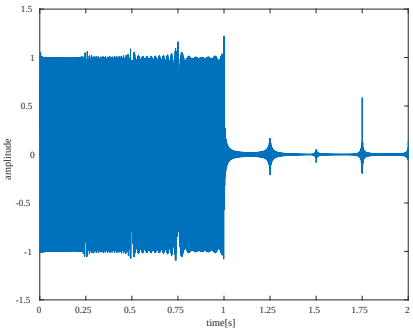


図 2-9: LPF 適用後の波形

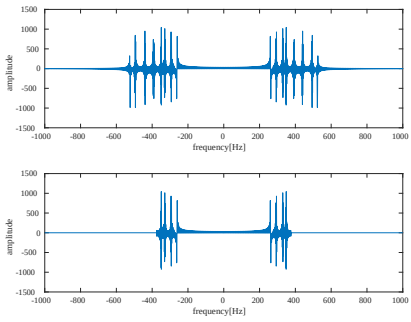


図 2-10: 振幅スペクトルの確認

2.2.4　考察

結果より 1 秒後の音以降が聞こえない．1 音階 0.25 秒あるので，4 音階分（C4，D4，E4，F4）までが聞こえることを考えると，実験結果は理論的に正しい．図 2-10 より，特定周波数以降が切り取られていることも確認できる．ただし，1 秒後以降の波形も完全に 0 ではない理由は分からなかった．

第 3 章

音声合成

3.1　合成波における位相の操作

3.1.1　実験の目的

ノコギリ波は，矩形波と同様周期関数である．ゆえにノコギリ波は式 (1.8) で表すことができる．矩形波とノコギリ波は合成波である．

この実験ではノコギリ波と矩形波の，初期位相の変化に対して波形の変化と音の変化を確かめる．初期位相の変化は，固定値と高調波成分ごとにランダムな値で確かめる．

3.1.2　実験の方法と考え方

■ノコギリ波　ノコギリ波は式 (3.1) を周期 2π の関数として周期的に式 (3.2) へ拡張したものであり，ノコギリ波をフーリエ級数展開すると，式 (3.3) の $N = \infty$ で表せる．

$$f(t) = t \tag{3.1}$$

$$f(t) = f(t + 2k\pi) \qquad (k \in \mathbb{Z}) \tag{3.2}$$

$$f(t) = \sum_{k=1}^{N} (-1)^{k-1} \frac{2}{k} \sin(kt) \tag{3.3}$$

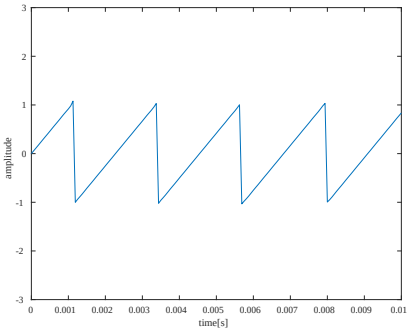


図 3-1: ノコギリ波 ($N = 50$)

■乱数の生成　乱数の生成は MATLAB®の `rand` 関数を用いる．`rand` 関数は引数や演算を与えない状態（`r = rand`）で用いると，区間 $(0, 1)$ の一様分布から取り出された乱数スカラを返す [8]．今回は，引数や演算を与えずに `rand` 関数を用いる．

波形に高調波成分ごとのランダムな初期位相を与えるためには，繰り返し処理の都度 `rand` 関数を呼び出すと良い（src.3-1）．

src. 3-1: ランダム初期位相

```
for k=1:50
 y =y+sin(2*pi*f*t+rand);
end
```

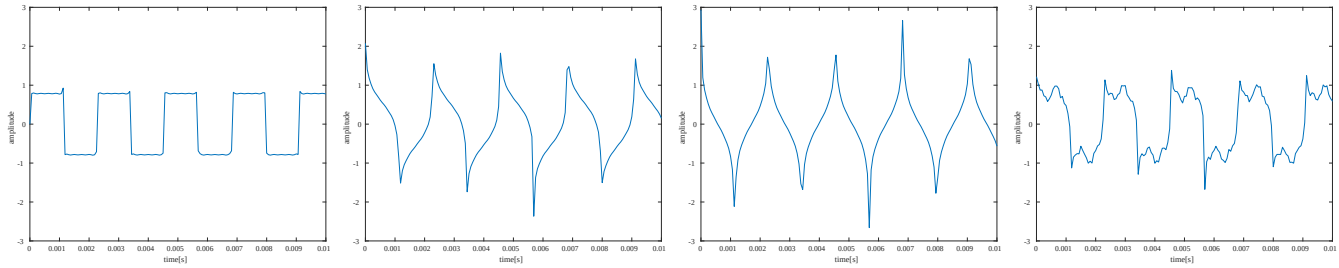
■実験の内容　今回の実験では，ノコギリや初期位相に対して，固定値の初期位相 $\pi/4$，$\pi/2$ を与え，高調波ごとにランダムな初期位相を与える．また，式 (1.9)，式 (3.3) ともに $N = 50$ として実装する．3.1 章のソースコードは，src.C-1，src.C-2.
▶p.28　▶p.29

3.1.3　実験の結果

■矩形波　音の波形を図 3-2 に示す．聴音確認の結果，図 3-2(a) に比べて，図 3-2(c)，図 3-2(b) の方が滑らかな音に聞こえた．図 3-2(a) と図 3-2(d) の音は似ていた．矩形波は純音に比べて，掠れた音が聞こえた．音量について，図 3-2(c)，図 3-2(b) は図 3-2(a) の 3/4 程度の音量に聞こえた．図 3-2(a) と図 3-2(d) の音量変化は確認できなかった．

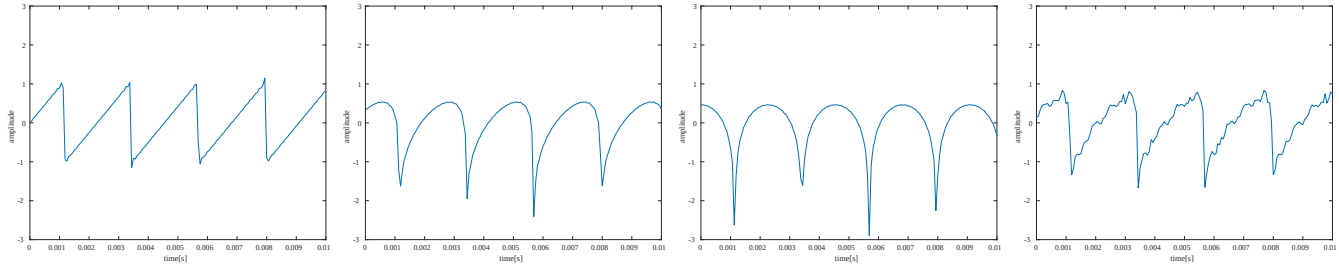
■ノコギリ波　音の波形を図 3-3 に示す．聴音確認の結果，図 3-3(a) に比べて，矩形波と同様に図 3-3(c)，図 3-3(b) の方が滑らかな音に聞こえた．図 3-3(a) と図 3-3(d) の音は似ていた．ノコギリ波は純音に比べて，尖った音が聞こえた．音量について，図 3-2(c)，図 3-3(b) は図 3-3(a) の 3/4 程度の音量に聞こえた．図 3-3(a) と図 3-3(d) の音量変化は確認できなかった．

■波形　初期位相を固定値で与えると，初期位相が 0 の波形とは異なる波形を示す．



(a) 矩形波　(b) 初期位相 $+\pi/4$　(c) 初期位相 $+\pi/2$　(d) 初期位相 `rand`

図 3-2: 矩形波の初期位相



(a) ノコギリ波　(b) 初期位相 $+\pi/4$　(c) 初期位相 $+\pi/2$　(d) 初期位相 `rand`

図 3-3: ノコギリ波の初期位相

3.1.4　考察

純音（正弦波）では初期位相を与えると，その波形は横軸に並行移動する．しかし，矩形波やノコギリ波は並行移動せずに波形が変わる．これは，矩形波やノコギリ波がさまざまな周波数の高調波から成り立っているからであろう．また，初期位相を与えた波の振幅が，源波形振幅の 2 から 3 倍である理由として，源波形の音波で弱めあっていた点の高調波が並行移動することによって，強めあいに転じたことが考えられる．

