ディジタル・オーディオの基礎

高岡　明

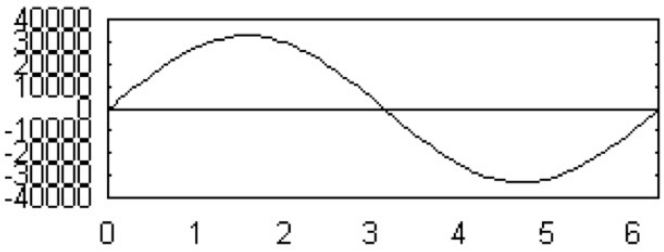
1. サウンド・エディターのダウンロードとインストール

以下のURLからAudacity 2.1.0をダウンロードしインストールする：
`http://web.audacityteam.org/`

2 音の属性

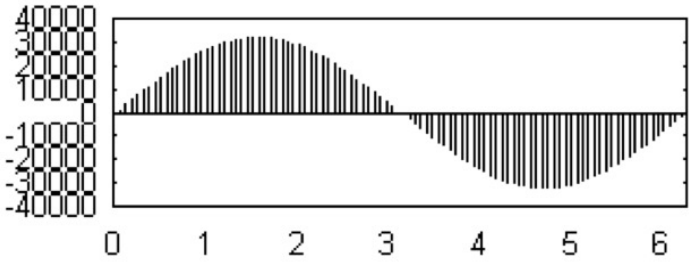
2.1 音のディジタイズ (サンプリング)

音は空気の気圧の変化であり、コンピュータに音を録音するためには、まず、マイクロフォンを使って、連続的な気圧の変化を電圧の変化に変換する。



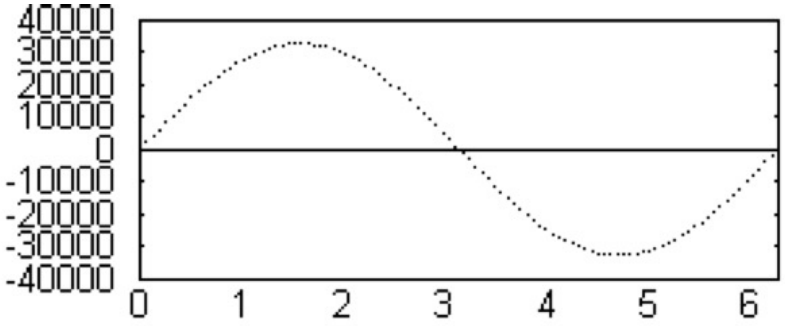
時間とともに変化する気圧(電圧)

次に、ADC によって、連続的に変化する 電圧の値を一定時間毎に測定し、あらかじめ決めた範囲の数値に変換する。この変換を、量子化 (quantization) という。



一定時間毎に測定される電圧（サンプル）

一定の時間はサンプリング周波数 (sampling rate) で表し、測定された値 をサンプル (sample) という。例えば、CD は 16 ビット (216 = 65536) で量子化され、サンプリン グ周波数は 44.1kHz なので、1/44100 秒毎にサンプルが取られ、サンプルは −32768 から +32767 の範囲の値を取る。サンプルは配列に格納される。



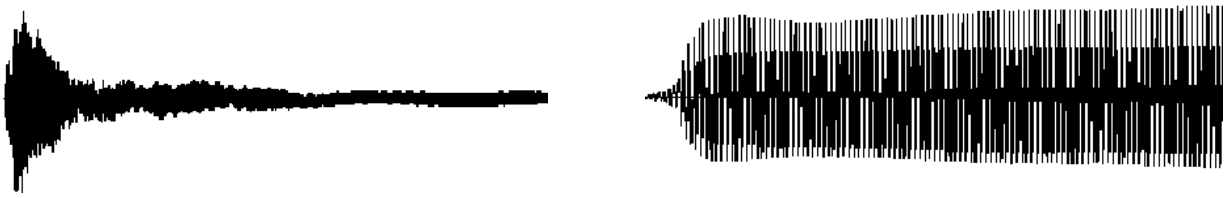
量子化

サンプルは表としてディスクに書き出される。この表をウェーブ・テーブル (波形テーブル wave table) という。サウンド・ファイル(オーディオ・ファイル)の実体は波形テーブルである。

