音律研究のためのWAVファイルの作成法

――数値計算による人工音について――

How to make WAV-files for the studies of tuning and intonation

――Artificial sound by numeric operations――

菅　　千　索
Sensaku SUGA
（心理学教室）

2013年８月19日受理

１．はじめに

西洋音楽においては、最古の音律とされるピタゴラス律と、その修正形とみなすことができる純正律、中全音律、ウェル・テンパード律、平均律などの音律が存在している（菅、2011、2012a、2012b参照）。それらは音名が同じであっても周波数（音高）が微妙に異なっており、たとえば音程でいえば、完全５度は純正律よりも平均律の方が1.955セント（平均律半音の１/50以下）狭い。

こうした僅かな違いが、同じ音楽の演奏に対する旋律性や協和性の知覚・認知および嗜好・選好性に、どのような影響を及ぼすのかについては、大変興味深いところである。こうした問題を芸術や美学の観点から主観的に論じるのではなく、音楽心理学という立場から客観的かつ定量的に検討するためには、それぞれの音律に準拠して正しく演奏される音響刺激が作成されねばならない。

このような条件を満たす刺激を作成する方法は、おおむね以下のように分類することができる。

１．自然楽器の使用
　（１）鍵盤楽器（ex．ピアノ）の調律替え
　（２）弦楽器（ex．バイオリン）の弾き分け
２．MIDI対応電子楽器・音源の使用
　（１）システム・エクスクルーシブを利用
　（２）ピッチベンドを利用
３．コンピュータの使用
　（１）Audio I/F（音源ボード）DACを直接制御
　（２）数値計算によるWAVファイルの作成

１の方法はあまり現実的ではないと判断されるが、自然楽器に近い音が必要ならば２の方法を選ばざるを得ない。一方、サイン波など比較的簡単な数値計算で求められる人工音であれば３の方法が有力であり、とりわけOSがサポートする機能をアプリケーション側から利用できる（２）が望ましいと予想される。本発表においては、そのような判断に基づいて開発された数値計算による音響刺激の作成法について報告する。

２．開発ツール

OSとしてWindows７を搭載したPC上において、言語はStrawberry Perl（５.14．２．１）、統合開発環境としてPedra，The Perl IDE（0.94）、PerlからWin32 APIを簡単に利用することを可能にしてくれるCPANモジュールのWin32::Soundを利用した。なお、Pedraのインストールの際に同時に導入されるStrawberry Perlには、同モジュールが含まれていないため、Pedra附属のGUIアプリケーションPedra CPAN clientでダウンロードする必要がある。また、作成されたWAVファイル内のデータ検証にはAudacity，Rなどを使用している。

３．作成方法

ここでは開発されたPerlコードの簡略版に基づいて、WAVファイル作成のためのプログラミングについて解説する。ただし、この簡略版では、全体としての整合性を欠いている部分があり、またエラー処理などはすべて省略されている。

3.1 初期化（List１）**1**：でuse文によりモジュール使用の宣言を行い、**2**：でWin32::Sound::WaveOutのインスタンス$WAVEを作成する。その際、コンストラクタへの引数１はサンプリング・レート$SR（11025｜22050｜44100｜48000｜96000）、引数２は量子化ビット数$QB（８｜16）、引数３はチャンネル数$CH（１｜２）である。**3**：は必須ではないが念のため。

3.2 波高計算（List２）基音だけしかもたない純音であるサイン波（正弦波）は、Perlの組込数学関数であるsinを使って求めればよい。それに対して基音と倍音からなる複合音の場合は、数値計算によって波形そのものを直接求める方法と、逆フーリエ変換によって基音と倍音（いずれも純音）を合成することにより間接的に求める方法が考えられる。前者の場合は、定義通りの理想波が得られるが、そこには基音から無限大倍音まで含まれることになる（換言すれば、無限大倍音まで合成しなければ、波形のエッジが直角や鋭角にはならない）。この波形をDA変換によってアナログ化すると、

List 1 初期化

```
1: use Win32::Sound;                    # モジュール使用宣言
2: $WAVE=new Win32::Sound::WaveOut($SR,$QB,$CH);
                                        # インスタンスの作成
3: $WAVE->Volume('100%','100%');
```