音量について，振幅と音量は比例する．初期位相を与えた場合，波形の振幅が大きくなるにもかかわらず，音量は小さくなることが分かった．この理由は今後の課題として調査したい．

また，実験結果からわかるように，矩形波やノコギリ波に初期位相を与えると，波形が滑らかになるとともに音も滑らかに知覚された．

3.2　録音した音声の操作

3.2.1　実験の目的

最近のイヤホンやヘッドホンには「ノイズキャンセリング」という機能が搭載されている．これは周りの雑音を音声加工によって軽減する技術である．この技術は雑音と，それをもとに生成した音の加算合成波を 0 にすることで再現している．今回の実験では MATLAB®を用いて，ある音波に対して合成波が 0 になるような音波を生成する．生成した波と，元となった波の音と波形の違いを明確にする．

3.2.2　実験の方法と考え方

正弦波 $y_1 = \sin(x)$ に対して，合成波が 0 になるような正弦波は，$y_2 = -\sin(x)$ であり，y_1 と -1 の積である．無論，$y_1 + y_2 = 0$ となる．

これは正弦波に限った話ではない．任意の実数値関数 $f : x \longrightarrow f(x) \quad (x \in \mathbb{R})$ に対して，横軸 x，縦軸 $f(x)$ のグラフを

src. 3-2: モノラルへの変換

```
% y : N行2列
% ステレオオーディオデータ
mono = y(:,1); % 2列目を削除
```

描画する．横軸を対称の軸として，値 $f(x)$ を線対称移動すると $-f(x)$ になる．つまり，ある音波に対して，その波の値と -1 の積を取った値を加算合成すれば，合成波は 0 になる．

■実装　音波 A の各要素と -1 の積を演算するには，`A.*(-1)` と入力する．`.*` は各要素との積をとる演算子である．また，録音した音声がステレオオーディオの場合，モノラルオーディオに変換する（src.3-2）．

■実験の内容　今回は母音「a」,「i」の 2 つを処理する．それぞれ振幅データと -1 の積をとる．処理後の波形，元の波形，合成波の，それぞれ一部描画する．「a」の音声データを `sound1.wav`,「i」の音声データを `sound2.wav` に格納している．3.2 章のソースコードは，src.C-3, ▶p.30　src.C-4. ▶p.31

3.2.3　実験の結果

波形を図 3-4，図 3-5 に示す．聴音確認の結果，処理の前後で音の変化は確認できなかった．さらに，合成波（`synthetic wave`）は常に 0 であった．

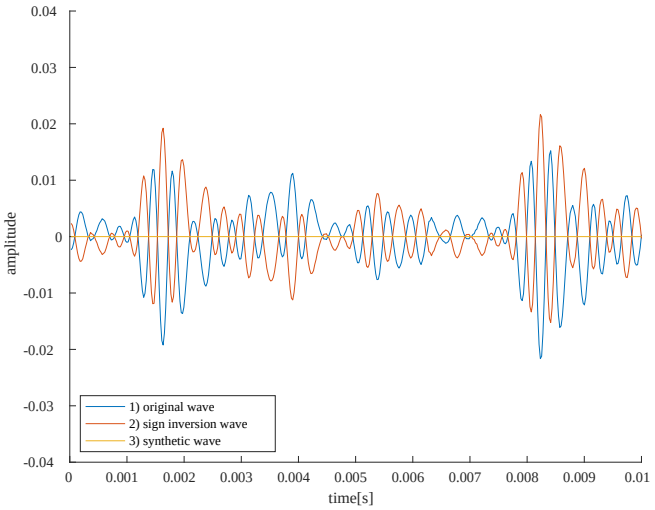


図 3-4: 処理前後の波形：「a」

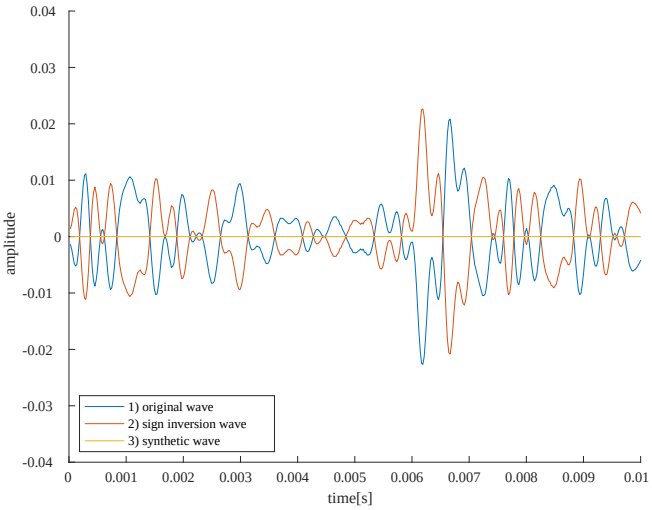


図 3-5: 処理前後の波形「i」

3.2.4　考察

結果より，合成波が 0 であることを確認した．上下（正負）を逆にして生成した音波と元となった音波を合成すると無音になる．波の正負が反転しても，聞こえ方は変わらないと知覚する理由として，ヒトの聴覚は振動を「音圧」としてとらえていることが考えられる．

3.3　音声合成

3.3.1　実験の目的

1.4 章より，任意の波はフーリエ変換し，周波数解析できる．今回の実験では，音声を周波数解析し，振幅スペクトル上でピークのある周波数および振幅データを抽出した後，それらを元に音波を生成する．生成した音波と元となった音波の違いを聴き比べる．

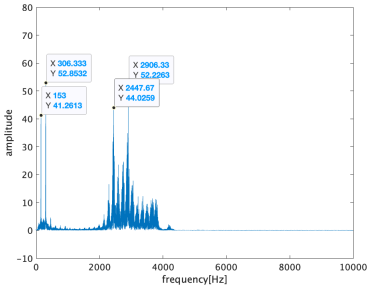


図 3-6: 縦横軸の値を抽出する

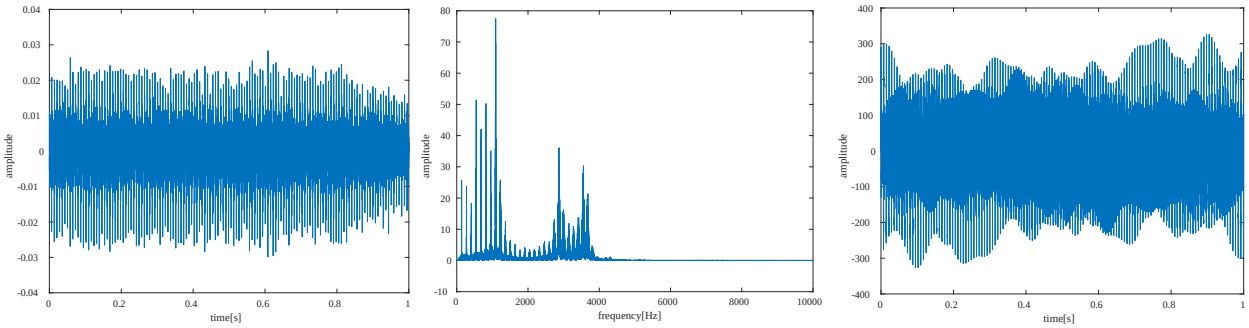
3.3.2　実験の方法と考え方

3.2 章で用いた `sound1.wav`, `sound2.wav` に対して，周波数解析する．MATLAB®の `plot` 関数で出力したグラフは，ピークの点をクリックするとその点の値を取り出すことができる（図 3-6）．今回は，0Hz から 10000Hz までを振幅スペクト

ルとして書き出す．取り出した点を行列に保存し，その振幅と周波数から純音を生成する．このとき，時刻 t に対して，周波数 f と振幅 A に対して，$f(t) = A\sin(2\pi ft)$ と表せること，すべての波は正弦波の和で表せることをもとに，生成した純音を合成し，音波を作成する．3.3 章のソースコードは，src.C-5.
▶p.31

