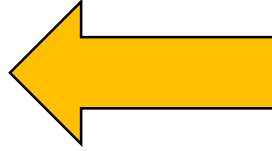


第6回
サウンドメディア論
および演習 講義編

今日やること

- 続・音を広げる



- オートパン
- 疑似ステレオ化

- 音を削る

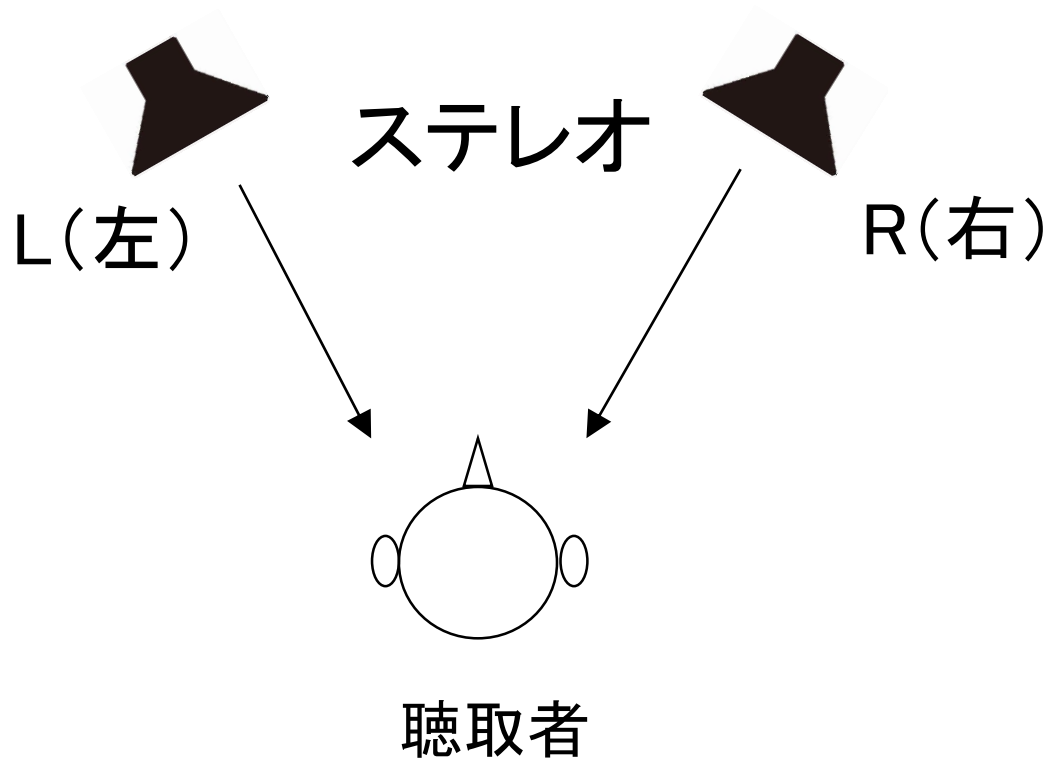
ボーカルキャンセラ

- 音の高さを上下させる

リサンプリング

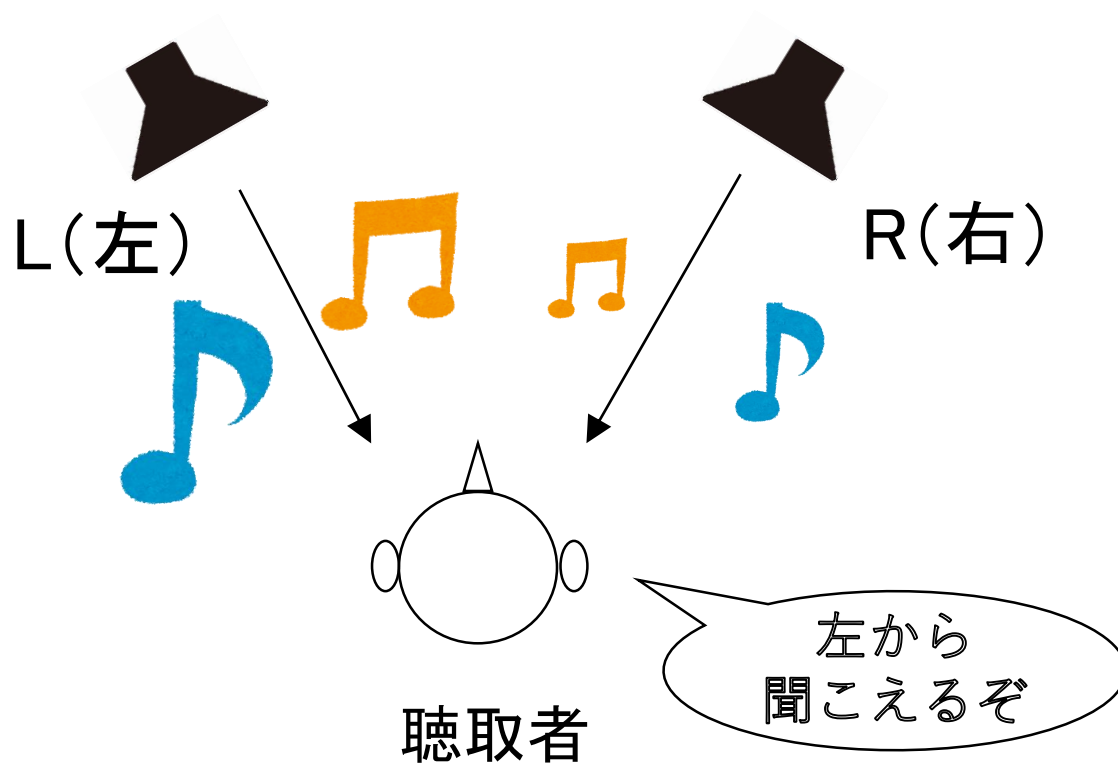
ステレオ再生

- スピーカを2つ用意して音データを再生



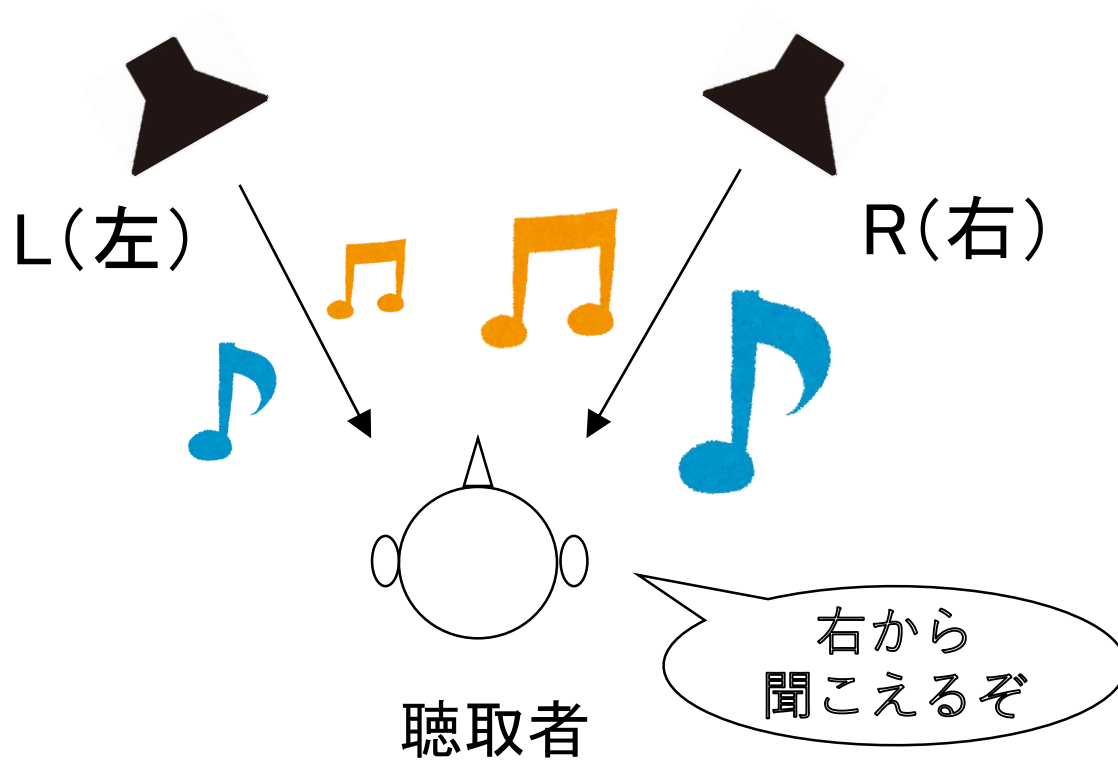
ステレオ再生の効果

- Lチャンネルの音が大きい場合
⇒ 音源が左にあるように知覚



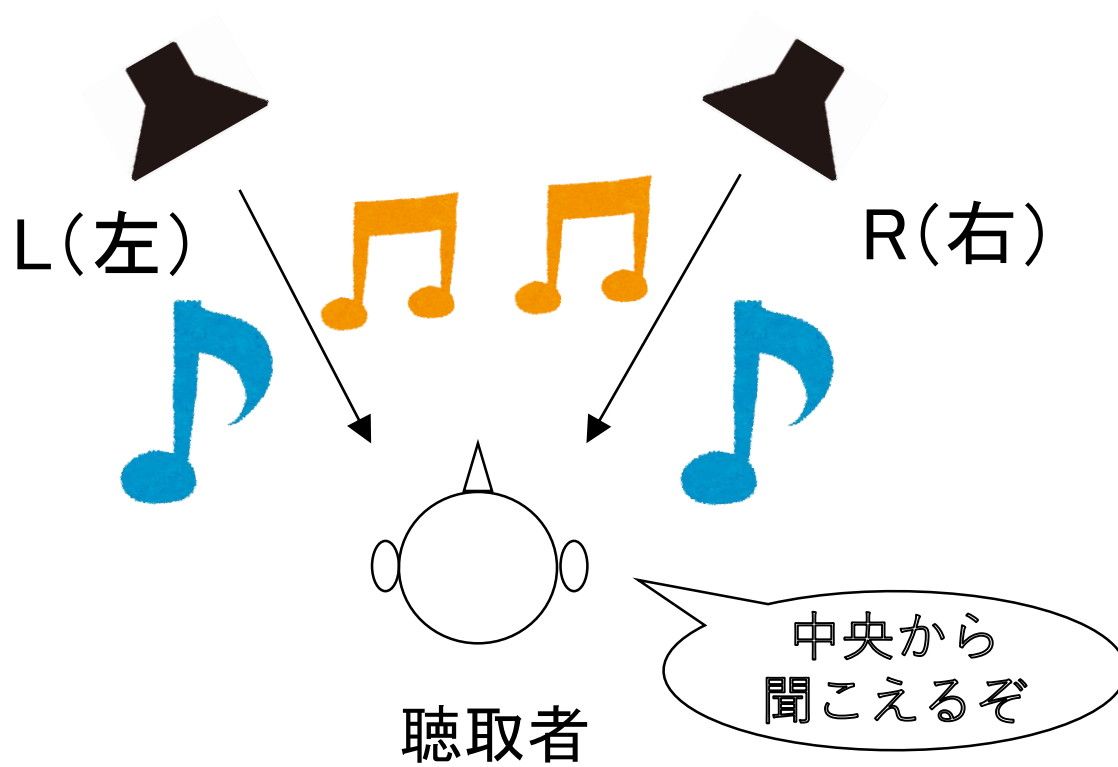
ステレオ再生の効果

- Rチャンネルの音が大きい場合
⇒ 音源が右にあるように知覚



ステレオ再生の効果

- どちらのチャンネルも音の大きさが等しい場合
⇒ 音源が中央にあるように知覚



ステレオ再生の効果

- 人間の聴覚は左右の耳から聞こえてくる音の大きさを手がかりの一つとして、**音源の位置を判断**
- ステレオ再生は聴覚の仕組みを利用して音の空間的な広がりを演出するための仕掛け