List 2 波高計算

```
1: $t=2.0*$PI*$fq*($i/$SR);
2: return sin ($t);                     # サイン波
3: $v=0.0;
4: for (my $i=1; $i<=$MH; $i+=2)  {
5:          $v+= (1/$i) *sin ($i*$t);   # 矩形波
6: }
7: for (my $i; $i<=$MH; $i++)  {
8:          $v+= (1/$i) *sin ($i*$t);   # 鋸歯状波
9: }
7: for (my $i; $i<=$MH; $i++)  {
8:          $v+=sin ($i*$t);            #「等倍音」波
9: }
A: return ($v);
```

List 3 オシレータ

```
1: $tp=int ($SR*$dr+0.5);
2: for (my $i=0; $i<$tp; $i++)  {
3:          $da [$i] =&get_value ($wf, $fq,$i);
4: }                                    # 波高計算
5: return @da;
```

List 4 エンベロープ・ジェネレータ

```
1: $na=int($SR*$at+0.5);                # アタック　点数
2: $nd=int($SR*$dt+0.5);                # ディケイ　点数
3: $nr=int($SR*$rt+0.5);                # リリース　点数
4: $ns=$tp-($na+$nd+$nr);               # サステイン点数
5: $p=0;
6: for(my $i=0;$i<$na;$i++) {  # Attack   ($na > 1)
7:          $da[$p++]*=(1/($na-1))*$i;
8: }
9: for(my $i=0;$i<$nd;$i++) {  # Decay    ($nd > 1)
A:          $da[$p++]*=1-((1-$sl)/($nd-1))*$i; }
B: }
C: for(my $i=0;$i<$ns;$i++) {  # Sustain ($ns > 1)
D:          $da[$p++]*=$sl;
E: }
F: for(my $i=0;$i<$nr;$i++) {  # Release ($nr > 1)
G:          $da[$p++]*=$sl-($sl/($nr-1))*$i;
H: }
I: return @da;
```

サンプリング定理により、サンプリング周波数の１/２であるナイキスト周波数を超える領域の倍音が折り返されて（エイリアシング）、ナイキスト周波数よりも下（通常は可聴域）に現れるという不具合が避けられない。サンプリング周波数を高くすることで（たとえば192 kHzなど）、ある程度は抑制できるが、根本的な解決方法ではない。

一方、後者で逆フーリエ変換を行う際に、合成する倍音を有限で打ち切れば近似波しか得られないことになるが、その結果として不要な周波数成分が可聴域に混入する事は確実に避けることができる。経験的にではあるが、広く一般に普及している自然楽器音の場合、20倍音あたりまでが含まれておれば基本的なニュアンスは再現できると考えられるし、ここで扱うのは人工音であるから、十分に妥協できるものと判断される。

そこで（１）サイン波のほかに、21倍音までもつ人工の複合音として、（２）基音とその奇数倍の倍音（第n倍音の振幅は基音の１/n）からなる矩形波（方形波）、（３）基音とその整数倍の倍音（振幅は矩形波と同じ）からなる鋸歯状波（ノコギリ波）、（４）基音とその整数倍の倍音（振幅はすべて基音と同じ）からなる「等倍音」波を作成する。21倍音が可聴域内（～22kHz）に収まるためには、基音は1000Hz程度でなければならないから、想定している発音域はFig. 1 に示すとおりである。なお、折り返しのリスクを低減するため、このでのサンプリング周波数は96kHzとしている。

さて、実際の波高計算はList 2 に示しているが、1：の$tを時刻点として波高を求めている。

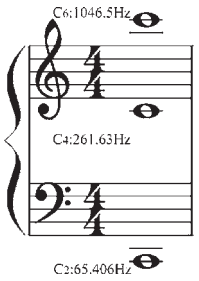

Fig. 1　発音域の下限C_2と上限C_6

3.3 オシレータ（List 3）$drを１音の長さとして、１音中の標本点数$tpを求め、そのすべてについて波高計算して@daに格納している。

3.4 エンベロープ・ジェネレータ（List 4）**1**：の$atはアタック・タイム、**2**：の$dtはディケイ・タイム、**3**：$rtはリリース・タイム（sec）、$slはサステイン・レベル（[0.0, 1.0]）である（Fig. 2）。**1**：～**3**：は各タイムから標本点数を求めたもので、**4**：のサステインについては総点数から各点数を減じている。**5**：の$pは求めるべき系列位置を指している。**6**：～**H**：では１