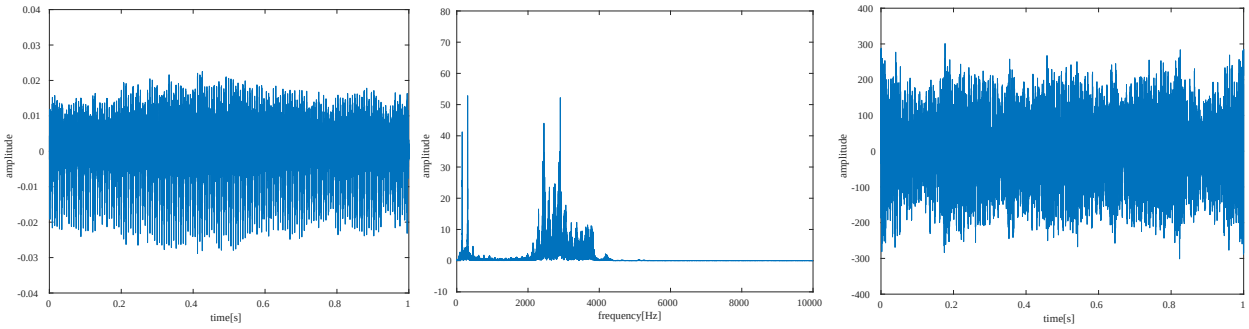
3.3.3　実験の結果

聴音確認の結果，合成後の音は，「a」，「i」と聞こえたが，原音波に比べて「ガサガサ」とした音であった．



(a) 源音波（一部）　(b) 振幅スペクトルの絶対値　(c) 合成音波（一部）

図 3-7: 母音「a」



(a) 源音波（一部）　(b) 振幅スペクトルの絶対値　(c) 合成音波（一部）

図 3-8: 母音「i」

3.3.4　考察

源音波と合成音波を比べてみると，縦軸が異なる．これは，特定の周波数を取り出すことによって，弱めあう正弦波を合成しなかったためであろう．また，「ガサガサ」と音が聞こえるのは，抽出していない大部分の高調波成分が欠け，「本来必要な周波数の正弦波がないこと」が原因であろう．

第 4 章

聴覚と音声信号

4.1 音脈分凝を生じる刺激の作成

4.1.1 実験の目的

■音脈分凝とは 例として，音列 C4 G4 C4 G4 C4 G4 ... をあげる．この音列を長く聴いた場合，テンポが遅いと音列は「旋律」として聞こえるが，テンポを早くするについれて C4 C4 C4 ... G4 G4 G4 ... と，音列が分離して聞こえる．この現象を音脈分凝と呼ぶ．また，テンポを一定にした場合，2 つの音程差（C4 と G4 は完全 5 度）が大きいほど，音脈分凝が生じやすい [9, p.182].

■実験 今回の実験では，テンポを一定にした場合の，周波数による音脈分凝の生じ方について調査する．音脈分凝が生じる波形を作成し，音脈分凝が生じる周波数組み合わせと，音脈分凝が生じにくい組み合わせを実験により発見する．

4.1.2 実験の方法と考え方

サンプリング周波数を 16kHz，1 音を 0.2 秒に設定し，音 A と音 B を連結させた音 C を 10 回繰り返す．正弦波の行列を連結するために，`repmat` 関数を用いる．実験する周波数の組み合わせを表 4-1 に示す．周波数 f_A と周波数 f_B の差を $f_D = \left|f_A - f_B\right|$ と定義する．4.1 章のソースコードは，src.D-1. ►p.33

4.1.3 実験の結果

聴音確認による結果を表 4-1 に示す．

表 4-1: 音脈分凝 実験結果

	周波数 f_A	周波数 f_B	周波数差 f_D	実験結果
実験 1	1000	990	10	音脈分凝は生じず，滑らかな音に聞こえた．
実験 2	1000	800	200	音脈分凝は生じず，別音の組み合わせによる旋律に聞こえた．
実験 3	1000	200	800	音脈分凝が生じ，1000Hz，200Hz の 2 音が分離して聞こえた．

4.1.4 考察

結果より，周波数組み合わせの差 f_D が大きいほど音脈分凝が生じやすいこと，$f_D = 10$ 程度であれば音の切れ目すら聞こえないことが分かった．今回の実験では，3 つの組み合わせのみ実験したため，音脈分凝が生じる具体的な周波数差 f_D は分からなかった．テンポによる音脈分凝の生じやすさについても，今後の課題として調査したい．

4.2　連続聴効果を生じる音刺激の作成

4.2.1　実験の目的

■連続聴効果とは　ある一連の音を一部，短時間だけ削除し，雑音に置き換える．ヒトは，「元の音声の中に雑音が混入された」と知覚し，部分的な削除には気付かない．この現象を聴覚的補完，または連続聴効果と呼ぶ [9, p.182 - p.183].

■実験　連続聴効果が生じやすい刺激と，連続聴効果が生じにくい刺激を作成する．雑音や純音の周波数と，連続聴効果の関係について明らかにする．

4.2.2　実験の方法と考え方

■雑音の作成　雑音を，白色雑音（ノイズ A）と白色雑音からフィルタ処理により，周波数の 1020Hz から 1620Hz を除去した雑音（ノイズ B）を作成する．白色雑音は，乱数を `rand` 関数を用いて作成する．`rand` 関数は，0 から 1 までの実数を戻り値としているため，`rand` 関数で −1 から 1 までの実数を戻り値としたいときは，各要素から −0.5 し，全データを 2 倍する．

周波数の 1020Hz から 1620Hz を除去するとき，`fft` の戻り値を考慮してフィルタを作成する（src.4-1）．図 4-1 と図 4-2 を比較すると，ノイズ B に周波数フィルタの適用が確認される．フィルタを通した振幅スペクトルに対して，逆フーリエ変換することで音波を形成する．

■連続聴効果が生じる音刺激の作成　純音の周波数は，純音 A= 400Hz，純音 B= 1300Hz で作成する．純音 B はノイズ B に存在しない周波数を設定した．純音 1 秒，雑音 0.1 秒を 4 回繰り返したものを実験する音刺激とする．実験する周波数の組み合わせを表 4-2 に示す．4.2 章のソースコードは，src.D-2.
▶p.34

src. 4-1: 白色雑音の作成

```
white_noise = 2 * (rand(1,Fs) - 0.5); % 白色雑音の作成
B_fft = fftshift(fft(白色雑音)); % フーリエ変換
noise_filter = ones(1,Fs);
filter_rangeS = 1020; filter_rangeG = 1620; % カット終始 周波数
noise_filter((Fs/2)-filter_rangeG : (Fs/2)-filter_rangeS) = 0; % フィルタの作成
noise_filter((Fs/2)+filter_rangeS : (Fs/2)+filter_rangeG) = 0; % フィルタの作成
B_fft = noise_filter.* A_fft; % 各要素とフィルタの積を取る
noise_B = real(ifft(ifftshift(B_fft))); % 虚部を含むため，real関数を用いる
```

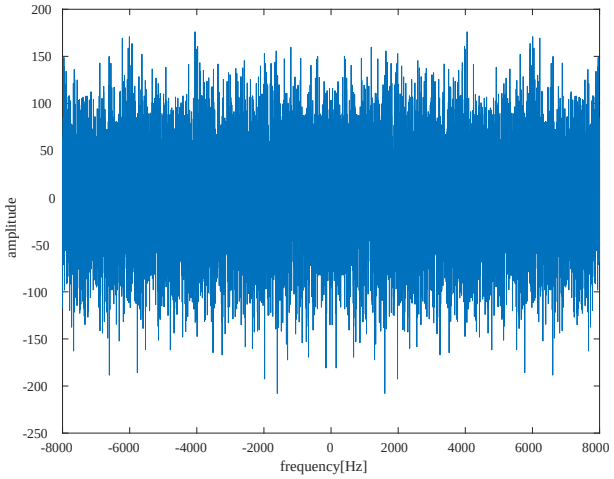


図 4-1: ノイズ A の `fft` 後

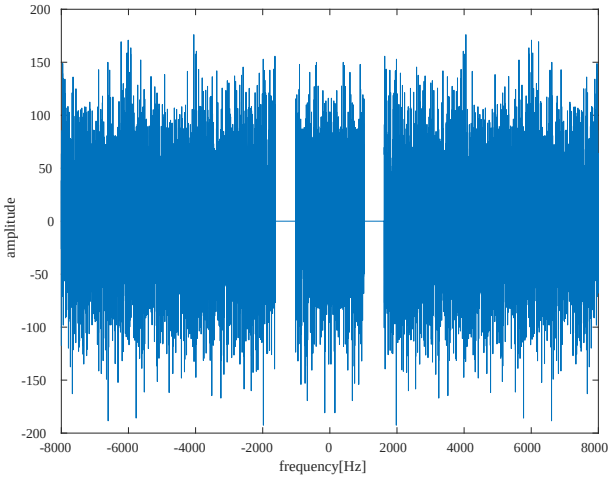


図 4-2: ノイズ B の `fft` 後

4.2.3 実験の結果

実験結果を表 4-2 に示す.