音像定位とオートパン

- 音像

音源の位置のイメージ

- 音像定位

音像を知覚し、音源の位置を判断すること

- オートパンとは？

- トレモロを応用したサウンドエフェクト
- 自動的に音像を左右に動かして音像定位を誘起
⇒ 空間的な音の広がりを演出

※「パン」は「パノラマ (panorama)」から

音像定位のデモンストレーション動画

- YouTubeで見つけました

イヤホン必須なので、個人で視聴して下さい

<https://www.youtube.com/watch?v=NOvJSTQhcyM>

オートパンの詳細

モノラルの音データに対してトレモロをかける

- Lチャンネル

$$s_{1L}(n) = a_L(n)s_0(n)$$

$$a_L(n) = 1 + \text{depth} \cdot \sin\left(\frac{2\pi \cdot \text{rate} \cdot n}{F_s}\right)$$

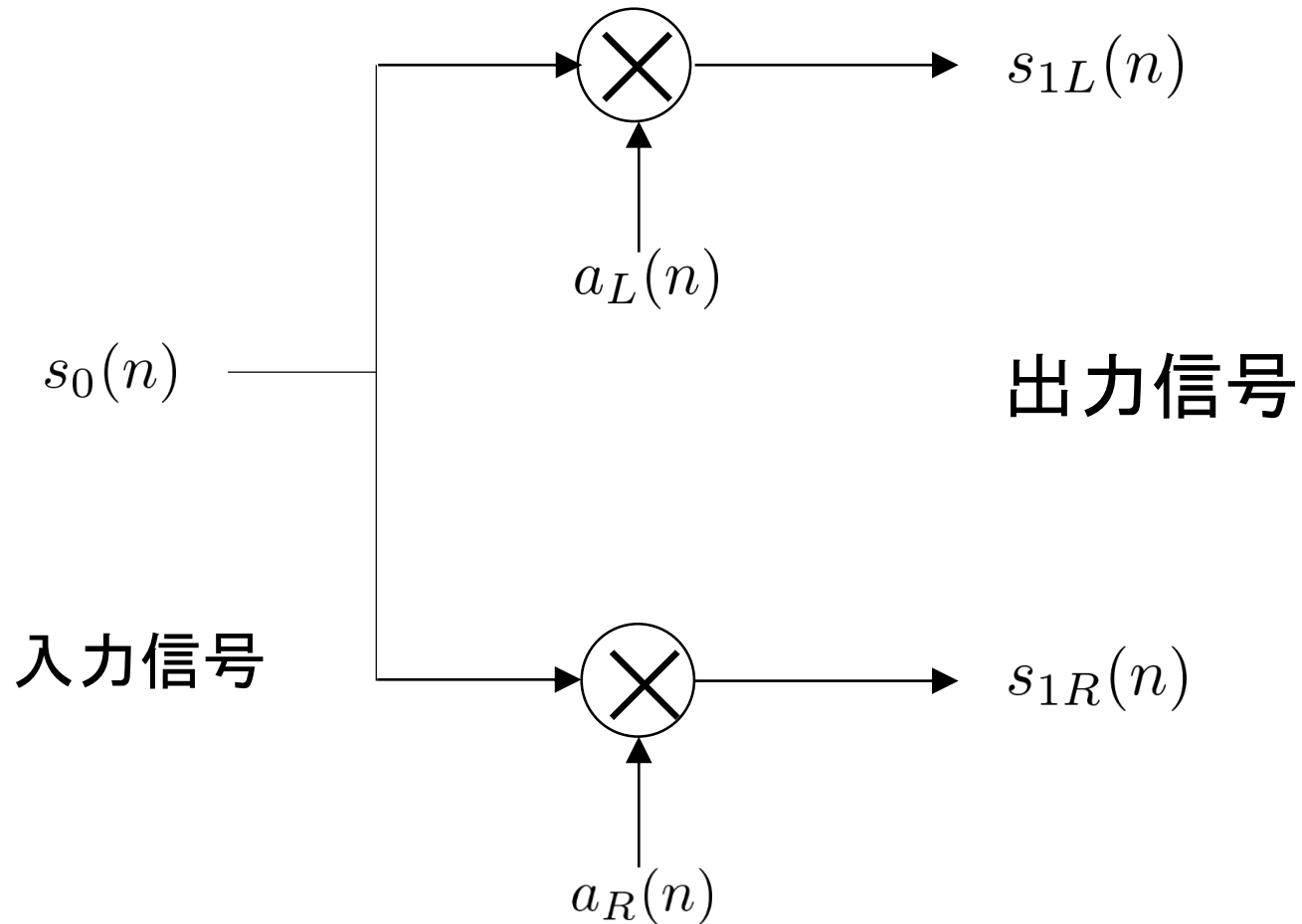
- Rチャンネル

$$s_{1R}(n) = a_R(n)s_0(n)$$

$$a_R(n) = 1 - \text{depth} \cdot \sin\left(\frac{2\pi \cdot \text{rate} \cdot n}{F_s}\right)$$

音の大小が互い違いに変化

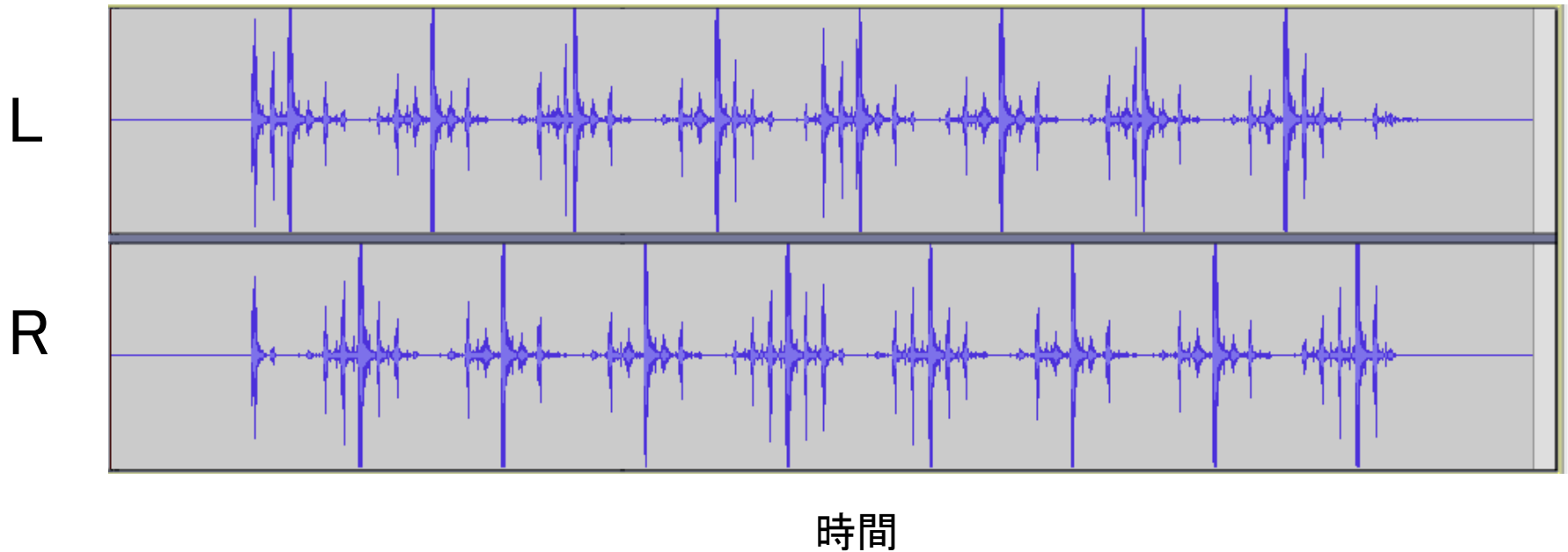
オートパンのブロック図



聴取者は音像があたかも左右を行ったり来たりしているように知覚する

オートパンのデモンストレーション

スピーカーだと正直分かりません



オートパンをかける前



オートパンをかけた後

オートパンの実装

疑似コード：トレモロと類似

N : 音データの長さ, Fs: 標本化周波数, s0 : 入力の音データ

s1L : 出力の音データ (Lチャンネル)

s1R : 出力の音データ (Rチャンネル)

depth \leftarrow 揺れの深さ, rate \leftarrow 揺れの細かさ

aL : トレモロの振幅変調信号 (Lチャンネル)

aR : トレモロの振幅変調信号 (Rチャンネル)

for n = 0 to N -1 **do** // 各時刻のデータに対して処理

 aL \leftarrow (スライドを参考に計算)

 aR \leftarrow (スライドを参考に計算)

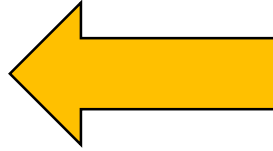
 s1L(n) \leftarrow aL * s0(n)

 s1R(n) \leftarrow aR * s0(n)