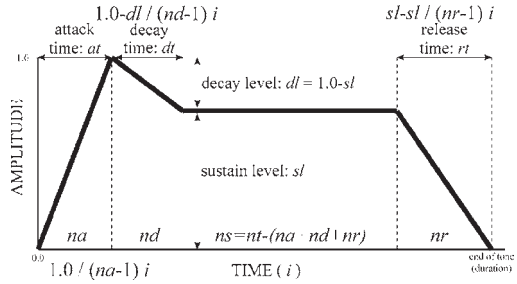

Fig. 2　ADSRエンベロープジェネレータのパラメタ

次関数によって各区間のエンベロープを定めている。

3.5 実数->整数変換(List 5)ここまでの処理で波形を求めることはできたが、数値精度の確保と丸めの誤差の回避のために、すべてを実数計算で行ってきた([-1.0, +1.0])。一方、WAVファイルに限らずデジタル・オーディオでの数値表現は整数であるから、実数を整数に変換する必要がある。WAVファイルにおいては、8-bit量子化(分解能)の場合は、C言語でいうunsigned char(byte)型であるため、正の整数で0～255の値をとる。そのために **1** ：で実数$vの0.0を128に対応させて、負の最小値を0、正の最大値を255となるように変換する。一方、16-bit量子化においてはsigned int(short) が使われるため、 **2** ：のように変換に際して特に注意を払う必要はない。なお、その際に0.5を加算しているのは、四捨五入によって誤差を軽減するためである。

3.6 整数->文字(List 6)WAVファイル内のデータ本体部分は、区切り記号や制御文字を一切含まない整数値の連続的な並びである。そのため整数化されたデータを文字コードとみなして、8-bit量子化では半角文字、16-bit量子化では全角文字に変換する。表現を換えれば、整数型データをそのままで文字型データとして扱うことになる。このような変換はPerlのpack関数を用いればよく(**1** ：～ **4** ：)、ステレオの場合は左ー右の順で並べられる。

3.7 ファイル保存と終了化(List 7) **1** ：で文字列化されたWAVデータを読み込み、 **2** ：でWAVファイルとして必要なヘッダ情報などを付加してファイルに書き出し、最後に **3** ：でデバイスを解放する。

List 5 実数->整数変換

```
1: return int(127*$v+128+0.5);        # 8-bit unsigned
2: return int(32767*$v+0.5);          #16-bit signed
```

List 6 整数->文字変換

```
1: return pack("C",$ls);              # 8-bit / mono
2: return pack("CC",$ls,$rs);         # 8-bit / stereo
3: return pack("s",$ls);              #16-bit / mono
4: return pack("ss",$ls,$rs);         #16-bit / stereo
```

List 7 ファイル保存と終了化

```
1: $WAVE->Load( $wd );                # データ読込
2: $WAVE->Save('test.wav');           # WAV ファイル保存
3: $WAVE->Unload();                   # SOUND I/F 解放
```

4．作成結果の検証

ここまで述べてきた手続きによって作成されたWAVファイルが、意図通りに仕様をみたすものであるか、また、音律に関する心理学研究での利用に耐えうるものであるのかについて、以下で順次検討していきたい。

4.1 周波数精度　発振周波数の精度の確認するため、440Hzのサイン波120秒のWAVファイルを作成し、PCからのアナログ・オーディオ出力をユニバーサル・カウンタ(Takeda Riken： TR-5151)に取り込み、測定周期10秒で周波数測定を行った(したがって周波数分解能0.1Hz)。その結果、12回の測定ともすべて440Hzを示していた(Fig. 3)。1000Hzにおける1セントは0.5778Hzに相当するため、音律研究に十分耐えうる周波数精度だといえるであろう。また、周波数測定は440Hzでしか行っていないが、原理的に考えて問題ないと判断される。