表 4-2: 連続聴効果の実験組み合わせと結果

	純音	挿入するノイズ	実験結果
実験 1	純音 A	ノイズ A	雑音部で小さい純音が聞き取れた.
実験 2	純音 A	ノイズ B	雑音部で小さい純音が聞き取れた.
実験 3	純音 B	ノイズ A	雑音部で小さい純音が聞き取れた.
実験 4	純音 B	ノイズ B	雑音部で純音は聞き取れず，純音が非連続に聞こえた.

4.2.4 考察

実験結果より，純音の周波数がノイズの周波数に含まれていないとき，連続聴効果は生じなかった．挿入されるノイズに純音の周波数が含まれているとき，雑音の周波数をもとに，脳内で純音を補完していると考えられる．ゆえに，純音の周波数をノイズから削除すると，連続聴効果が生じにくいと考えられる.

今回は純音に対する連続聴効果を実験したが，馴染み深い言語（日本語のフレーズ）に対して雑音を挿入すると，純音に比べて連続聴効果を知覚できるのではないだろうか．日ごろから日本語を聞きなれている我々なら，より強く補完が働くだろう.

4.3 ミッシングファンダメンタル波刺激の作成

4.3.1 実験の目的

■ミッシングファンダメンタルとは 人間の聴覚系では,「基本周波数の推定」を行っている．基本周波数成分が物理的に存在しなくても，倍音成分から基本周波数を推定できる．たとえば，300Hz，400Hz，500Hz が同時に存在すれば，基本周波数 100Hz を知覚できる [10]．ミッシングファンダメンタルとは，残っている周波数成分から推定された，基本周波数成分のことである.

■実験 今回の実験では，矩形波，ノコギリ波の各波形を加工してミッシングファンダメンタル波を作成する．作成したミッシングファンダメンタル波と原波形を比較して，波形や音の違いについて考察する.

4.3.2 実験の方法と考え方

矩形波を例に考える．1.4 章より，矩形波は式 (1.9) のように三角関数の合成で表すことができる．基本周波数は $k = 1$ で表され，基本周波数の三角関数 f_1 は，式 (4.1) である.

$$f_1(t) = \sin(2\pi f t) \tag{4.1}$$

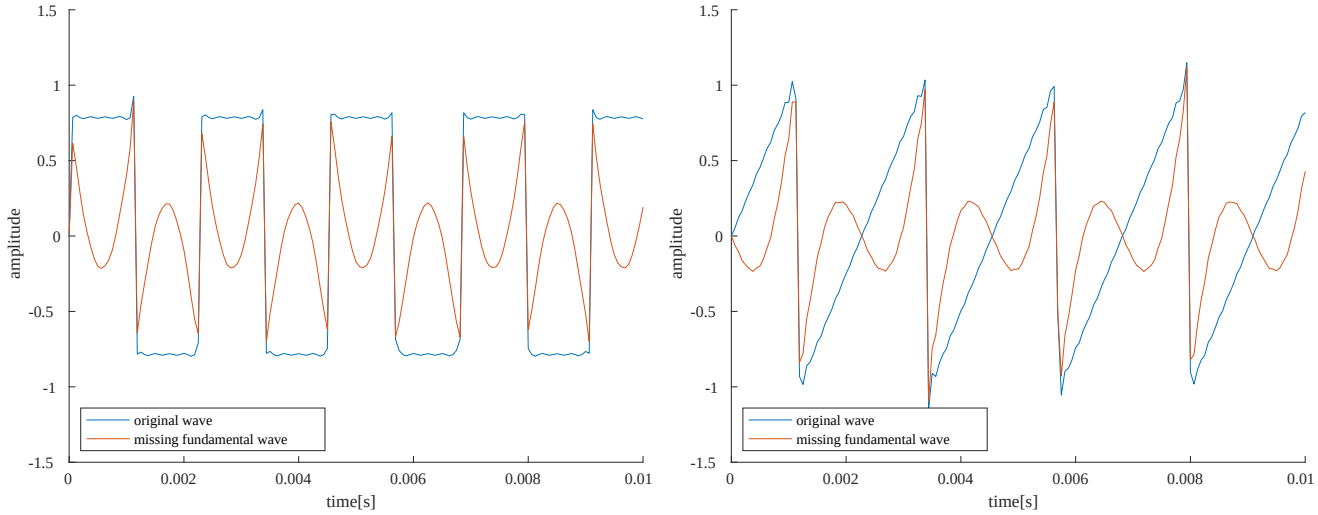
今回は，原波形とミッシングファンダメンタル波の両方を作成する必要があるので，作成した矩形波から，基本周波数の値 $f_1(t)$ を引いてミッシングファンダメンタル波を生成する．ノコギリ波の場合も同様である.

今回は，基本周波数を 440Hz として実験する．つまり，基本周波数 440Hz を除いた音波から基本周波数が聞こえると，その音波はミッシングファンダメンタル波であることを確認できる．4.3 章のソースコードは，src.D-3.
►p.35

4.3.3 実験の結果

■聴音確認 聴音確認の結果，周波数を取り除いた音波でも基本周波数 440Hz を聞き取ることができた.

■波形　出力された波形を図 4-3 に示す．グラフの概形は異なるが，矩形波，ノコギリ波の両方とも周期関数となっており，傾きが最大の時刻が原波形と一致している．



(a) 矩形波　　(b) ノコギリ波

図 4-3: 原波形とミッシングファンダメンタル波の比較

4.3.4　考察

ミッシングファンダメンタルを聞き取ることができる理由として考えられるのが，原波形とミッシングファンダメンタル波の周期が一致している点である．図 4-3(a) より，原波形のひげの部分とミッシングファンダメンタル波の最大値が目視では一致している．また，図 4-3(b) の最大値と最小値の時刻も目視では一致している．これらから基本周波数が知覚できるのではないだろうか．

第 5 章

結論

今回の実験を通して，正弦波の操作と実際の音波を比較した．振幅の変化は音量の変化として聞こえ，初期位相の変化による音の変化は知覚できなかった．さらに，フーリエ級数展開を用いた周期関数や，統計的な雑音も音波や波形で表現し，考察した．

波形の周期に対して振幅をとる周波数解析を用いて，特定周波数鑿を通過させるフィルタの作成，適用や，音声を合成した．音声合成は，特定周波数のみを抽出して再合成した．原音とは程遠いまでも，母音の発音は聞き取れた．また，正弦波では知覚できなかったが，矩形波やノコギリ波など高調波を多く含む波形では，初期位相の変化による音の変化を知覚できた．

最後に，人間の聴覚特性を再現するための音波を作成した．音脈分凝では周波数の組み合わせによって生じやすさが変わること，連続聴効果ではノイズの周波数にない音に対しては生じにくこと，ミッシングファンダメンタル波刺激では高調波成分のみでも基本周波数が聞き取れることを確認した．