サンプリングとは逆にコンピュータからサウンド・ファイルを再生する場合、コンピュータはサウンド・ファイルの波形テーブルに記録されたサンプルの値を DAC に送る。デジタル信号は DAC によってアナログの電気信号に変換される。DAC には、通常、アンプとスピーカーあるいはヘッドフォンなどが接続可能であり、DAC から送られた電気信号によってスピーカーやヘッドホンは振動体を振動させる。ADC および DAC は、コンピュータのサウンド・カード(オーディオ・インターフェイス)に搭載されている。

2.2 音量とエンベロープ

コンピュータでディジタイズしたピアノとクラリネットの音を、グラフの縦軸に音量 (amplitude)、横軸に時間をとって表示すると、図 1 のようになる。グラフから明かなように、ピアノの音は、クラリネットに比べて立ち上がりがより鋭い。音が出始めてから音量が最大点に達するまでの時間をアタック (attack time) という。また、クラリネットの音は、アタック以後音量が持続するのに対し、ピアノの音量は、指数関数的に減衰する。音量が一定に持続している時間をサステイン (sustain time)、減衰している時間をディケイ (decay time) という。こうした音量の時間的変化を、エンベロープ (amplitude envelope) という。図 2 は、ピアノの音のエンベロープ・カーブである。エンベロープは、音色を決定する要因の一つである。ピアノとクラリネットのエンベロープの違いは、音色の違いとして知覚される。



ピアノの音　　クラリネットの音 (冒頭部分)

図 1: ピアノとクラリネットの音

図 2: ピアノの音のエンベロープ · カーブ

また、人間の聴覚は、低い音よりも高い音の音量の変化により敏感である。Fletcher-Munson の図は、音量と周波数のこのような関係を表している。このように、コンピュータによって音を操作するためには、聴覚に関する音響心理学的な知識が必要不可欠である。音響心理学についての理 解は、音色など、音量以外の音の属性を理解するときにも欠かせない。

2.3 音の高さ

図 1 のクラリネットの音の中間部分を拡大すると、図 3 のようになる。

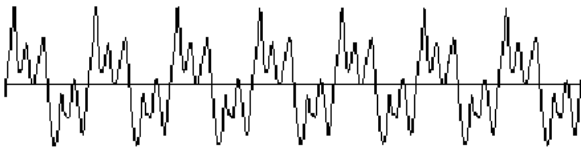


図 3: クラリネットの波形

この図から明らかなように、クラリネットの音は、気圧の変化のパターンに周期性がある。つまり、一定のパターンが 繰り返されている。音の気圧の変化のパターンを波形 (wave form) と呼び、波形に周期性があると き、その音は明確な音の高さを持つ (第 2.3 節参照)。音の高さは、周波数 (単位は Hz) によって 表す。周波数は、波形のパターンが 1 秒間に繰り返される回数である。音は、周波数が高い程、高 い音として知覚される。繰り返される一つのパターンの時間を周期という。従って、周期:*t*、周波 数:*f* とすると、周期と周波数の間には次の関係がある。

$$t = 1/f \qquad f = 1/t$$

従って、図 3 のクラリネットの音の高さは 123.5Hz なので、波形の 1 周期は 1/123.5 秒である。

2.4 フーリエ級数

フランスの数学者、フーリエ (Jean Baptiste Fourier 1768-1830) によると、周期性を持ったあ らゆる波形は、複数の正弦関数を加算して表すことができる。一つの正弦関数

$$y = \sin(x)$$

式 1

によっ て表される波形をサイン波 (sine wave) という。例えば、正弦関数 $y = \sin(x)\quad 0 \le x \le 3 \cdot 2\pi$ は次のグラフを描く:

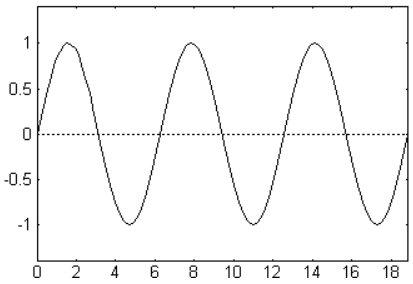


図 4: サイン波 (sine wave)

つまり、$y = \sin(x)$ の 1 周期は 2π に相当するので、$0 \le x \le 3 \cdot 2\pi$ の場合、波形が 3 周 期 (3 回) が描かれる。

音量 (amplitude) は、既に見たように波形の振幅の大きさで決まるので、音量の変化は、音量を A とすると、

$$y = A \sin(x)$$

式２

と表すことができる。図 1、2、3で見たように、音は、しばしば時間軸上に表示する。そこで、y = A sin(x) の周期は 2π なので、 式２ を上述した式 $t = 1/f$ および $f = 1/t$ によって次のように書き換える。

$$f(t) = A \sin(2\pi f t)$$

式３

ここで、角速度 (radian frequency) という単位を導入する。角速度: ω は、角度: θ、時間: t とすると、

$$\omega = \theta / t$$

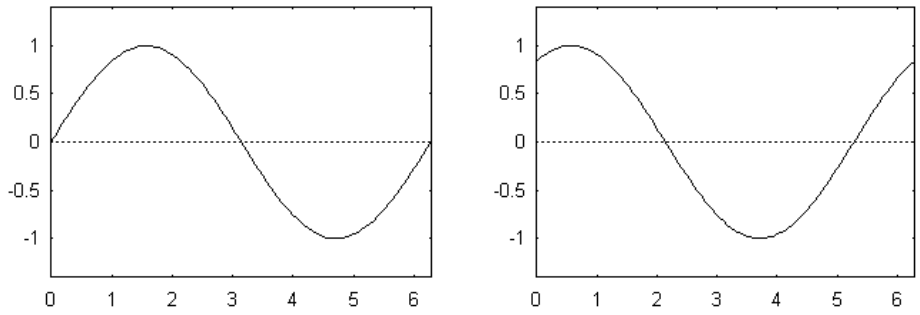
式４

と定義される。このことは、「速度=距離/時間」という関係から類推できる。さらに、波は常に 0 から始まるとは限らないので、定数 “φ” によって、波が始まる場所、位相 (phase) を決める。位相を加えて式３を式４によって書き換えると、次の式５になる。

$$f(t) = A \sin(\omega t + \varphi)$$

式５

この式によって、どのような位相のサイン波も表現可能である。例えば、$\varphi = 1$ のとき、$f(t) = \sin \omega t$ と $f(t) = \sin(\omega t + 1)$ は、それぞれ次のようになる：