end for

今日やること

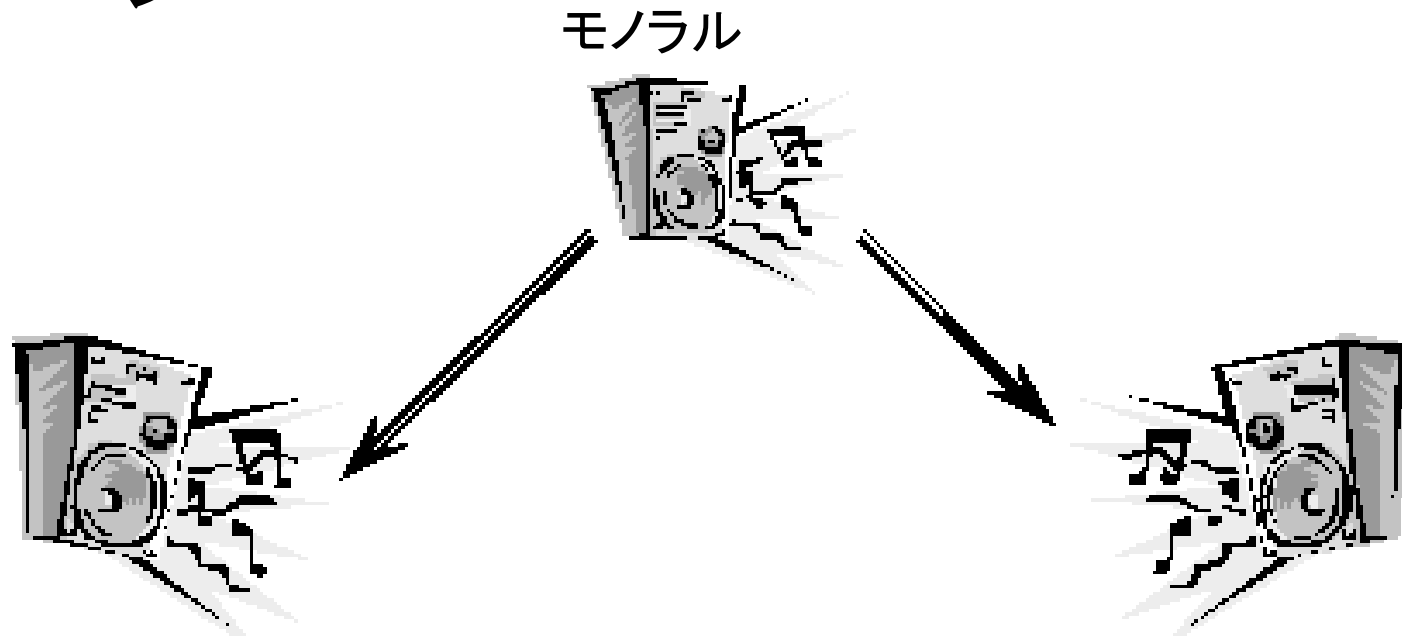
- 続・音を広げる
 - オートパン
 - 疑似ステレオ化



- 音を削る
ボーカルキャンセラ
- 音の高さを上下させる
リサンプリング

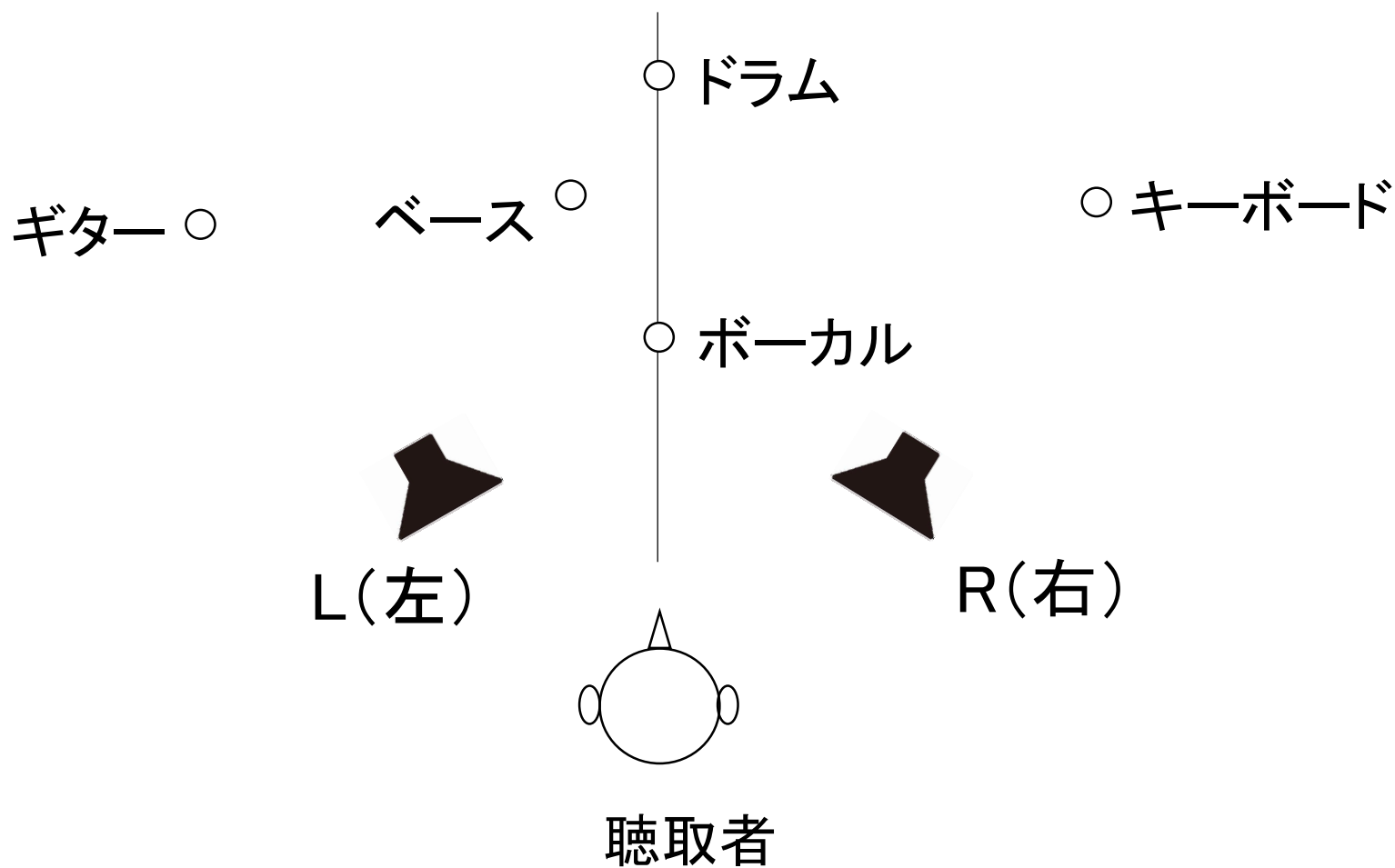
疑似ステレオ化とは？

- モノラルの音データをステレオに変換
⇒ 空間的な音の広がりを演出
- イメージ



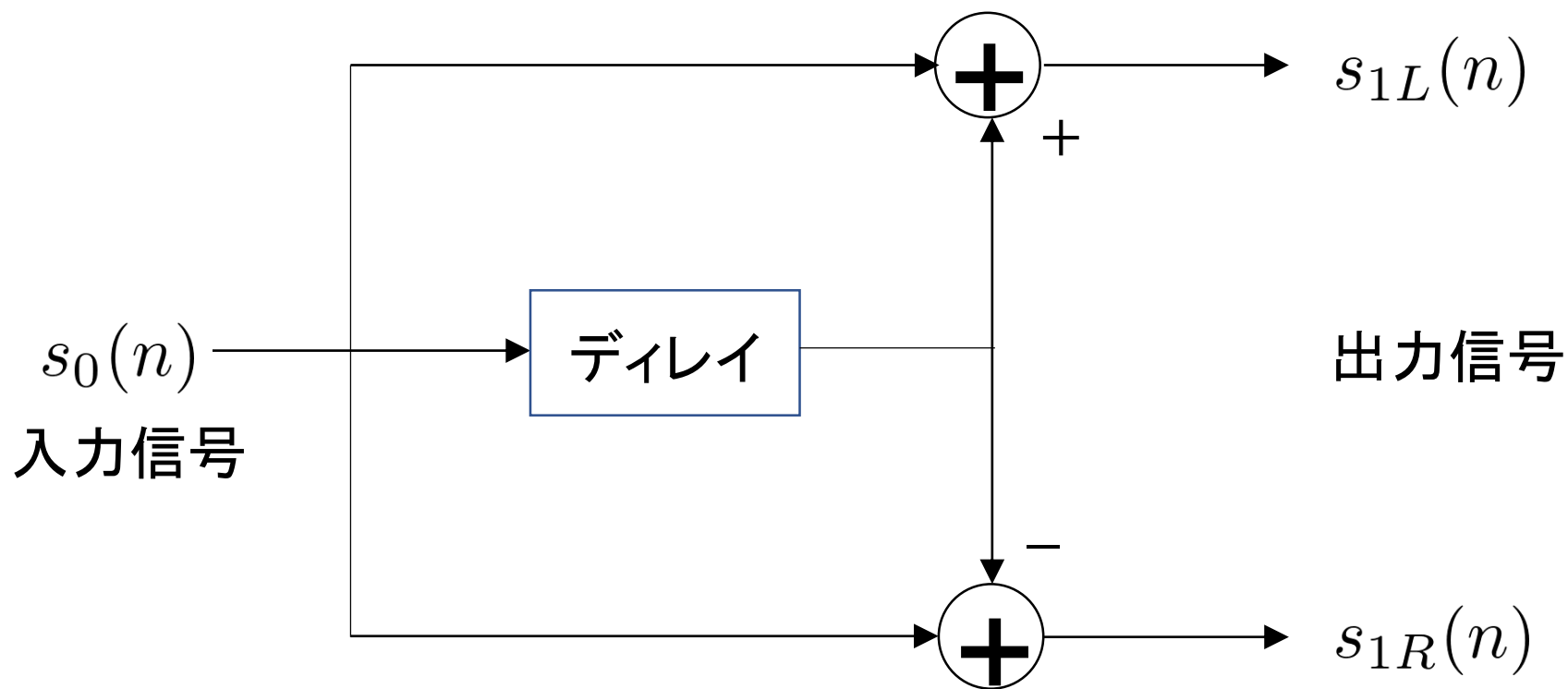
音楽CDのステレオデータの音像定位

ステージ上の演奏者の位置などを考慮し、臨場感のある音像定位となるように左右のチャンネルの音量を調整



疑似ステレオ化の詳細(1/2)

ブロック図



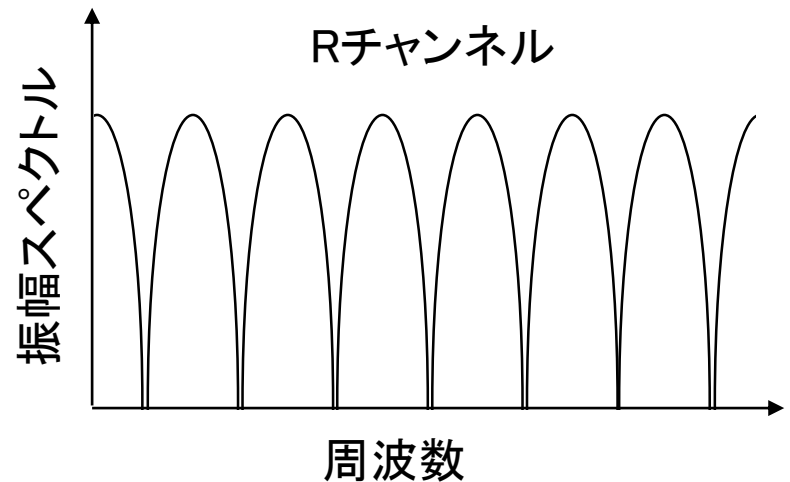
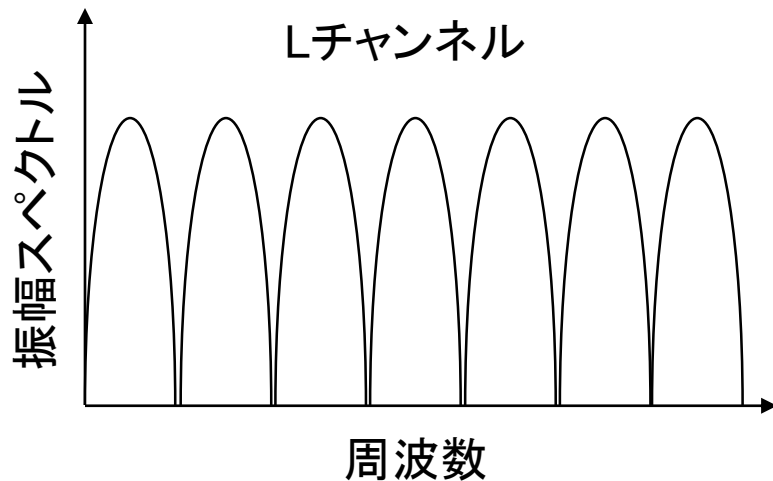
モノラルの音データに数ミリ秒程度のディレイをかけ、
本来の音データと加算または減算を実行