第 6 章

関連語句

6.1 FIR フィルタ，IIR フィルタ

FIR，IIR はデジタルフィルタである．

■FIR フィルタ FIR（Finite Impluse Response）フィルタは，「インパルス応答が有限長である」という意味である [11, p.92]．インパルス応答とは，あるシステムにインパルスと呼ぶ短い信号を入力したときのシステムの出力を指す．インパルス応答 $h(k)$，入力データ $x(n)$ に対して，FIR フィルタの出力 $y(n)$ は，式 (6.1) である．

$$y(n) = \sum_{k=0}^{N-1} x(n-k) \cdot h(k) \tag{6.1}$$

FIR では，その名の通り，「有限」範囲での和になるので，N は有限である．位相歪*が許容されない場合や，安定性が求められるときには FIR が最適である．しかし，FIR は IIR に比べて演算回数が増え，回路設計により必要メモリ数が増えるというデメリットもある．

■IIR フィルタ IIR（Infinite Impluse Response）フィルタは，「インパルス応答が無限長である」という意味である [11, p.92].

$$y(n) = -\sum_{k=0}^{N} a_k \cdot y(n-k) + \sum_{k=1}^{M} b_k \cdot x(n-k) \tag{6.2}$$

IIR は FIR に比べてメモリ消費量を抑えられ，高速で演算量も抑えられる．また，周波数特性の滑らかさで決まる，インパルス応答の長が長いと FIR による演算量が膨大になる．このとき IIR を用いる．

■関連 振幅スペクトルに対して，フィルタ処理を行った 2.2 章，4.2 章に関連する．両方とも有限範囲に対するフィルタを適用したので，FIR を適用したと言える．

6.2 ノイズキャンセリング

ノイズキャンセリングは，「雑音除去」を表す．大きく 2 つあり，イヤフォンやヘッドフォンなどの音声出力装置についているものと，音声入力装置（マイク）についているものである．どちらも，外部雑音を除去するために用いられる．

■音声出力装置 イヤフォンやヘッドフォンに搭載されているノイズキャンセリングでは，イヤフォン外の雑音を一度取り込み，それに対して 3.2 章で扱ったように，各データと -1 の積を取り，その波をイヤフォンから出力することで，雑音を打ち消せる．

■音声入力装置 Apple 社の iOS 15 以降では，「声を分離」という機能がある．これは，入力された音声から雑音を取り除く技術である．これは音声入力装置のものとは異なり，雑音を機械学習でとらえ，除去する技術である．

*周波数によって遅延時間が異なること．

6.3　聴覚におけるマスキング

対象とする音（マスキ）の聴き取りが，別の音（マスカ）の存在によって妨害される現象．マスキングには 2 種類あり，同時マスキング，継時マスキングがある．

■同時マスキング　同時マスキングは，周波数軸上のマスキング効果．マスキとマスカが同時に提示されたとき，マスカに近いマスキの周波数が聴き取りにくく現象 [12]．図 6-1 中の着色部は，不可聴音の領域である．同時マスキングの特徴として，高音が低音よりも音量が大きい場合，高音は低音をマスクしにくいのに対して，低音が高音よりもマスクしやすいことがあげられる．

■継時マスキング　継時マスキングは，時間軸上のマスキング効果．マスカを提示した直前や直後のマスキが聴き取りにくくなる現象．図 6-2 中の着色部は，不可聴音の領域である．マスカ提示時刻より前のマスキングを逆行マスキング，マスカ提示時刻より後のマスキングを順行マスキングと呼ぶ．

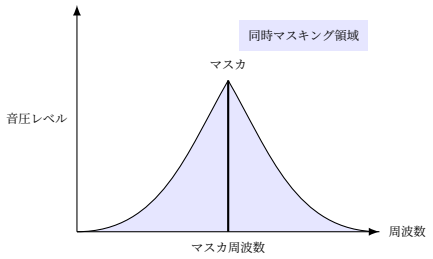


図 6-1: 同時マスキング

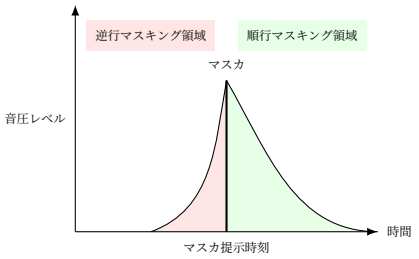


図 6-2: 継時マスキング

■関連　逆行マスキングに着目する．逆行マスキングは，何らかの刺激（マスカ）の直前の刺激を知覚できない現象である．これは，連続聴効果と関連付けることができる．（4.2）逆行マスキングの効果による，日常生活で聞こえない部分を，連続聴効果で補っていると考えられる．

6.4　音声認識

音声認識は，機械が音声をテキスへ変換するために用いられている技術である．音声認識は，大きく 2 つの処理を経る．入力音声に対して，音響モデルを用いて音響分析した後，言語モデルを用いて言語化する．

1. **音響分析**
   AI が扱いやすいデータにするための作業を音響分析という．背景音やノイズの除去をする．
2. **音響モデル**
   音声分析によって抽出したデータを，AI が過去のデータをもとに音素を抽出する．たとえば，3.3 章の図 3-7(b) と図 3-8(b) 比べると，母音によって含まれる高調波の周波数が異なる（3.3 章）．この違いをもとに音素*を抽出する．
3. **発音辞書**
   抽出した音素を，単語に構成する過程．
4. **言語モデル**
   言葉を，単語をもとに生成する．言語モデルは，単語の出現確率でモデル化したもの．

[13]

*音素とは，音韻の最小単位．例えば，「カ」の音素は `k` と `a` である．

参考文献

[1] 飯田一博. 音響工学基礎論. コロナ社, 2012.

[2] 岩田 修一（総監修）. かたち創造の百科事典. 丸善出版, 2012.

[3] 和田成夫. よくわかる信号処理 ―フーリエ解析からウェーブレット変換まで―. 森北出版, 2009.

[4] Tudor Barbu. Variational image denoising approach with diffusion porous media flow. In *Abstract and Applied Analysis*, Vol. 2013. Hindawi, 2013.

[5] 日本機械学会機械工学事典. 白色雑音. https://www.jsme.or.jp/jsme-medwiki/doku.php?id=13:1010098, 2017.

[6] MathWorks. 指定の平均値と分散を持つ正規分布からの乱数. https://jp.mathworks.com/help/matlab/math/random-numbers-with-specific-mean-and-variance.html, Confirmation date: May 6th, 2023.

[7] 花泉弘, 小泉悠, 伊藤克. MATLAB で学ぶ実践画像・音声処理入門. コロナ社, 2019.

[8] Mathworks. rand. https://jp.mathworks.com/help/matlab/ref/rand.html, Confirmation date: May 6th, 2023.

[9] 海保博之（監修）, 菊地正（編）. 感覚知覚心理学（朝倉心理学講座）. 朝倉心理学講座. 朝倉書店, 2008.

[10] NTT Communication Science Laboratories. ミッシング・ファンダメンタル. https://illusion-forum.ilab.ntt.co.jp/missing-fndamental/index.html, Confirmation date: May 6th, 2023.

[11] 相川清明, 大渕康成. 音声音響インタフェース実践（メディア学大系）. メディア学大系. コロナ社, 2017.

[12] 桑野園子. マスキングと騒音対策. 騒音制御, Vol. 29, No. 4, pp. 241–246, 2005.

[13] Ltd. Human Science Co. Ai 音声認識とは. https://www.science.co.jp/annotation_blog/32604/, Confirmation date: May 6th, 2023.