$f(t) = \sin(\omega t)$ $f(t) = \sin(\omega t + 1)$

図 5: 位相がシフトしたサイン波

さらに、

$$A\sin(\omega t + \varphi) = a\cos(\omega t) + b\sin(\omega t)$$

式 6

という関係が成り立つので、サイン波を合成してできる波形は、どのような波形でも (8) 式で表すことができる。

$$f(t) = \sum(a_n \cos(n\omega t) + b_n \cos(n\omega t))$$

式 7

式 7 をフーリエ級数式という。

2.5 矩形波の合成

例えば、図 6 の矩形波 (square wave) と呼ばれる波形は、

$$f(t) = \sum(1/2n-1) \sin((2n-1)\omega t)$$

式 8

というサイン波の加算によって合成することができる。

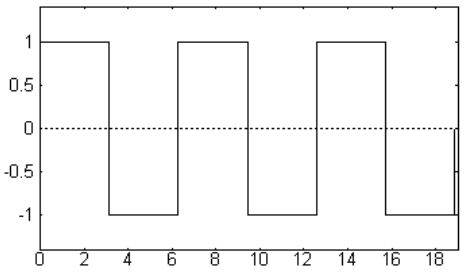
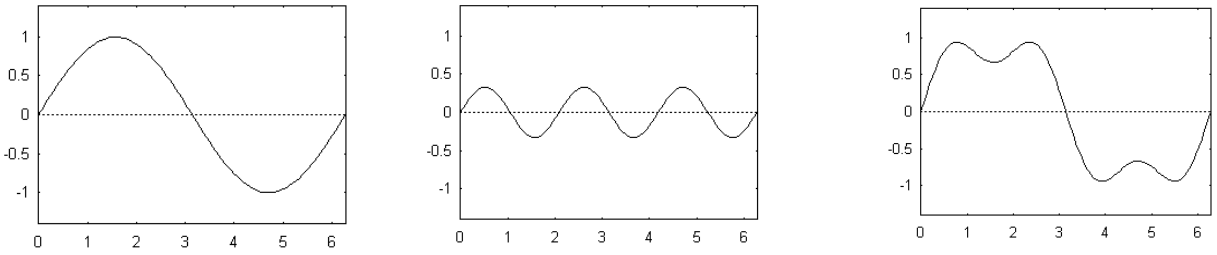


図 6: 矩形波

ある波形を構成する個々のサイン波を倍音（harmonic または harmonic partial）と呼び、特に $n = 1$ の倍音を基音 (fundamental) という。つまり、矩形波は、奇数倍 $(2n - 1)$ の倍音のみを持ち、その音量が $1/(2n - 1)$ であるサイン波を加算することによって合成される。矩形波は、木管楽器などの音色を作るときに用いられる。サイン波の加算によるこうした 音響合成の方法を、加算合成 (additive synthesis) またはフーリエ合成 (Fourier synthesis) という。

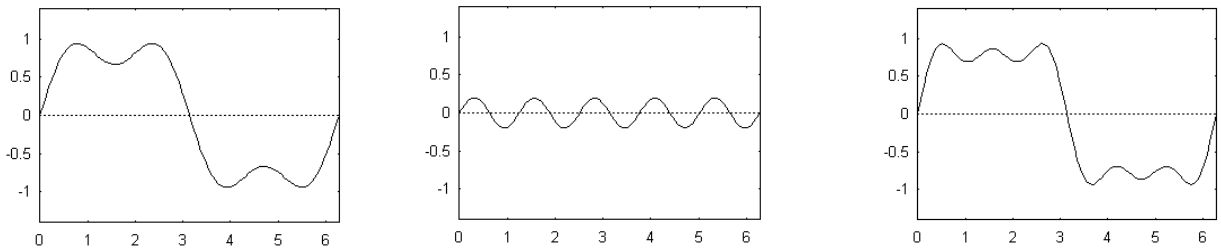
次に、式８に従ってサイン波を次々と加算し、波形がどのように変化するか実際に見てみる。まず、音量が 1 のサイン波 $f(t) = \sin(\omega t)$ と、音量が 1/3 で 3 倍の周波数を持つサイン波 $f(t) = (1/3)\sin(3\omega t)$ の二つのサイン波を加算する。



$\sin(\omega t)$ + $(1/3)\sin(3\omega t)$ = $\sin(\omega t) + (1/3)\sin(3\omega t)$

図 7: 矩形波の合成

5 倍の周波数を持ち音量が 1/5 であるサイン波 $f(t) =(1/5)\sin(5\omega t)$ を加算すると、次のような波形になる。



$\sin(\omega t) + (1/3)\sin(3\omega t)$ + $(1/5)\sin(5\omega t)$ = $\sin(\omega t) + (1/3)\sin(3\omega t) + (1/5)\sin(5\omega t)$

図 8: 矩形波の合成

さらに、以下のように第 17 倍音まで 3 周期加算すると図 9 になる。

$f(t) = \sin(\omega t) + (1/3)\sin(3\omega t) + (1/5)\sin(5\omega t) + \ldots + (1/17)\sin(17\omega t)$