疑似ステレオ化の詳細

- 次式で定義

$$s_{1L}(n) = s_0(n) + s_0(n - d) \quad s_0(n) : \text{入力信号}$$
$$s_{1R}(n) = s_0(n) - s_0(n - d) \quad \begin{matrix} s_{1L}(n) \\ s_{1R}(n) \end{matrix} : \text{出力信号}$$

- 周波数特性は「くし型」: 山と谷が互い違い
帯域(=周波数の範囲)ごとに音像を左右に割り振る
⇒空間的に広がりのある音データを作り出す



疑似ステレオ化のデモンストレーション

- YouTubeで見つけた動画

<https://www.youtube.com/watch?v=cd7n4NPahe0>

- 演習のプログラムで作成

疑似ステレオ化 前	疑似ステレオ化 後
	

疑似ステレオ化の実装

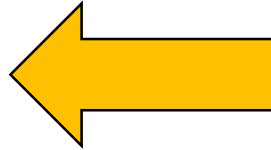
疑似コード

N : 音データの長さ, $s0$: 入力の音データ,
 $s1L$: 出力の音データ(Lチャンネル),
 $s1R$: 出力の音データ(Rチャンネル),
 $d \leftarrow$ デレイの時間

```
for  $n = 0$  to  $N - 1$  do           // 各時刻のデータに対して処理
     $m = n - d$ 
    if  $m \geq 0$  then
         $s1L(n) \leftarrow$  (スライドを参考に計算)
         $s1R(n) \leftarrow$  (スライドを参考に計算)
    end if
end for
```

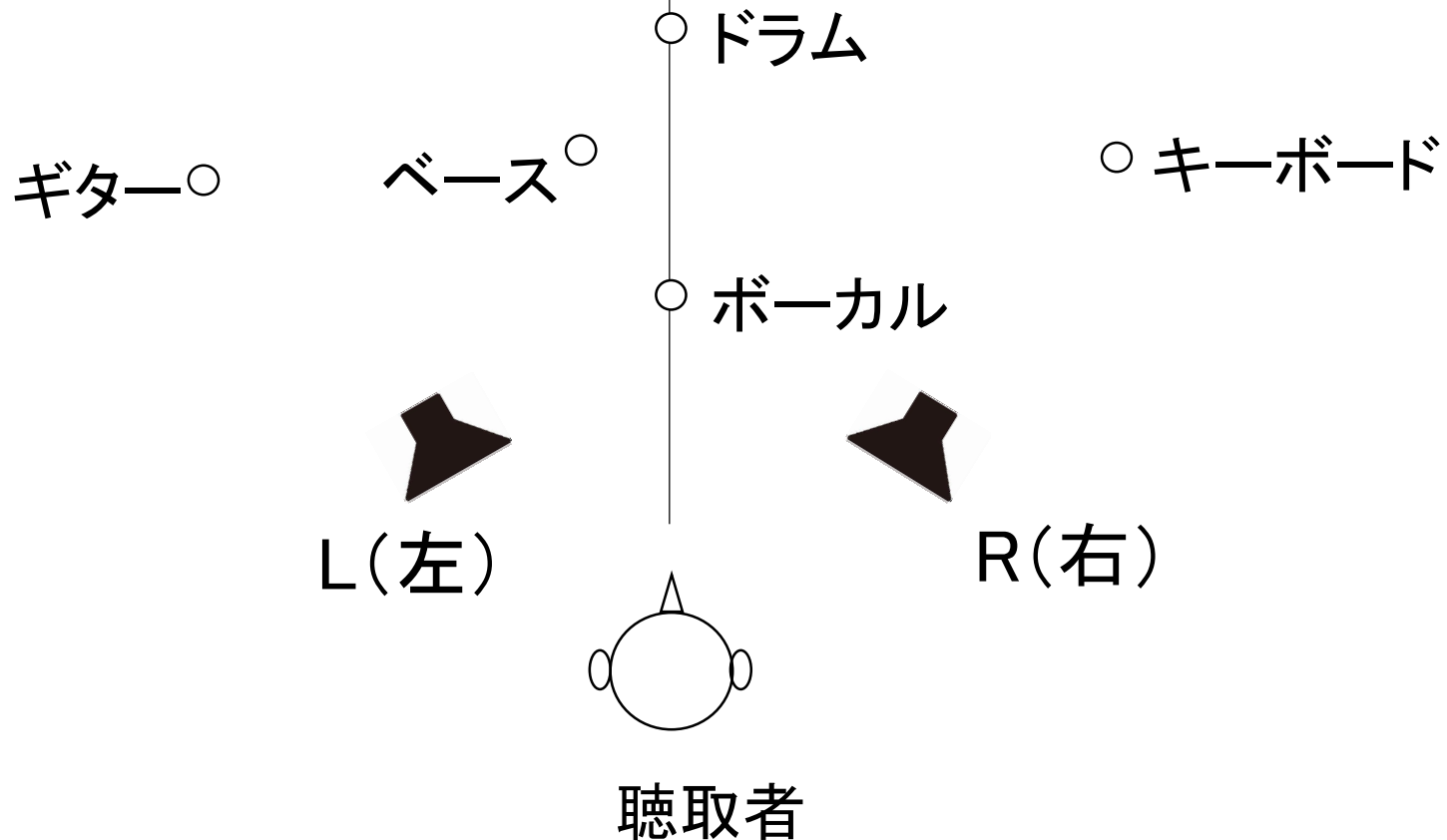
今日やること

- 続・音を広げる
 - オートパン
 - 疑似ステレオ化
- 音を削る
ボーカルキャンセラ
- 音の高さを上下させる
リサンプリング



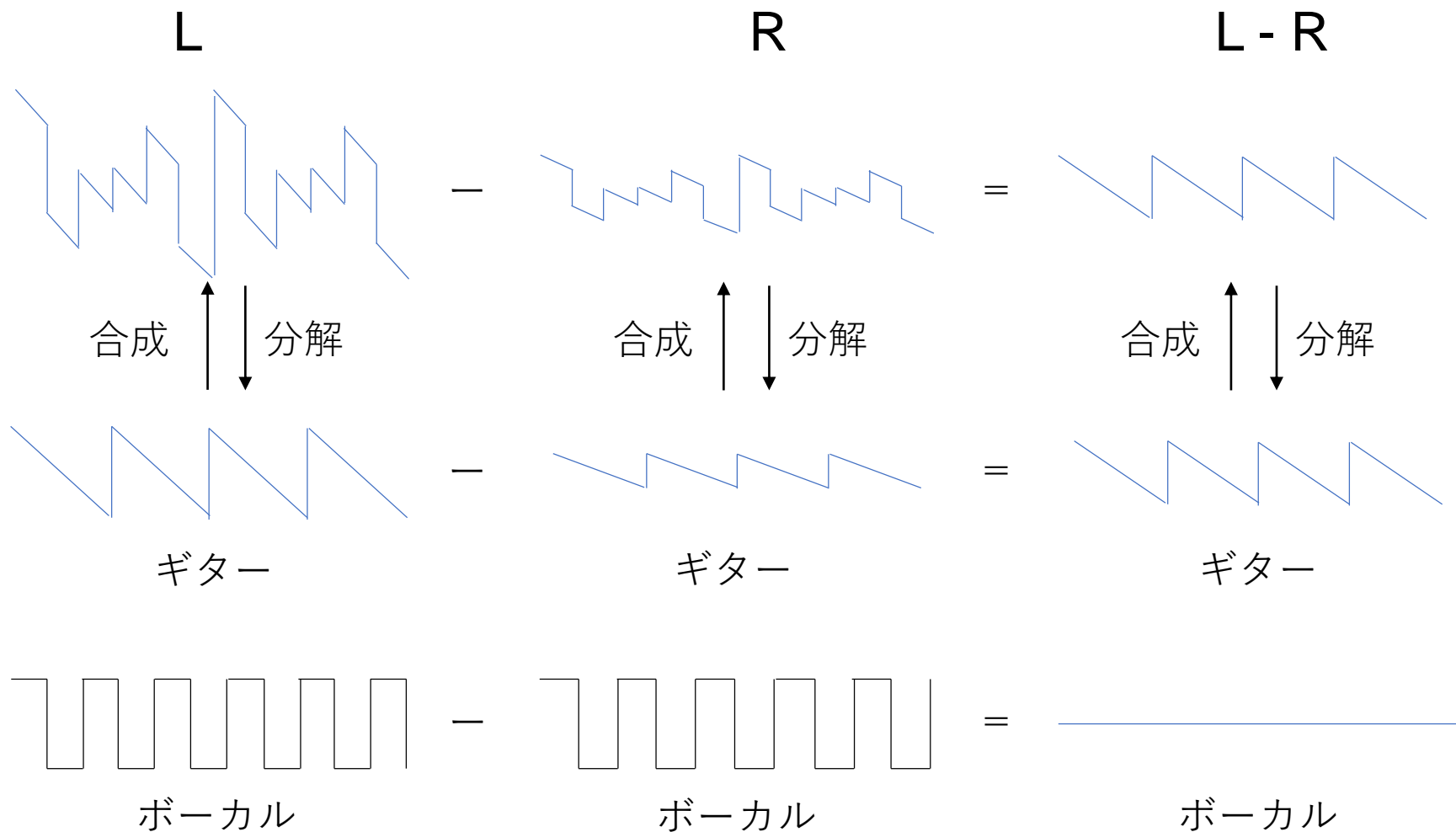
ステレオの音データ(再掲)

- ボーカル音像が中央になるように、歌声はLとRで音量は同じ大きさに録音
- ギターなどの楽器の音はどちらかのチャンネルを大きくすることで音像を左右に割り振る



ボーカルキャンセラ

- LとRのチャンネルの差分を取ることでボーカルの歌声を取り除くテクニック



ボーカルキャンセラ

- 次式で定義

$$s_1(n) = s_{0L}(n) - s_{0R}(n)$$

$s_{0L}(n)$: 入力信号(Lチャンネル)

$s_{0R}(n)$: 入力信号(Rチャンネル)

- メリット

めっちゃ簡単・お手軽な処理で実現

- デメリット

- 出力がモノラル
- 音像が中央に位置する楽器音も除去される

ボーカルキャンセラのデモ

楽曲番号	キャンセル前	キャンセル後
#1		
#2 ※		
#3 ※		

※ <http://music-note.jp/vocal/index.html>

ボーカルキャンセラの実装

擬似コード

N : 音データの長さ

s0L : 入力の音データ(Lチャンネル),

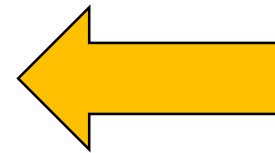
s0R : 入力の音データ(Rチャンネル),

s1 : 出力の音データ

```
for n = 0 to N -1 do      // 各時刻のデータに対して処理  
    s1(n) ← (スライドを参考に計算)  
end for
```