付録

A　音の工学的特徴 (April 13th, 2023)

src. A-1: 周波数の異なる純音の生成（第 1.1 章 → p.1）

```
clear;
Fs = 16000; %
f1 = 440;   % [Hz]
f2 = 660;   % [Hz]

t1 = (0:(Fs-1)) /Fs;
y1 = sin(2*pi*f1*t1);
t2 = (0:(Fs-1)) /Fs;
y2 = sin(2*pi*f2*t2);

sound(y1,Fs);
pause(3)
sound(y2,Fs);

fig0 = figure;
hold on;
plot(t1,y1);
plot(t2,y2);
hold off;
xlabel('time[s]');
ylabel('amplitude');
legend({'440Hz','660Hz'},'Location','southwest');
axis([0 0.01 -1.01 1.01]);
exportgraphics(fig0,'../Figures/01_01.pdf','ContentType','vector');
```

src. A-2: 振幅・位相の確認（第 1.2 章 → p.2）

```
clear;
Fs = 16000; %
f1 = 440;   % [Hz]
t1 = (0:(Fs-1)) /Fs;
for k=0:Fs
    y1 = sin(2*pi*f1*t1);
end
p = pi/2;
for k=0:Fs
    y2 = sin(2*pi*f1*t1 + p);
end

p = pi;
for k=0:Fs
    y3 = sin(2*pi*f1*t1 + p);
```

```
end

for k=0:Fs
    y4 = 0.5 * sin(2*pi*f1*t1);
end

for k=0:Fs
    y5 = 0.25 * sin(2*pi*f1*t1);
end

ys2 = [y1;y2]';
ys3 = [y1;y3]';
ys4 = [y1;y4]';
ys5 = [y1;y5]';

sound(ys2,Fs); % pi/2
pause(3)
sound(ys3,Fs); % pi
pause(3)
sound(ys4,Fs); % 0.5倍
pause(3)
sound(ys5,Fs); % 0.25倍

fig0 = figure;
hold on;
plot(t1,y1);
plot(t1,y2);
plot(t1,y3);
xlabel('time[s]');
ylabel('amplitude');
axis([0 0.01 -1.01 1.01])
legend({'pure tone','shift by 90 degrees','shift by 180 degrees'},'Location','southwest');
hold off;

fig1 = figure;
hold on;
plot(t1,y1);
plot(t1,y4);
plot(t1,y5);
yticks(-1:0.25:1);
xlabel('time[s]');
ylabel('amplitude');
axis([0 0.01 -1.1 1.1]);
legend({'pure tone','1/2 times amplitude','1/4 times amplitude'},'Location','southwest');
hold off;
exportgraphics(fig1,'../Figures/01_02_1.pdf','ContentType','vector');
exportgraphics(fig0,'../Figures/01_02_2.pdf','ContentType','vector');
```

src. A-3: うなり（第 1.3 章 → p.4）

```
clear;
Fs = 16000; %
f1 = 440;   % [Hz]
f2 = 441;   % [Hz]
t = (0:4*(Fs-1)) /Fs;
for k=0:Fs
```

```
    y1 = sin(2*pi*f1*t);
end

for k=0:Fs
    y2 = sin(2*pi*f2*t);
end

y = y1 + y2;
sound(y,Fs);

fig0 = figure;
plot(t,y);
xticks(0:1:4);
xlabel('time[s]');
ylabel('amplitude');
axis([0 4 -2.01 2.01]);
exportgraphics(fig0,'../Figures/01_03.pdf','ContentType','vector');
```

src. A-4: フーリエ級数展開（第 1.4 章 → p.4）

```
clear;
Fs = 16000;
T = 1/2; % 周期
f = 1/T; % 周波数
t = (0:4*(Fs-1)) /Fs;

y = 0;
for k=1
    y = y + (1/(2*k-1)) * sin(2*pi*f*(2*k-1)*t);
end

fig0 = figure;
hold on;
plot(t,y);
y = 0;
for k=1:5
    y = y + (1/(2*k-1)) * sin(2*pi*f*(2*k-1)*t);
end
plot(t,y);
y = 0;
for k=1:25
    y = y + (1/(2*k-1)) * sin(2*pi*f*(2*k-1)*t);
end
plot(t,y);
axis([0 1 -3 3]);
xlabel('time[s]');
ylabel('振幅');
hold off;
legend({'N=1','N=5','N=25'},'Location','southwest');
fig1 = figure;
y = 0;
for k=1:50
    y = y + (1/(2*k-1)) * sin(2*pi*f*(2*k-1)*t);
end
plot(t,y);
axis([0 1 -3 3]);
```

```
xlabel('time[s]');
ylabel('amplitude');
exportgraphics(fig0,'../Figures/01_04_2.pdf','ContentType','vector');
exportgraphics(fig1,'../Figures/01_04_1.pdf','ContentType','vector');
```

src. A-5: 白色ガウス雑音（第 1.5 章 → p.5）

```
clear;
Fs = 16000;
t = (0 : (Fs-1)) /Fs;
tl = length(t);
rng(0,'twister'); % 乱数生成器 初期化
a = 0.2; % 標準偏差
b = 0; % 平均
y = a.*randn(tl,1) + b;

fig0 = figure;
plot(t,y);
xlabel('time[s]');
ylabel('amplitude');
num=100; % ヒストグラム分割数
[h, c] = hist(y, num);

fig1 = figure;
plot(c,h);
xlabel('amplitude');
ylabel('frequency[times]');

sound(y,Fs);
exportgraphics(fig0,'../Figures/01_05_0.pdf','ContentType','vector');
exportgraphics(fig1,'../Figures/01_05_1.pdf','ContentType','vector');
```

B 周波数解析 (April 17th, 2023)

src. B-1: フーリエ変換と周波数解析（第 2.1 章 → p.7）（純音）

```
clear;
Fs = 8192;
t = (0:(Fs-1)) / Fs;
f = 440;
fs = (-Fs/2 : Fs/1024 : (Fs/2) - Fs/1024);
y = sin(2*pi*f*t);

fft_y = fft(y,1024);
ln = length(abs(fftshift(fft(y))));
fft_ys = fftshift(fft_y);
fft_ys = abs(fft_ys);
fig0 = figure;
plot(fs,fft_ys);
xlabel('frequency[Hz]');
ylabel('|amplitude|');
axis([0 1000 -10 600]);
exportgraphics(fig0,'../Figures/02_01.pdf','ContentType','vector');
```

src. B-2: フーリエ変換と周波数解析（第 2.1 章 → p.7）（矩形波）

```
clear;
Fs = 8192;
t = (0 : 4*(Fs-1)) / Fs;
f = 440;
fs = (-Fs/2 : Fs/(128*2*4) : (Fs/2)-Fs/(128*2*4));
y_0 = zeros(1,128);
y_1 = ones(1,128);
y = [y_0 y_1];
y = [y y y y];

fig0 = figure;
plot(y);
xlabel('point');
ylabel('amplitude');
axis([0 1024 -0.5 1.5]);

fft_y = fft(y);
ln = length(abs(fftshift(fft(y))));
fft_ys = fftshift(fft_y);
fft_ys = abs(fft_ys);
fig1 = figure;
plot(fs,fft_ys);
xlabel('frequency[Hz]');
ylabel('|amplitude|');
axis([0 200 -10 600]);
exportgraphics(fig0,'../Figures/02_021.pdf','ContentType','vector');
exportgraphics(fig1,'../Figures/02_022.pdf','ContentType','vector');
```

src. B-3: LPF（ローパスフィルタ）（第 2.2 章 → p.9）

```
clear;

Fs = 8192;
t = (0 : 0.25*(Fs-1)) /Fs;
t8 = (0 : 2*(Fs)-1) /Fs;

f0 = 261.38;
f1 = 293.67;
f2 = 329.63;
f3 = 349.23;
f4 = 392.00;
f5 = 440.00;
f6 = 493.88;
f7 = 523.23;
y_0 = sin(2 * pi * f0 * t);
y_1 = sin(2 * pi * f1 * t);
y_2 = sin(2 * pi * f2 * t);
y_3 = sin(2 * pi * f3 * t);
y_4 = sin(2 * pi * f4 * t);
y_5 = sin(2 * pi * f5 * t);
y_6 = sin(2 * pi * f6 * t);
y_7 = sin(2 * pi * f7 * t);
y = [y_0 y_1 y_2 y_3 y_4 y_5 y_6 y_7];