Fig. 3　ユニバーサル・カウンタによる周波数の確認

4.2 波形　作成されたWAVファイルをデジタル・オーディオ・エディタAudacity Ver. 2.0.3に読み込ませて、その波形の一部分を表示させたものをFig. 4-1(65.406Hz)とFig. 4-2(1046.5Hz)に示す(上からサイン波、矩形波、鋸歯状波、「等倍音」波)。理想波形を直接求めるのではなく、21倍音までに限定して逆フーリエ変換を行った場合は、当然のことながら近似波にしかならないが、それぞれの波形の名称に応じた基本的な輪郭は再現されている。また、サンプリング周波数を96kHzとしているため、想定している発音域の下限(65.406Hz)と上限(1046.5Hz)で、波形にほとんど違いがない点も注目される。

4.3 スペクトル　つぎに作成されたWAVファイル内の波形データに対して、フーリエ解析を行った結果をFig. 5-1(65.406Hz)とFig. 5-2(1046.5Hz)に示す(上からサイン波、矩形波、鋸歯状波、「等倍音」波)。その際に使用したのは統計解析ソフトウェア「R」と、その機能拡張のためのライブラリ「tuneR」であるが、

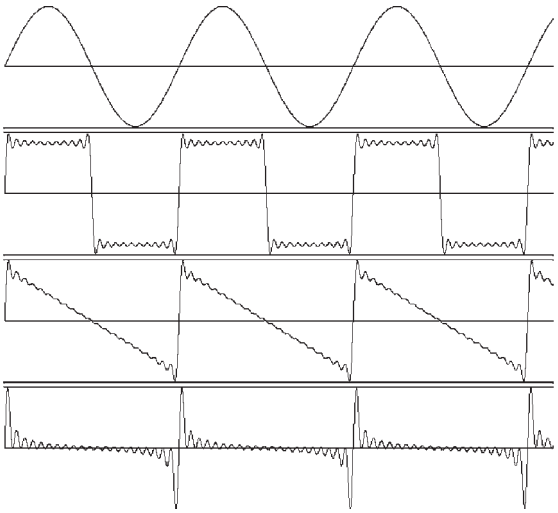

Fig. 4－1　WAVファイル内の波形(65.406Hz)

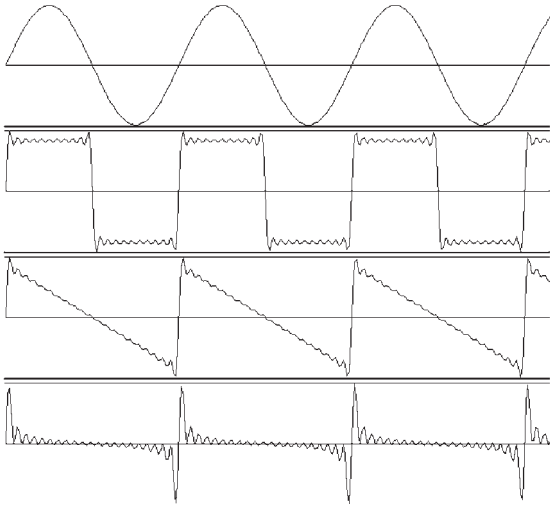

Fig. 4－2　WAVファイル内の波形(1046.5Hz)

List 8「R」と tuneR によるフーリエ解析

```
1: library("tuneR")
2: Wav=readWave("filename.wav")
3: Dat=Wav@left
4: Pwr=abs(fft(Dat))
5: plot(Pwr, type="1")
```

List 8 に示すようにWAVファイルから波形データ部分だけを抽出し、FFT解析後、その結果をプロットすることが極めて簡単に行えるのである。

これらの結果によると、自明なことではあるが、21倍音を越える周波数成分はまったく含まれていないことが確認できる。さらに、最高音の基音である1046.5 Hzに対して十分高いサンプリング周波数を設定しているため、１周期当たりのサンプル数が少なくなりことによって生じる波形の歪みも特に認められていない。なお、スペクトルの表示に若干の乱れがあるのは、フーリエ変換の際のパラメタ設定を、Rの関数であるfftのデフォールト値としたためで、audacityで適切なパラメタを与えると、期待通りの表示が得られることが確認されている。