今日やること

- 続・音を広げる
 - オートパン
 - 疑似ステレオ化
- 音を削る
 - ボーカルキャンセラ
- 音の高さを上下させる
 - リサンプリング



標本化周波数の変更による音高変化

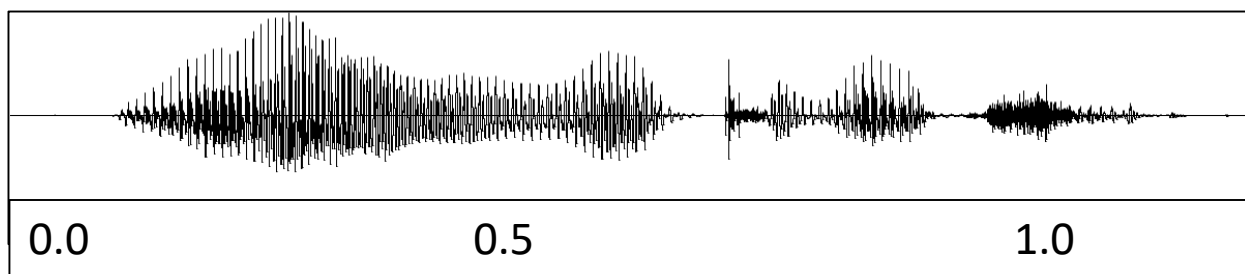
標本化周波数を変化させて音を再生

⇒ **再生速度の変化**により音の高さを変化

本来の音データ



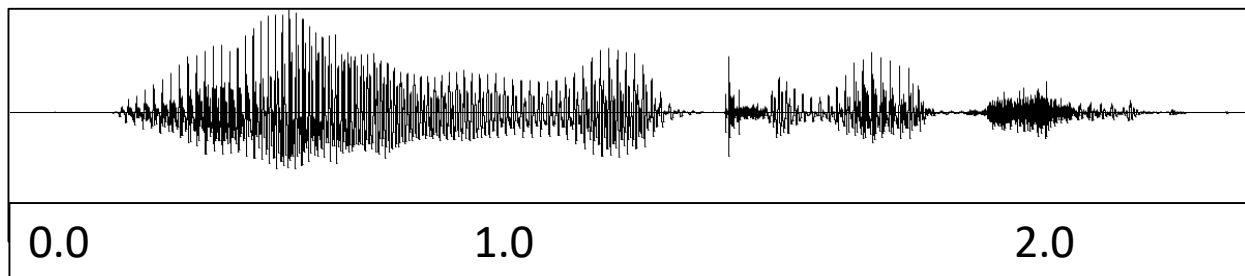
時間 (s)



標本化周波数 1/2 倍



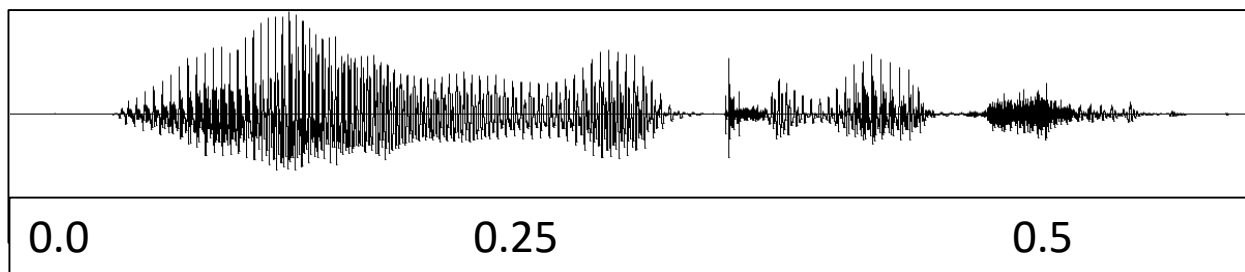
時間 (s)



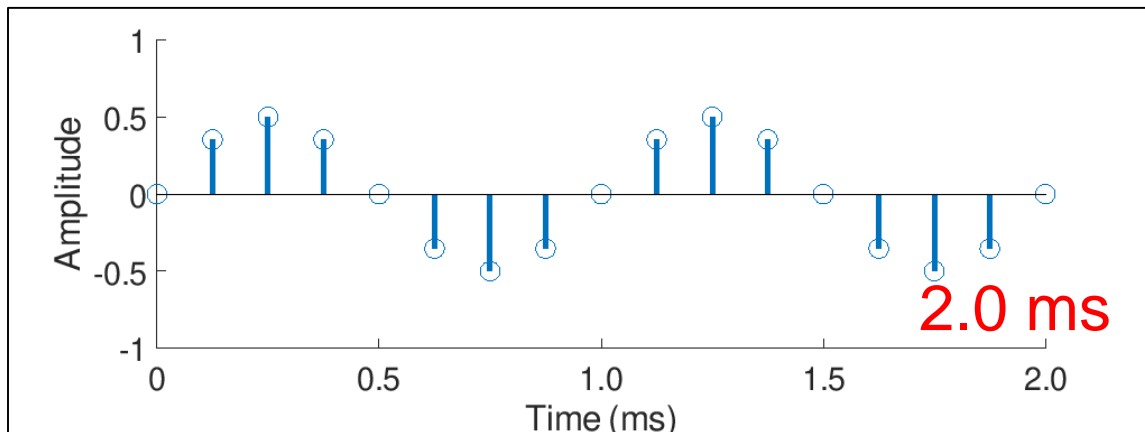
標本化周波数 2倍



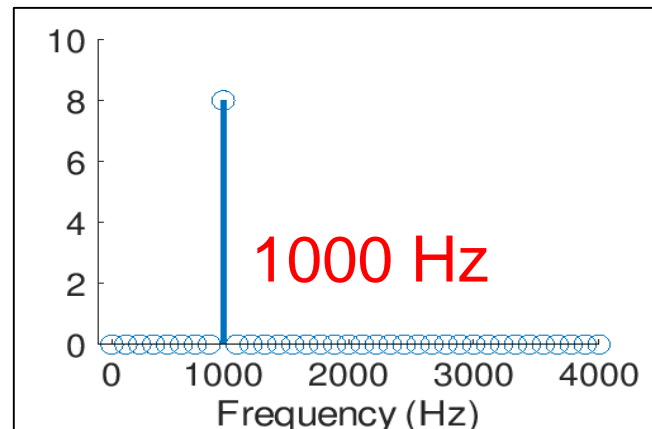
時間 (s)



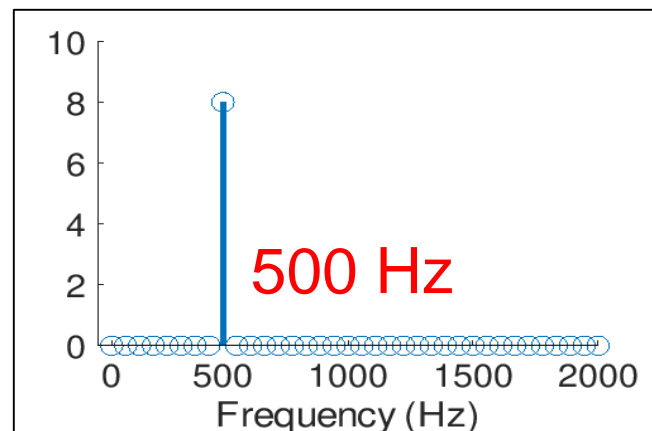
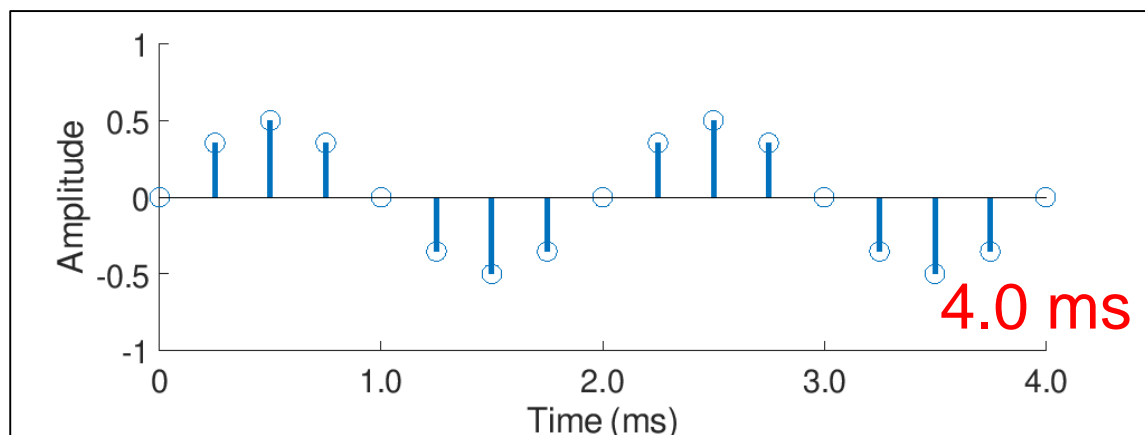
標本化周波数の変更(1/2)