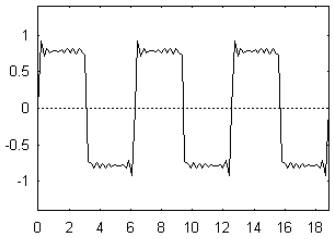
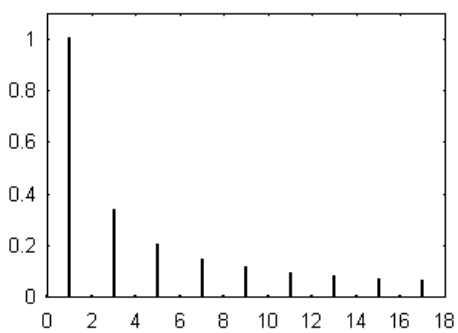
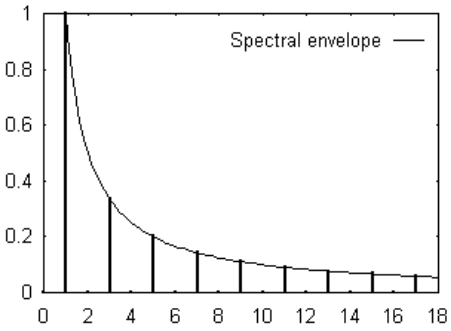


図 9: 矩形波の合成

図 9 の波形は、縦軸に音量、横軸に倍音 (周波数) を取ると、図 10 の様に倍音構成 (spectra) を表示することができる。また、各倍音を線で結んだ曲線をスペクトラル · エンベロープ (spectral envelope) という。倍音構成は、音色を決定する要因の一つである。



矩形波の倍音成分 (spectra)



矩形波の spectral envelope

図 10: 矩形波の倍音構成とスペクトラル · エンベロープ

3 加算合成のコンピュータ・プログラム

3.1 サンプリング、量子化、ウェーブ · テーブル

まず最初に、サイン波をコンピュータで合成する。ある波形の音をコンピュータから再生するには、波形の 1 周期分のウェーブ · テーブルを作成し、メモリにロードしたテーブルを再生したい時間だけ繰り返し DAC(Digital-to-Analog Converter) に送る。
サイン波 1 周期 のウェーブ · テーブルは、次のプログラム 1 によって生成される。

```
for(int i = 0; i < waveTableSize; i++)
   waveTable[i]  = (byte)(quantization*amplitude*Math.sin(1*i*(2*Math.PI)/waveTableSize));
```

プログラム 1

次に、矩形波を合成するプログラムを示す (プログラム 2)。

```
    for(int i = 0; i < waveTableSize; i++)
       waveTable[i] = (byte)(quantization * amplitude * (Math.sin(1 * i * (2 * Math.PI) / waveTableSize)
                                                        + Math.sin(3 * i * (2 * Math.PI) / waveTableSize) / 3
                                                        + Math.sin(5 * i * (2 * Math.PI) / waveTableSize) / 5
                                                        + Math.sin(7 * i * (2 * Math.PI) / waveTableSize) / 7));
```

プログラム 2

3.2 サンプリング · インクリメント

ウェーブ · テーブルを用いて音響合成を行う際、周波数を変えるには、サンプリング · インクリメント (sampling increment) を用いる。

```
    // frequency = samplingIncrement * samplingRate / waveTableSize
    int samplingIncrement = (int) (frequency * waveTableSize / samplingRate);

for(int ii = 0; ii < samplingIncrement * numberOfSamples / waveTableSize; ii++)
  for(int i = 0; i < waveTableSize / samplingIncrement; i++)
    samples[(byte)(waveTableSize / samplingIncrement) * ii + i] = waveTable[i * samplingIncrement];
```