4.4 DA-AD変換後のスペクトル　WAVファイルをDA変換により再生(アナログ化)して、それを入力とするAD変換により録音(デジタル化)した。その際に使用したPCに搭載されているsound I/Fの周波数特性をFig. 6 に示す(WaveGeneとWaveSpectraというフ

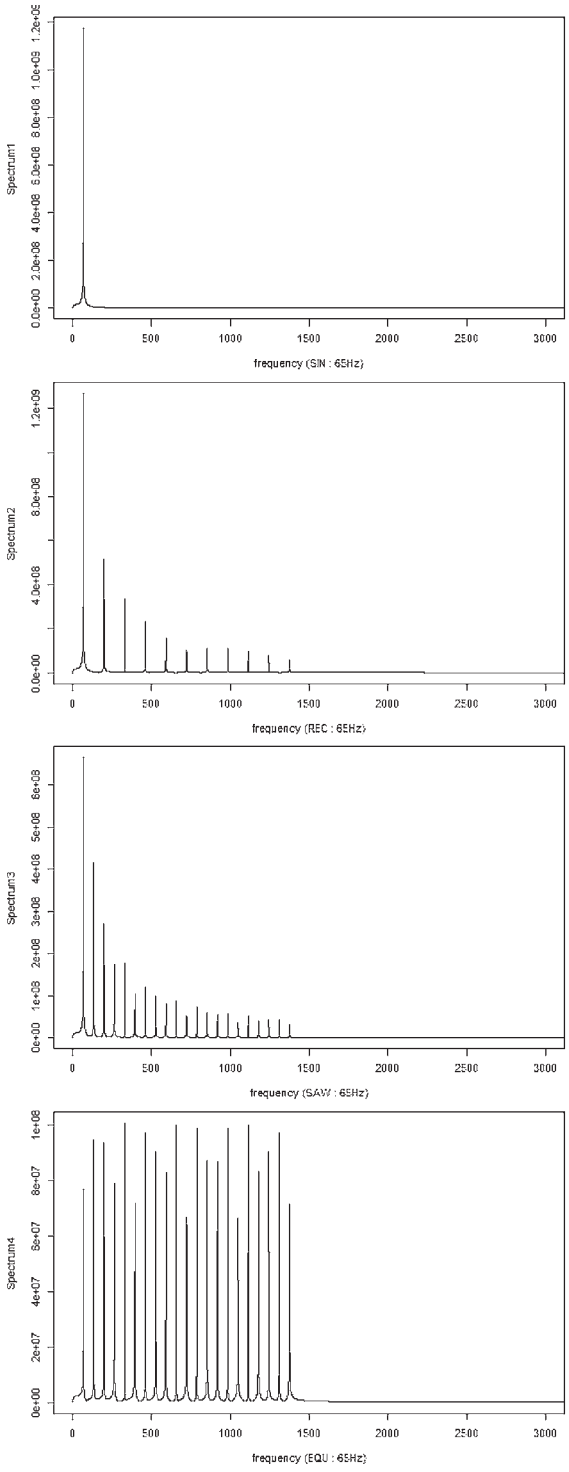

Fig. 5－1　WAVファイルのフーリエ解析(65.406Hz)