波形

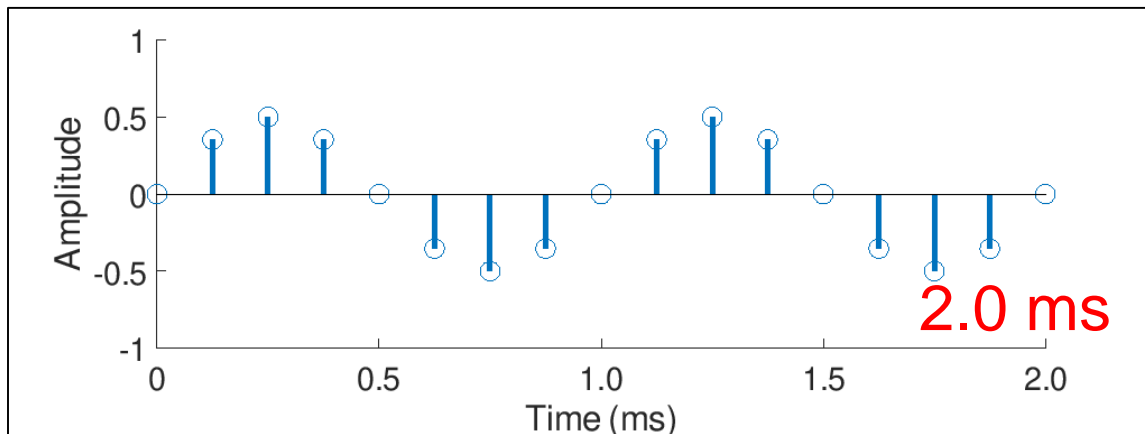


周波数特性

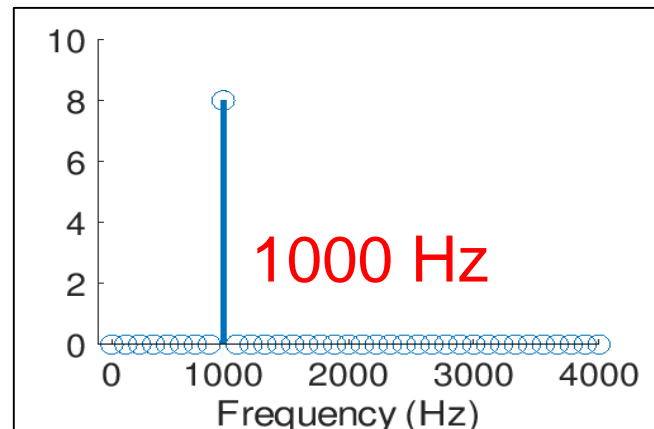


上段: 本来の音データ、下段: 標本化周波数 1/2倍

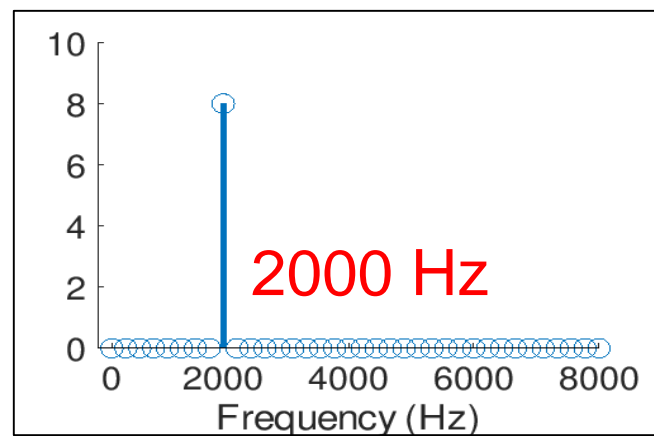
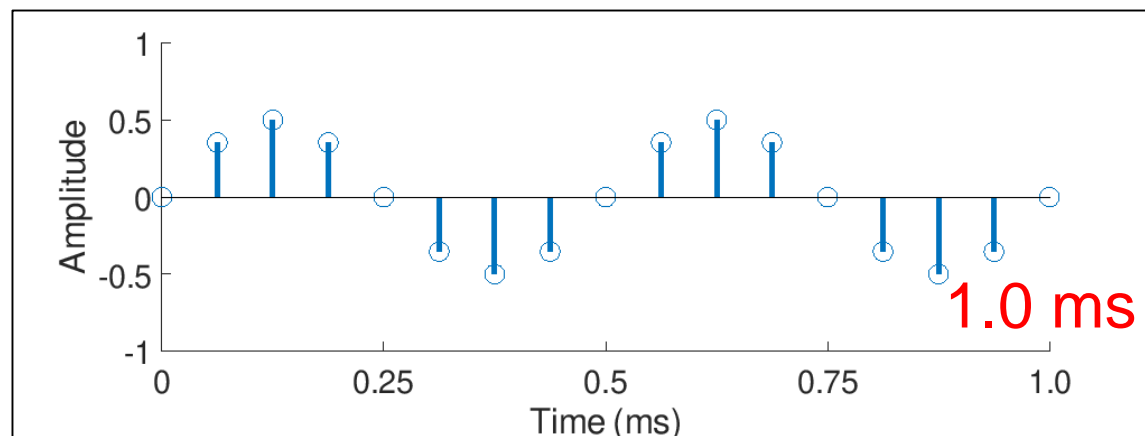
標本化周波数の変更(2/2)



波形



周波数特性



上段: 本来の音データ、下段: 標本化周波数 2倍

標本化周波数の変更(2/2)

- 波形も周波数特性も見かけ上は変化しない！
ただし：

- 時間軸の目盛りが変化
- 周波数軸の目盛りが変化
⇒再生速度・音の高さが変化

- 実は再生時の標本化周波数を変えずに
高さを変える方法もある
⇒リサンプリング(再標本化)による方法

Time (ms)

Frequency (Hz)

上段：本来の音データ、下段：標本化周波数 2倍

リサンプリングによる音高変化

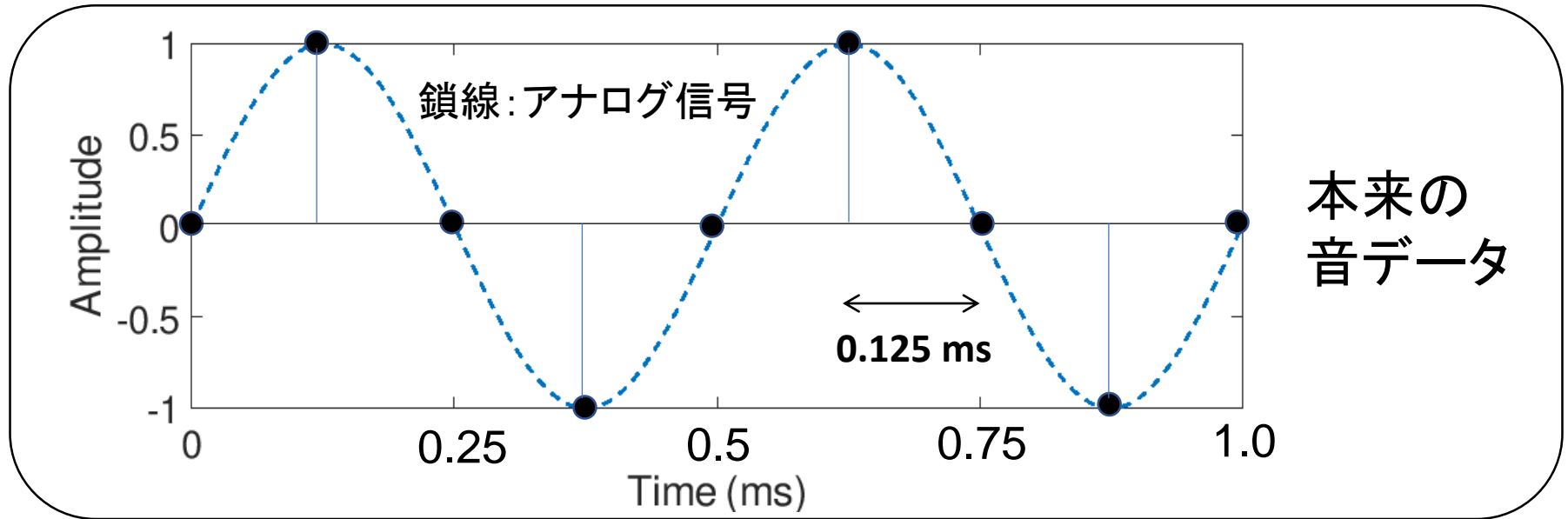
- 具体的な手順

1. デジタル信号(離散)をアナログ信号(連続)に戻す
2. 標本化周波数を変更して再び標本化
3. 元の標本化周波数で再生

⇒なるほど、わからん(次のスライドで説明)



リサンプリングにより音を低くする場合

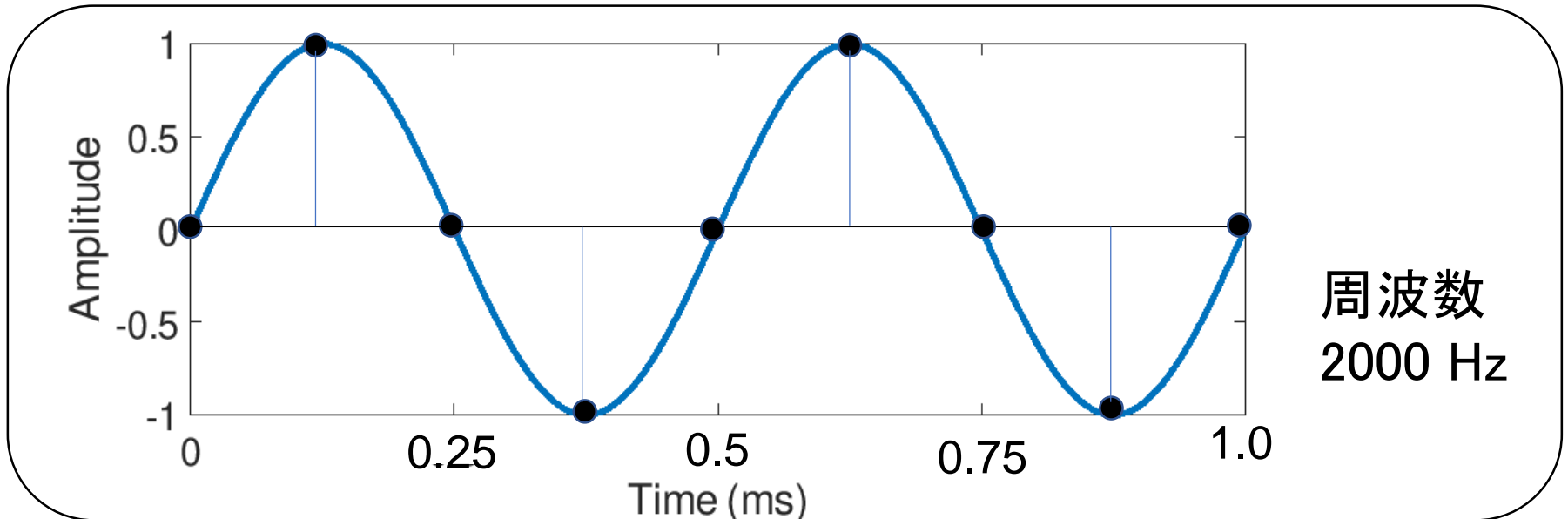


- 周波数: 2000 Hz (1msで2回振動⇒1000msで2000回振動)
- 標本化周期: 0.125 ms (= 1 / 8 ms)