プログラム 3

3.3 ノイズの合成

図 11 が示すように、ノイズ (white noise) の波形は周期性を持たず、それ故、音の高さは明確でない。

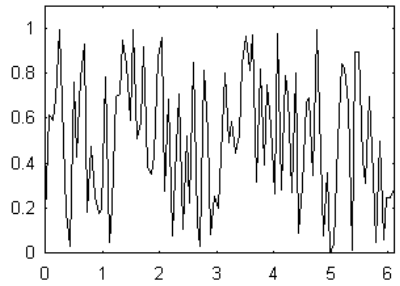


図 11: ホワイト・ノイズ (ランダム・サンプル) の波形

ノイズは次のように乱数によって生成する。

```
    for(int i = 0; i < numberOfSamples; i++)
      samples[i] = (byte) (amplitude * quantization * (Math.random() - 0.5));
```

プログラム 4

図 12 が示すように、ピアノの音は、鳴った瞬間にはノイズを含んでいる。

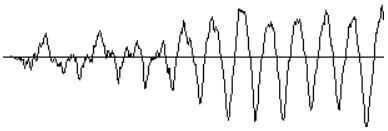


図 12: ピアノの音の冒頭部分

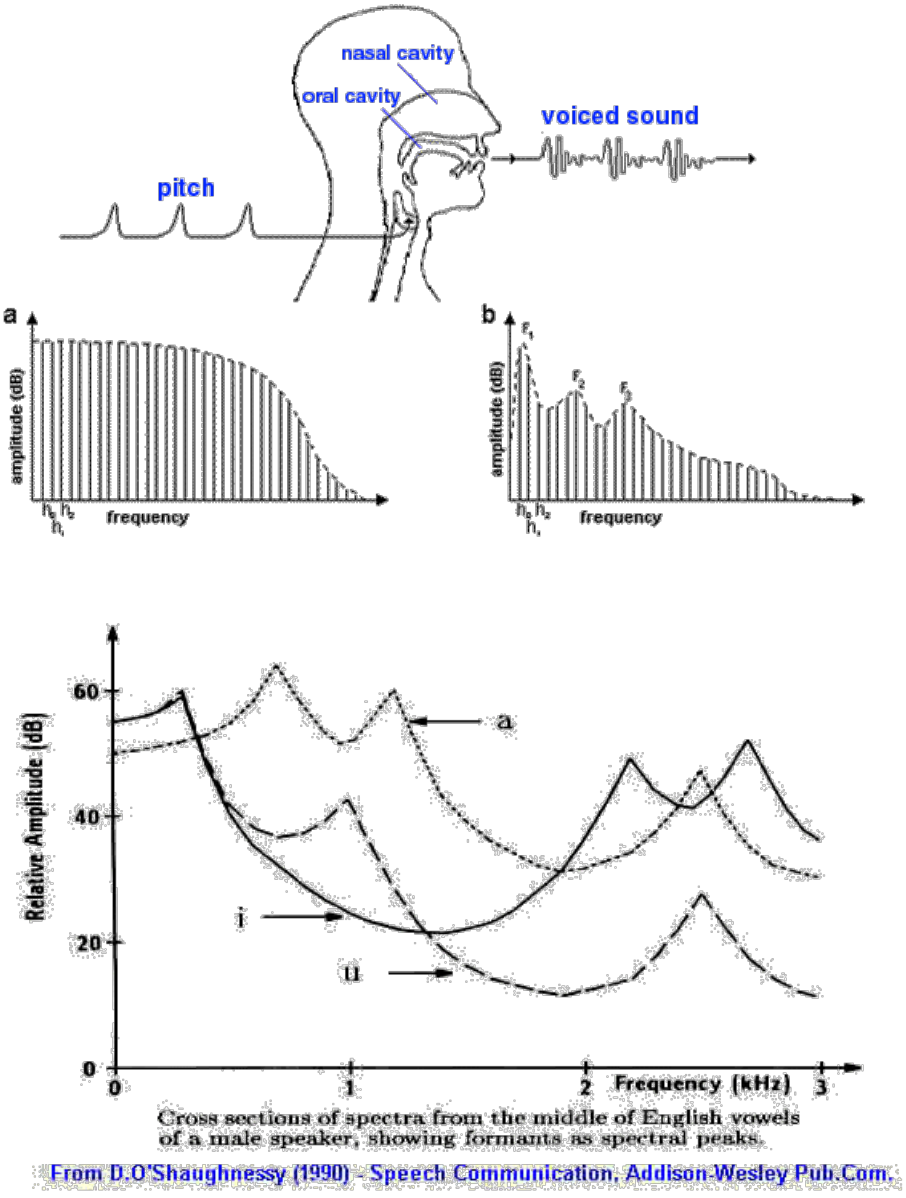
なぜならば、弦が ハンマーで打たれた瞬間は、弦が不規則に振動するからである。対照的に、次の図13は、図12と同じ音の中間部分であり、明確に周期性が認められる。