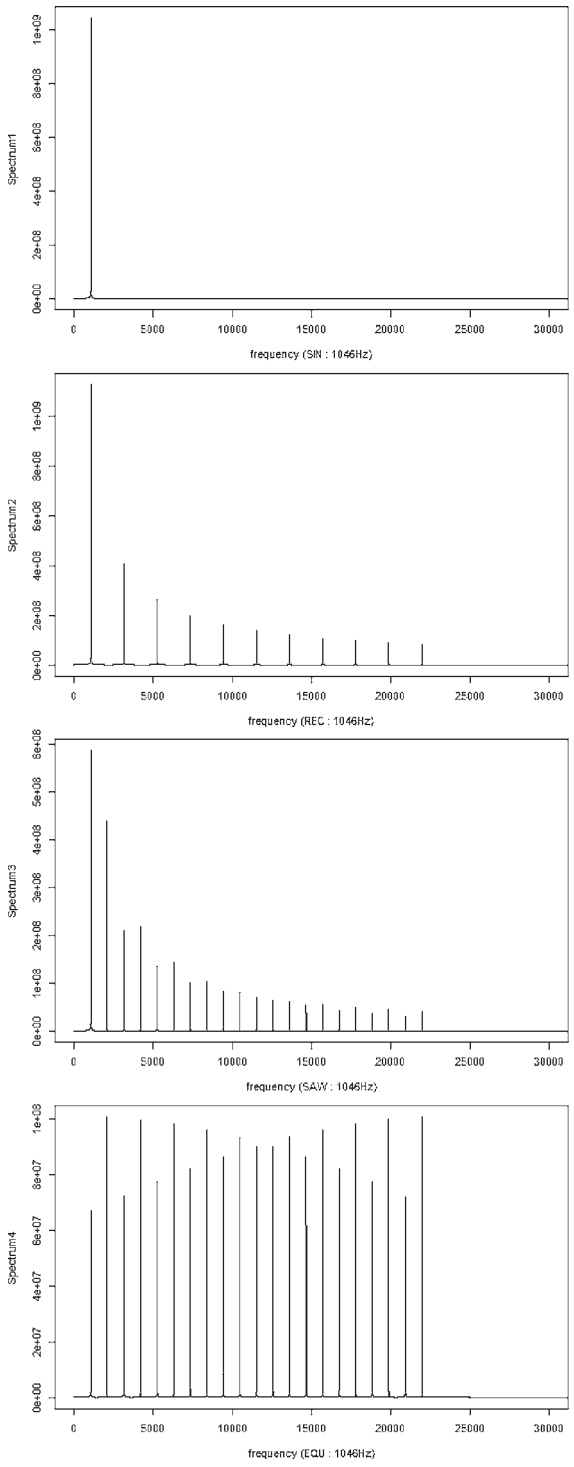

Fig. 5－2　WAVファイルのフーリエ解析(1046.5Hz)

リーウェアの組合せで測定)。それによると低域に厚みをもたせ、高域を刺激的にするような音作りがなされているといえるであろう。また15kHzあたりをから急激に減衰しているは、AD変換側のアンチ･エイリアス処理によるものと推察される。なお、これらの特徴は使用するPCに搭載されているsound I/Fによって異なることはいうまでもない。

さてDA変換出力を、そのままAD変換することで作

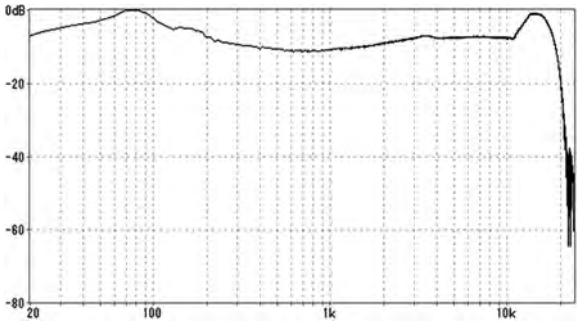

Fig. 6　DA-AD変換用PCのsound I/Fの周波数特性

成したWAVファイルを、先ほどと同じ手順でフーリエ解析した結果をFig.7-1(65.406Hz)とFig.7-2(1046.5Hz)に示す(上からサイン波、矩形波、鋸歯状波、「等倍音」波)。25kHzより上に-40～-60dBの水準で予期せぬ周波数成分がみられるが、可聴範囲を超えた領域であるため、ここでは一応は問題なしと判断しておく(今後の検討課題ではある)。また、1046.5Hzについては、高次の倍音がsound I/Fの周波数特性上での減衰域に入っているため、レベルの低下が起こっていることも読み取れる。この点が気になる場合は、高音域での減衰が少ないPC(sound I/F)の使用を検討するか、あるいは使用する音域を低めに設定すること(たとえば上限をC５の523.25Hzとする)などが必要となる。