```

```
fig0 = figure;
plot(t8,y);
xticks(0:0.25:2);
xlabel('time[s]');
ylabel('amplitude');
axis([0 2 -1.5 1.5]);

fft_y = fft(y);
fft_y = fftshift(fft_y);
f = (-Fs/2 : Fs/length(fft_y)  : Fs/2 - Fs/length(fft_y));

fig1 = figure;
subplot(2,1,1);
plot(f,fft_y);
xlabel('frequency[Hz]');
ylabel('amplitude');
axis([-1000 1000 -1500 1500]);

for k=1:length(fft_y)
    if f(k) >= 375.00 || f(k) <= -375.00
        fft_y(k) = 0;
    end
end
subplot(2,1,2);
plot(f,fft_y);
axis([-1000 1000 -1500 1500]);
xlabel('frequency[Hz]');
ylabel('amplitude');
ifft_y = ifftshift(fft_y);
ifft_y = ifft(ifft_y);
ifft_real = real(ifft_y);

fig2 = figure;
plot(t8,ifft_y);
xlabel('time[s]');
ylabel('amplitude');
xticks(0:0.25:2);
axis([0 2 -1.5 1.5]);
sound(ifft_real,Fs)
exportgraphics(fig0,'../Figures/02_20.pdf','ContentType','vector');
exportgraphics(fig1,'../Figures/02_21.pdf','ContentType','vector');
exportgraphics(fig2,'../Figures/02_22.pdf','ContentType','vector');
```

C　音声合成 (April 20th, 2023)

src. C-1: 合成波における位相の操作（第 3.1 章 → p.11）（矩形波）

```
clear;
Fs = 16000;
f = 440;
t = (0:2*Fs-1) /Fs;
phi1 = pi/4;
phi2 = pi/2;

y1=0;
```

```
y2=0;
y3=0;
y4=0;
% 矩形波
for k = 1:50
    y1 = y1 + 1/(2*k-1) * sin(2*pi*f*(2*k-1)*t);
    y2 = y2 + 1/(2*k-1) * sin(2*pi*f*(2*k-1)*t+phi1);
    y3 = y3 + 1/(2*k-1) * sin(2*pi*f*(2*k-1)*t+phi2);
    y4 = y4 + 1/(2*k-1) * sin(2*pi*f*(2*k-1)*t+rand);
end

sound(y1,Fs);
pause(3);
sound(y2,Fs);
pause(3);
sound(y3,Fs);
pause(3);
sound(y4,Fs);
pause(3);

fig0 = figure;
plot(t,y1);
xlabel('time[s]');
ylabel('amplitude');
axis([0 0.01 -3 3]);
exportgraphics(fig0,'../Figures/03_11_kukei.pdf','ContentType','vector');

fig1 = figure;
plot(t,y2);
xlabel('time[s]');
ylabel('amplitude');
axis([0 0.01 -3 3]);
exportgraphics(fig1,'../Figures/03_12.pdf','ContentType','vector');

fig2 = figure;
plot(t,y3);
xlabel('time[s]');
ylabel('amplitude');
axis([0 0.01 -3 3]);
exportgraphics(fig2,'../Figures/03_13.pdf','ContentType','vector');

fig3 = figure;
plot(t,y4);
xlabel('time[s]');
ylabel('amplitude');
axis([0 0.01 -3 3]);
exportgraphics(fig3,'../Figures/03_14.pdf','ContentType','vector');
```

src. C-2: 合成波における位相の操作（第 3.1 章 → p.11）（ノコギリ波）

```
clear;
Fs = 16000; % サンプリング周波数
f = 440; % 基本周波数
t = (0 : 2*Fs-1) /Fs; % 時刻
phi1 = pi / 4;
phi2 = pi / 2;
```

```
7
8  y1 = 0;
9  y2 = 0;
10 y3 = 0;
11 y4 = 0;
12 % ノコギリ波
13 for k = 1 : 50
14     y1 = y1 + (-1)^(k-1) * 1/3 * 2/k * sin(2*pi*f*k*t);       % 級数を表現
15     y2 = y2 + (-1)^(k-1) * 1/3 * 2/k * sin(2*pi*f*k*t+phi1);
16     y3 = y3 + (-1)^(k-1) * 1/3 * 2/k * sin(2*pi*f*k*t+phi2);
17     y4 = y4 + (-1)^(k-1) * 1/3 * 2/k * sin(2*pi*f*k*t+rand);
18 end
19
20 sound(y1,Fs);
21 pause(3);
22 sound(y2,Fs);
23 pause(3);
24 sound(y3,Fs);
25 pause(3);
26 sound(y4,Fs);
27 pause(3);
28
29 fig1 = figure;
30 plot(t,y1);
31 xlabel('time[s]');
32 ylabel('amplitude');
33 axis([0 0.01 -3 3]);
34 exportgraphics(fig1,'../Figures/03_21_nokogiri.pdf','ContentType','vector');
35
36 fig2 = figure;
37 plot(t,y2);
38 xlabel('time[s]');
39 ylabel('amplitude');
40 axis([0 0.01 -3 3]);
41 exportgraphics(fig2,'../Figures/03_22.pdf','ContentType','vector');
42
43 fig3 = figure;
44 plot(t,y3);
45 xlabel('time[s]');
46 ylabel('amplitude');
47 axis([0 0.01 -3 3]);
48 exportgraphics(fig3,'../Figures/03_23.pdf','ContentType','vector');
49
50 fig4 = figure;
51 plot(t,y4);
52 xlabel('time[s]');
53 ylabel('amplitude');
54 axis([0 0.01 -3 3]);
55 exportgraphics(fig4,'../Figures/03_24.pdf','ContentType','vector');
```

src. C-3: 録音した音声の操作（第 3.2 章 → p.12）（母音 a）

```
1 clear;
2 [y, Fs] = audioread('sound2.wav');
3 y = y(:,1); % ステレオからモノラルへの変換
4 N = length(y); % yの長さ
```

```
t = (1:N) /Fs; % 時間
z = y.*(-1);

soundsc(y,Fs);
pause(5);
soundsc(z,Fs);

fig = figure;
hold on;
plot(t,y);
plot(t,z);
plot(t,y+z);
hold off;
legend({'1) original wave','2) sign inversion wave','3) synthetic wave'},'Location','southwest
  ');
xlabel('time[s]');
ylabel('amplitude');
axis([0 0.01 -0.04 0.04]);
exportgraphics(fig,'../Figures/03_2_a.pdf','ContentType','vector');
```

src. C-4: 録音した音声の操作（第 3.2 章 → p.12）（母音 i）

```
clear;
[y, Fs] = audioread('sound1.wav');
y=y(:,1); % ステレオからモノラルへの変換
N = length(y); % yの長さ
t = (1:N) /Fs; % 時間
z = y.*(-1);

soundsc(y,Fs);
pause(5);
soundsc(z,Fs);

fig = figure;
hold on;
plot(t,y);
plot(t,z);
plot(t,y+z);
hold off;
legend({'1) original wave','2) sign inversion wave','3) synthetic wave'},'Location','southwest
  ');
xlabel('time[s]');
ylabel('amplitude');
axis([0 0.01 -0.04 0.04]);
exportgraphics(fig,'../Figures/03_2_i.pdf','ContentType','vector');
```