図 14: ピアノの音の中間部分

このように、現実の音は、どのような音 でもノイズ成分をある程度含んでいる。例えば、話し声も子音はノイズ成分である。

4 スピーチ合成

4.1 フォルマント formant



Cross sections of spectra from the middle of English vowels of a male speaker, showing formants as spectral peaks.

From D.O'Shaughnessy (1990) - Speech Communication, Addison-Wesley Pub.Com.

Table 3. Mean harmonic frequency values for the first three formants (F1, F2, F3) in Hz, for each vowel and gender group.

VOWELS	FEMALES			MALES		
	F1	F2	F3	F1	F2	F3
/a/	1002,90	1549,95	2959,70	753,87	1278,70	2483,44
/ɛ/	672,45	2242,93	3018,60	588,44	1745,11	2566,00
/e/	437,03	2429,76	3087,09	406,63	1955,60	2540,33
/i/	361,90	2583,89	3378,14	297,80	2150,85	2925,14
/ɔ/	715,34	1073,27	2981,69	580,15	947,25	2525,52
/o/	444,89	914,26	2899,80	411,62	832,84	2376,13
/u/	461,82	763,41	2902,55	345,27	799,51	2351,50

5 より複雑な音響合成 · 処理

コンピュータによる音響合成の研究は、Max Mathews によって、1950 年代後半にアメリカ、ニュージャージー州にある AT&T のベル研究所で始められた。彼が書いた Music4 というプログ ラムは、RTcmix や Csound など、現在使用されているあらゆる音響合成 · 処理ソフトウェアの規 範となった。

音響合成の方法には、ここで扱った加算合成の他にも多くの様々な方法がある。代表的なものを挙げると、AM(Amplitude Modulation)、Wave Shaping、スタンフォード大学の作曲家、John Chowning が開発した FM(Frequency Modulation)、スタンフォード大学の Julius Smith やプリン スト ン大学の Perry Cook などによる物理モデル (physical modeling)、様々なフィルターを使った 減算合成 (subtractive synthesis)、そして更に、現実音の分析結果を使って合成を行う LPC(Linear Predictive Coding) やフェイズ · ボコーダー (phase vocoder) などがある。また、音声の合成には、 フォーマント · トラッキング (formant tracking)、物理モデル、LPC などが用いられる。市販のシ ンセサイザーが使用しているのは、こうした数ある方法の内のほんの一部に過ぎない。

ここで説明したように、汎用言語（JavaやC++）によって加算合成以外の音響合成プログラムやフィルターなどの音響処理プログラムを書くことも可能であるが、既に、RTcmix、Csound、SuperColliderなど の音響合成および音響処理のためのソフトウェア · パッケージが利用可能である。これらのパッケージは、音響合成 · 処理に必要な 関数やクラスのライブラリである。こうしたライブラリを用いると、あらゆる音響合成や音響処理 のプログラムを比較的容易に作成することが可能であり、市販の製品よりも遥かに高性能なシンセ サイザーやイフェクターのプログラムを作成することができる。

参考資料 ：音響合成、音響処理のソフトウェア ・パッケージ

Csound: http://www.csounds.com/
RTcmix: http://www.music.columbia.edu/cmc/RTcmix/
SuperCollider: http://www.audiosynth.com/
PD (Pure Data): https://puredata.info/