周波数を半分にするにはどうするのか？

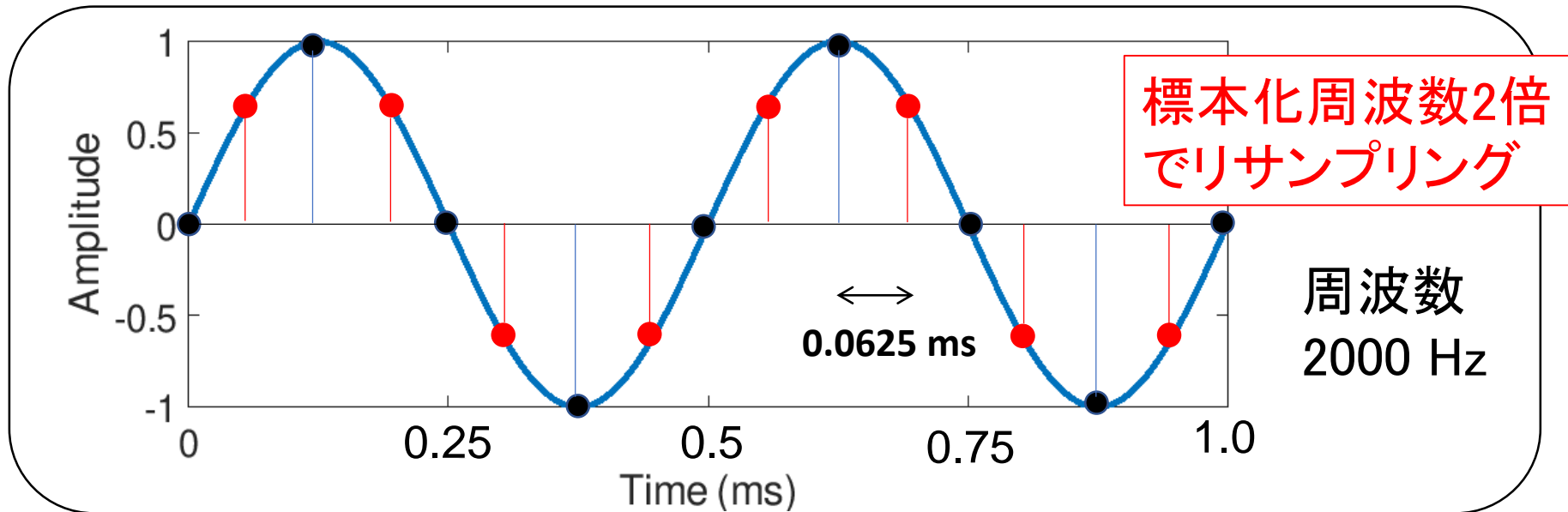
リサンプリングにより音を低くする場合

1. アナログ信号を復元



リサンプリングにより音を低くする場合

2. リサンプリング

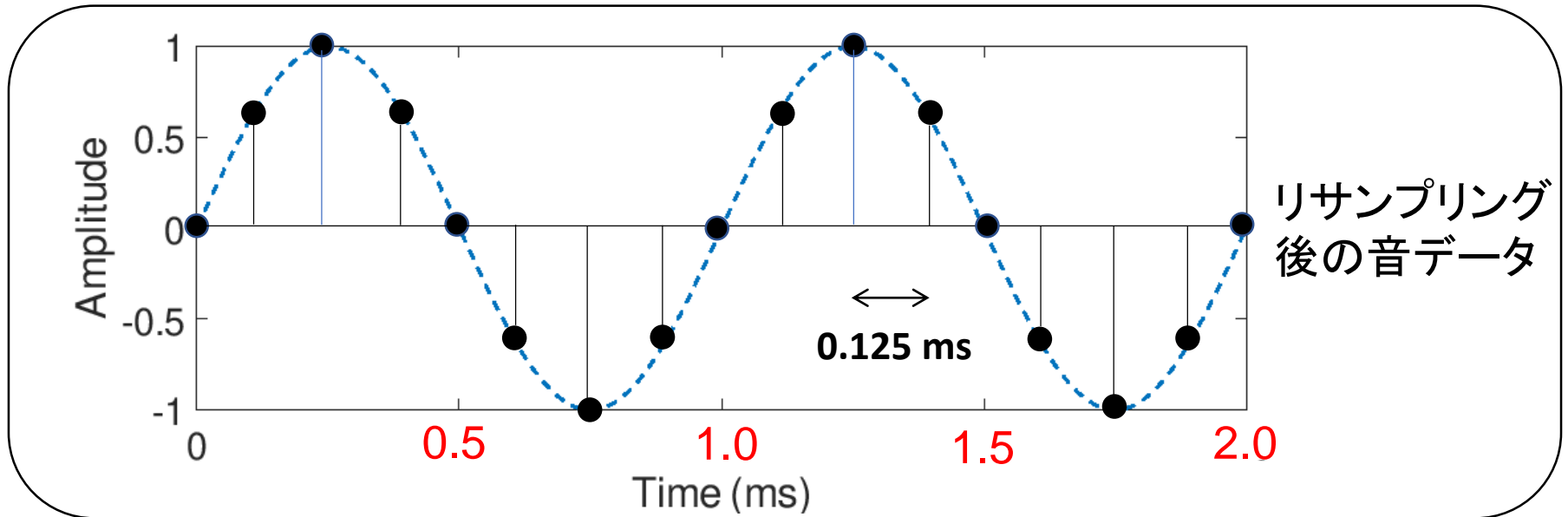


- 周波数: 2000 Hz
- 標本化周期: 0.0625 ms

音の高さはそのままデータの数が増えた！

リサンプリングにより音を低くする場合

3. 元の標本化周波数で再生



- 周波数: **1000 Hz**
- 標本化周期: 0.125 ms
- データの数が**増えた!**
- 標本化周期(間隔)は元通り
⇒ **再生時間が伸びる!**
⇒ **周期も長くなる!**

リサンプリングの詳細

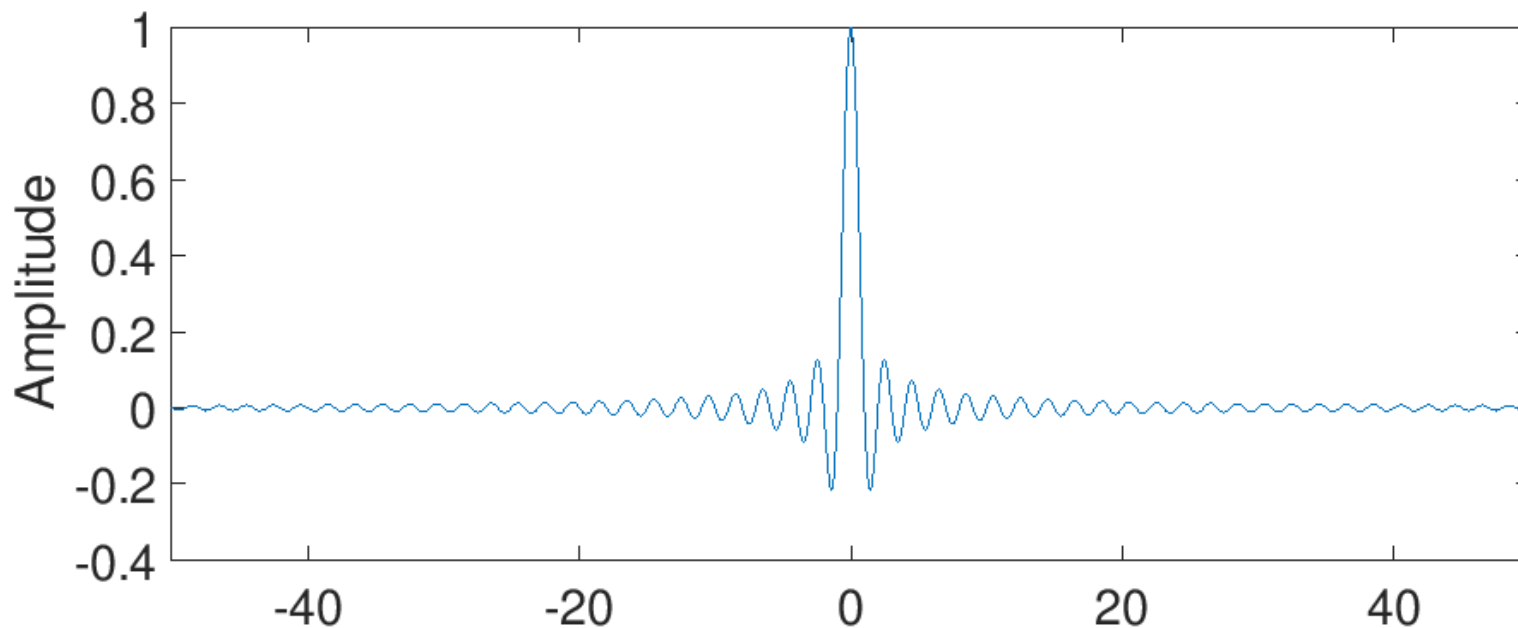
- サンプルとサンプルの中間値を求める必要
線形補間が最も簡単、ただし厳密には適切ではない
- 正しくリサンプリングを行うためには？
シャノンの標本化定理を考慮する必要
- シャノンの標本化定理
デジタル信号からアナログ信号を厳密に復元(誤差0)

$$s_a(t) = \sum_{n=-\infty}^{\infty} s_d(n) \text{ sinc }(\pi(t - n))$$

$s_a(t)$: アナログ信号 $s_d(n)$: デジタル信号

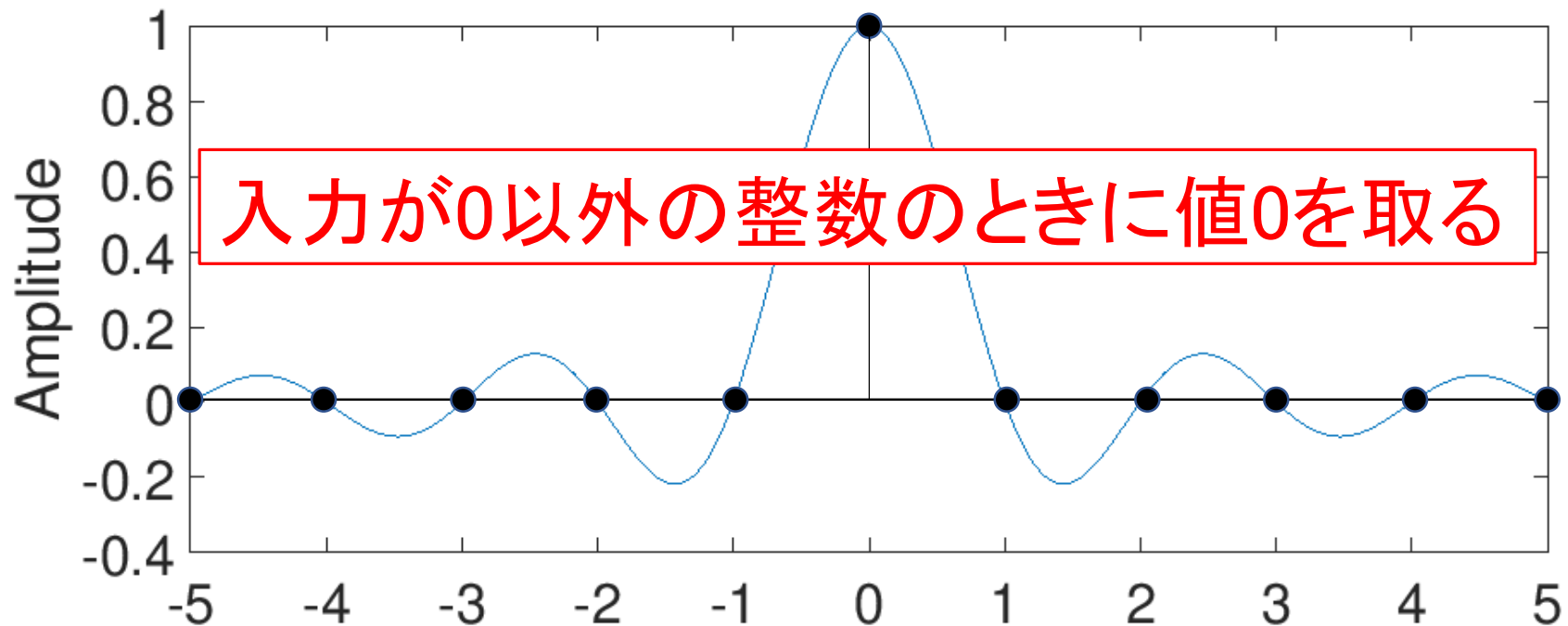
Sinc関数

$$\text{sinc}(\pi x) = \begin{cases} 1 & (x = 0) \\ \frac{\sin(\pi x)}{\pi x} & (\text{otherwise}) \end{cases}$$