src. C-5: 音声合成（第 3.3 章 → p.13）

```
clear;
[y1,Fs1] = audioread('sound1.wav');
[y2,Fs2] = audioread('sound2.wav');

y1 = y1(:,1);
y2 = y2(:,1);

t1 = (0 : length(y1)-1) /Fs1;
```

```
t2 = (0 : length(y2)-1) /Fs2;

figa = figure;
plot(t1, y1);
xlabel('time[s]');
ylabel('amplitude');
axis([0 1 -0.04 0.04]);
exportgraphics(figa,'../Figures/03_30_0a.pdf','ContentType','vector');

figi = figure;
plot(t2, y2);
xlabel('time[s]');
ylabel('amplitude');
axis([0 1 -0.04 0.04]);
exportgraphics(figi,'../Figures/03_30_0i.pdf','ContentType','vector');

y1l = length(y1);
y2l = length(y2);
fs1 = (-Fs1/2 : Fs1/y1l : (Fs1/2)-Fs1/y1l);
fs2 = (-Fs2/2 : Fs2/y2l : (Fs2/2)-Fs2/y2l);

Y1 = fft(y1);
Y2 = fft(y2);
Y1 = fftshift(Y1);
Y2 = fftshift(Y2);

Y1=abs(Y1);
Y2=abs(Y2);

fig0 = figure; % [a_fft]
plot(fs1, Y1);
axis([0 10000 -10 80]);
xlabel('frequency[Hz]');
ylabel('amplitude');
exportgraphics(fig0,'../Figures/03_30_afft.pdf','ContentType','vector');

fig1 = figure; % [i_fft]
plot(fs2, Y2);
axis([0 10000 -10 80]);
xlabel('frequency[Hz]');
ylabel('amplitude');
exportgraphics(fig1,'../Figures/03_31_ifft.pdf','ContentType','vector');

y1mx = [1092 545 819 682 956 2868 3555 136 1219 3680];
y1my = [77 51 50 42 36 35 30.3459 25 25 21];
y2mx = [306 2906 2447 153 2884 2442 2980 2436 302 2419];
y2my = [52 52 44 41 34 34 41 33 38 31];
ya = 0; yi = 0;

for k=1:10
   ya = ya + y1my(k) * sin(2*pi*y1mx(k)*t1);
   yi = yi + y2my(k) * sin(2*pi*y2mx(k)*t2);
end

fig2 =  figure;
plot(t1, ya);
```

```
xlabel('time[s]');
ylabel('amplitude');
axis([0 1 -400 400]);
exportgraphics(fig2,'../Figures/03_32_a.pdf','ContentType','vector');

fig3 = figure;
plot(t2, yi);
xlabel('time[s]');
ylabel('amplitude');
axis([0 1 -400 400]);
exportgraphics(fig3,'../Figures/03_33_i.pdf','ContentType','vector');

sound(ya,Fs1);
pause(4);
sound(yi,Fs2);
```

D　聴覚と音声信号 (April 24th, 2023)

src. D-1: 音脈分凝を生じる刺激の作成（第 4.1 章 → p.15）

```
clear;

Fs = 16000;
f1 = 1000;
f2 = 990;
f3 = 800;
f4 = 200;
t = (0 : (0.2*Fs-1)) /Fs; % 0.1秒時間軸
ta = (0 : (4*Fs-1)) /Fs; % 0.1 * 20 = 2秒時間軸

y1 = sin(2*pi*f1*t);
y2 = sin(2*pi*f2*t);
y3 = sin(2*pi*f3*t);
y4 = sin(2*pi*f4*t);

EX1 = [y1 y2];
EX1_rep = repmat(EX1, 1, 10);
EX2 = [y1 y3];
EX2_rep = repmat(EX2, 1, 10);
EX3 = [y1 y4];
EX3_rep = repmat(EX3, 1, 10);

sound(EX1_rep,Fs);
pause(5);
sound(EX2_rep,Fs);
pause(5);
sound(EX3_rep,Fs);

fig0 = figure;
plot(ta, EX1_rep);
xlabel('time[s]');
ylabel('amplitude');

fig1 = figure;
plot(ta, EX2_rep);
```

```
xlabel('time[s]');
ylabel('amplitude');
fig2 = figure;
plot(ta, EX3_rep);
xlabel('time[s]');
ylabel('amplitude');
```

src. D-2: 連続聴効果を生じる音刺激の作成（第 4.2 章 → p.16）

```
clear;
Fs = 16000;
freq = (-Fs/2 : 1 : Fs/2 - 1);
f = 400;
f2 = 1300;
t_noise= (0 : 0.1*Fs-1) /Fs;
t_y = (0 : Fs-1) /Fs;
y = sin(2*pi*f*t_y);
y2 = sin(2*pi*f2*t_y);
t_all = (0 : (4*Fs + 0.4*Fs - 1));
noise_A = 2 * (rand(1,Fs) - 0.5);

A_fft = fft(noise_A);
A_fft = fftshift(A_fft);

fig0 = figure; % ノイズA_fft
plot(freq,A_fft);
xlabel('frequency[Hz]');
ylabel('amplitude');

noise_filter = ones(1,Fs);
filter_rangeS = 1020;
filter_rangeG = 1620;
noise_filter((Fs/2)-filter_rangeG : (Fs/2)-filter_rangeS) = 0;
noise_filter((Fs/2)+filter_rangeS : (Fs/2)+filter_rangeG) = 0;
B_fft = noise_filter.* A_fft;

fig1 = figure; % ノイズB_fft
plot(freq,B_fft);
xlabel('frequency[Hz]');
ylabel('amplitude');

B_fft = ifftshift(B_fft);
noise_B = ifft(B_fft);
noise_B = real(noise_B);

noise_A = noise_A(1:1600);
noise_B = noise_B(1:1600);

fig2 = figure; % ノイズA
plot(t_noise, noise_A);
xlabel('time[s]');
ylabel('amplitude');
fig3 = figure; % ノイズB
plot(t_noise,noise_B);
xlabel('time[s]');
ylabel('amplitude');
```

```

EX1 = [y noise_A];
EX1 = repmat(EX1, 1, 4);
EX2 = [y noise_B];
EX2 = repmat(EX2, 1, 4);
EX3 = [y2 noise_A];
EX3 = repmat(EX3, 1, 4);
EX4 = [y2 noise_B];
EX4 = repmat(EX4, 1, 4);

fig4 = figure;
plot(t_all,EX1);
xlabel('time[s]');
ylabel('amplitude');

soundsc(EX1,Fs);    % pure tone A + noise A
pause(5);
soundsc(EX2,Fs);    % pure tone A + noise B
pause(5);
soundsc(EX3,Fs);    % pure tone B + noise A
pause(5);
soundsc(EX4,Fs);    % pure tone B + noise B

exportgraphics(fig0,'../Figures/04_20_Afft.pdf','ContentType','vector');
exportgraphics(fig1,'../Figures/04_21_Bfft.pdf','ContentType','vector');
exportgraphics(fig2,'../Figures/04_22_Anoise.pdf','ContentType','vector');
exportgraphics(fig3,'../Figures/04_23_Bnoise.pdf','ContentType','vector');
exportgraphics(fig4,'../Figures/04_24_gosei.pdf','ContentType','vector');
```

src. D-3: ミッシングファンダメンタル波刺激の作成（第 4.3 章 → p.17）

```
clear;
Fs = 16000;
f = 440;
t = (0 : Fs-1) /Fs;
y_kukei = 0;
y_nokogiri = 0;
for k=1:50
    y_kukei = y_kukei + (1/(2*k-1)) * sin(2*pi*f*(2*k-1)*t);
end
mfm_kukei = y_kukei - sin(2*pi*f*t);

for k=1:50
    y_nokogiri = y_nokogiri + (-1)^(k-1) * 1/3 * 2/k * sin(2*pi*f*k*t);
end
mfm_nokogiri = y_nokogiri - 1/3*2*sin(2*pi*f*t);
y = sin(2*pi*f*t);

fig0 = figure;
hold on;
plot(t,y_kukei);
plot(t,mfm_kukei);
hold off;
legend({'original wave','missing fundamental wave'},'Location','southwest');
axis([0 0.01 -1.5 1.5]);
xlabel('time[s]');
```

```
26 ylabel('amplitude');
27
28 fig1 = figure;
29 hold on;
30 plot(t,y_nokogiri);
31 plot(t,mfm_nokogiri);
32 hold off;
33 legend({'original wave','missing fundamental wave'},'Location','southwest');
34 axis([0 0.01 -1.5 1.5]);
35 xlabel('time[s]');
36 ylabel('amplitude');
37
38 sound(y_kukei,Fs);
39 pause(2);
40 sound(mfm_kukei,Fs);
41 pause(2);
42 sound(y_nokogiri,Fs);
43 pause(2);
44 sound(mfm_nokogiri,Fs);
45 pause(2);
46
47 exportgraphics(fig0,'../Figures/04_30_kukei.pdf','ContentType','vector');
48 exportgraphics(fig1,'../Figures/04_31_nokogiri.pdf','ContentType','vector');
```