Sinc関数

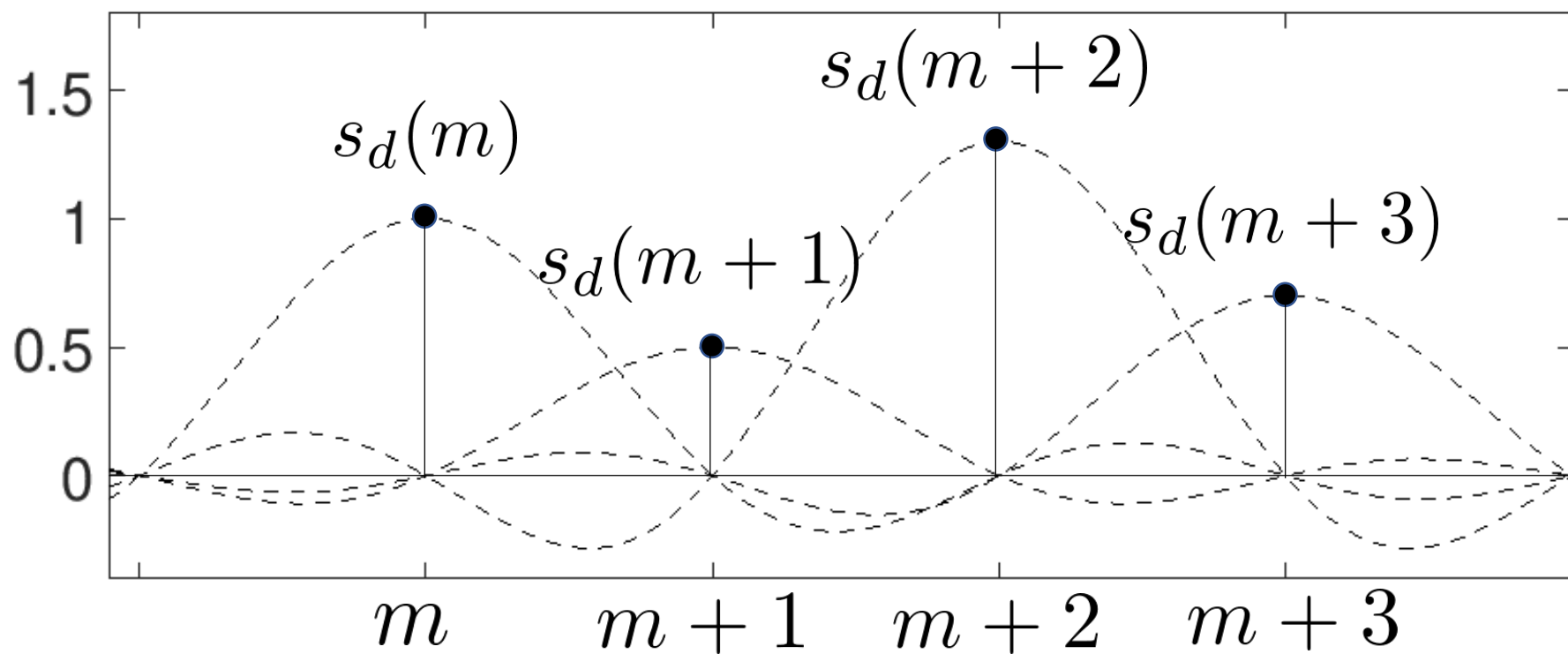
$$\text{sinc}(\pi x) = \begin{cases} 1 & (x = 0) \\ \frac{\sin(\pi x)}{\pi x} & (\text{otherwise}) \end{cases}$$



シャノンの標本化定理による復元

イメージ図

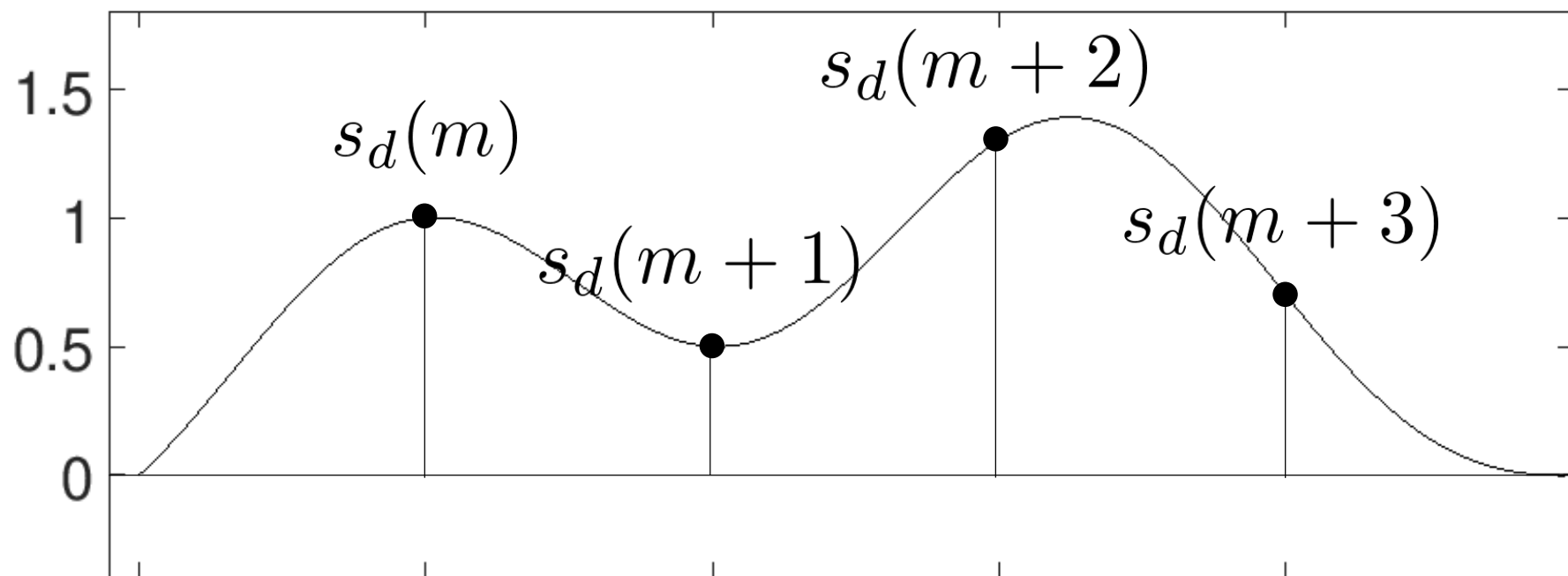
$$s_a(t) = \sum_{n=-\infty}^{\infty} s_d(n) \operatorname{sinc}(\pi(t - n))$$



シャノンの標本化定理による復元

イメージ図

$$s_a(t) = \sum_{n=-\infty}^{\infty} s_d(n) \operatorname{sinc}(\pi(t - n))$$



シャノンの標本化定理による復元

実は計算機で実現するのは不可能

なぜならば: **無限個の和**の計算を要するから

$$s_a(t) = \sum_{n=-\infty}^{\infty} s_d(n) \operatorname{sinc}(\pi(t - n))$$

実際は正の整数 J を取り $J + 1$ 個の和で近似

$$s_a(t) = \sum_{n=-J/2}^{J/2} s_d(n) \operatorname{sinc}(\pi(t - n))$$

リサンプリングの詳細(続き)

音の高さを変える場合は以下のように計算

$$\begin{aligned} s_1(n) &= s_a(\text{pitch} \times n) \\ &= \sum_{m=-J/2}^{J/2} s_0(m) \operatorname{sinc}(\pi(\text{pitch} \times n - m)) \end{aligned}$$

pitch : 音の高さを何倍にするかを表す倍率

ただし、和のインデックス m が負になる場合が発生
⇒インデックスが正の値を取る場合だけで計算すると
せっきくの近似が不十分になる問題($J/2$ 個の和)

リサンプリングの詳細(続き)

和の開始位置を調整することで和を取る際のインデックス m ができるだけ負にならないようにする

$$\begin{aligned} s_1(n) &= s_a(\text{pitch} \times n) \\ &= \sum_{m=\text{offset}-J/2}^{\text{offset}+J/2} s_0(m) \operatorname{sinc}(\pi(\text{pitch} \times n - m)) \end{aligned}$$




ただし $0 \leq m < N$

pitch : 音の高さを何倍にするかを表す倍率

offset : $\text{pitch} \times n$ の整数部分

もともと無限大の個数の和を近似するはずだったので、和の開始位置をずらしても影響はない

音高変化のデモンストレーション

リサンプリング 前	リサンプリング 後 (周波数 0.5倍)	リサンプリング 後 (周波数 1.5倍)
		

リサンプリングの実装

擬似コード

N : 音データの長さ, $s0$: 入力の音データ, $s1$: 出力の音データ

$\text{sinc}()$: sinc関数,

$\text{pitch} \leftarrow$ 音の高さの倍率, $J \leftarrow$ 和の項数

```
for  $n = 0$  to  $N - 1$  do           // 各時刻のデータに対して処理
     $t \leftarrow \text{pitch} * n$        // アナログ信号の時刻
     $\text{offset} \leftarrow (\text{int}) t$     //  $t$ の整数部分
    for  $m = \text{offset} - J / 2$  to  $\text{offset} + J / 2$  do
        if  $m \geq 0$  かつ  $m < N$  then
             $s1(n) += \dots$        // スライドを参考に
        end if
    end for
end for
```

まとめ

- 音を(空間的に)広げる
 - オートパン
 - ⇒ トレモロを応用し、音像を左右に動かす
 - 疑似ステレオ化
 - ⇒ ディレイを応用し、音像を左右に割り振る
- 音を削る
 - ボーカルキャンセラ
 - ⇒ 左右のチャンネルの音量差を利用しボーカルを消す
- 音の高さを上下させる
 - リサンプリング
 - ⇒ デジタル信号をアナログ信号に変換し再標本化

第6回
サウンドメディア論
および演習 演習編

準備

1. 20180518.zipをMoodleからダウンロードせよ
2. 適宜、適当なフォルダの下に解凍せよ
3. ターミナル上でmakeコマンドを実行せよ

演習 (1/3)

- オートパンのプログラムを作成せよ (auto_pan.c)
 - auto_pan_sample.wavはdepth = 1.0, rate = 0.2としてsample08.wavに対して実行した際の音データである
 - 特にrateを変えると効果が分かりやすい。好みの値に設定してみよ。
- 疑似ステレオ化のプログラムを作成せよ (pseudo_stereo.c)
 - pseudo_stereo_sample.wavはd = 0.01としてsample09.wavに対して実行した音データである
 - 疑似ステレオが破綻しないために、どこまでdの値を大きくできるか、を調べると良い。その値を設定せよ。

演習 (2/3)

- ボーカルキャンセラのプログラムを作成せよ
(vocal_cancel.c)
- vocal_cancel.wavはsample11.wavに対して実行したときの結果の音データである
- <http://music-note.jp/vocal/index.html> のサイトからボーカルつき楽曲(WAV)をダウンロードして効果を試してみるのも良いだろう

演習 (3/3)

- リサンプリングのプログラムを作成せよ (resampling.c)
 - まずはsinc関数の値を計算するC言語の関数を完成せよ:

$$\text{sinc}(x) = \begin{cases} 1 & (x = 0) \\ \frac{\sin(x)}{x} & (\text{otherwise}) \end{cases}$$

⇒ `double sinc(double x) { ... }` の中身を正しく書き換えることで作成する

- 作成したsinc関数を用いてプログラムを完成させる
- resampling_sample.wavはpitch = 0.5として sample12.wavに対して実行したときの音データである
- pitchの値を0.1や1.5など、大きくしたり小さくしたりした場合、音の高さとともに再生時間も変わることを確認せよ

提出するもの

- オートパン

- auto_pan.c (depth, rateの値は各自で設定)
- auto_pan.wav (対応する値を設定時の結果)

- 疑似ステレオ化

- pseudo_stereo.c (dの値は各自で設定)
- pseudo_stereo.wav (対応する値を設定時の結果)

- ボーカルキャンセラ

vocal_cancel.c

- リサンプリング

resampling.c

提出方法

1. フォルダを作成

フォルダ名は「学籍番号_0518」とする

例: 学籍番号がK123456ならば「K123456_0518」

2. 提出するファイルをその中に入れる

3. Finder上でフォルダをCtrl+左クリックし、圧縮ファイル(zip)を作成する

4. Moodleにアップロードして課題提出