4.5 エンベロープ　ここでの開発においては、エンベロープ・ジェネレータも組み込まれているため、その動作を確認しておく必要がある。そのため作成されたWAVファイルをAudacityで読み込み、時間軸を圧縮してエンベロープを可視化ものをFig. 8 とFig. 9 に示す。これらの図においては、グレースケールで色合いが薄い(白に近い)方がエネルギー密度の高いことを表している。Fig. 8 においては古典的かつ典型的なADSRエンベロープが、Fig. 9 においては持続音から減衰音までが例示されており、想定通りの結果が得られていることが確認されている。

4.6 音量補正　このようにして作成した音を実際に何度も聞いていると、最大振幅を一定にした場合、主観的な音量感に大きな違いがあることは明らかである(Fig.10の図上も参照)。この違いは波形が異なることによるエネルギーの差によるものであるから(振幅でいえば実効値の差)、21倍音波について数値積分により実効値を求めた。それらをもとにして、使用する波形の組合せの違いを考慮した音量補正係数を求めたものをTable 1に示す(Fig.10の図下も参照)。エネルギー密度が高い薄いグレーの領域が均衡化されており、主観的ではあるが試聴によっても、そのことが確認されている。

4.7 検証結果のまとめ －結論－　ここまで本論文で述べた方略によって作成されたWAVファイルについて、その特性を客観的に分析することにより妥当性の検証を行ってきた。その結果、(１)発振周波数の精度

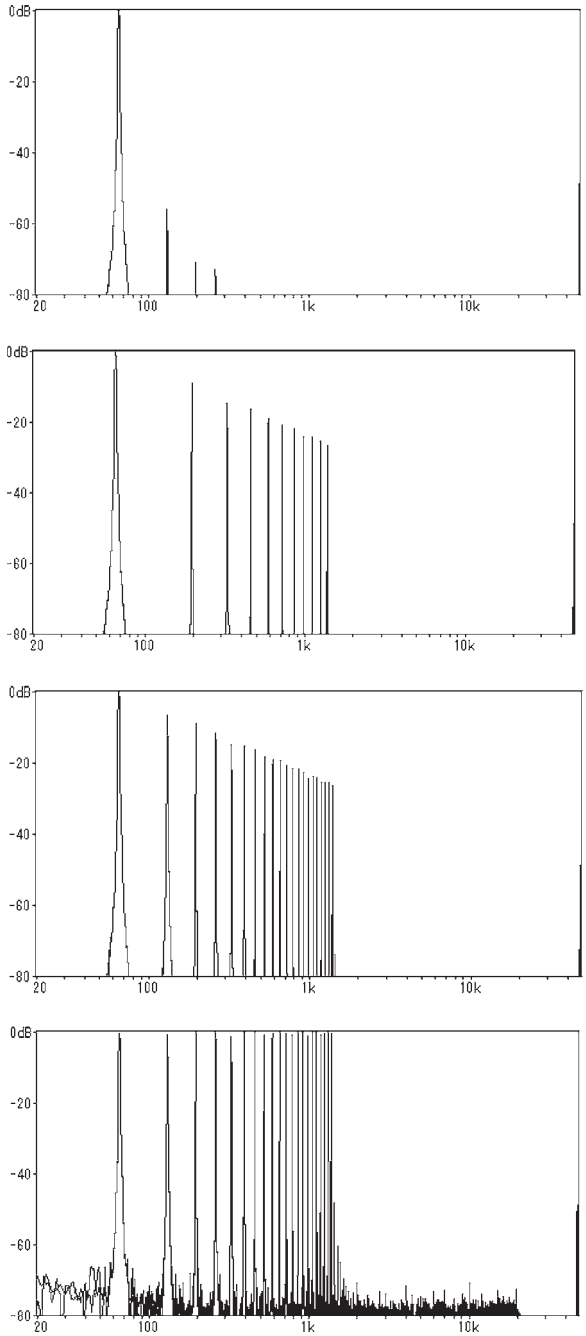

Fig. 7－1　DA-AD変換後のフーリエ解析(65.406Hz)

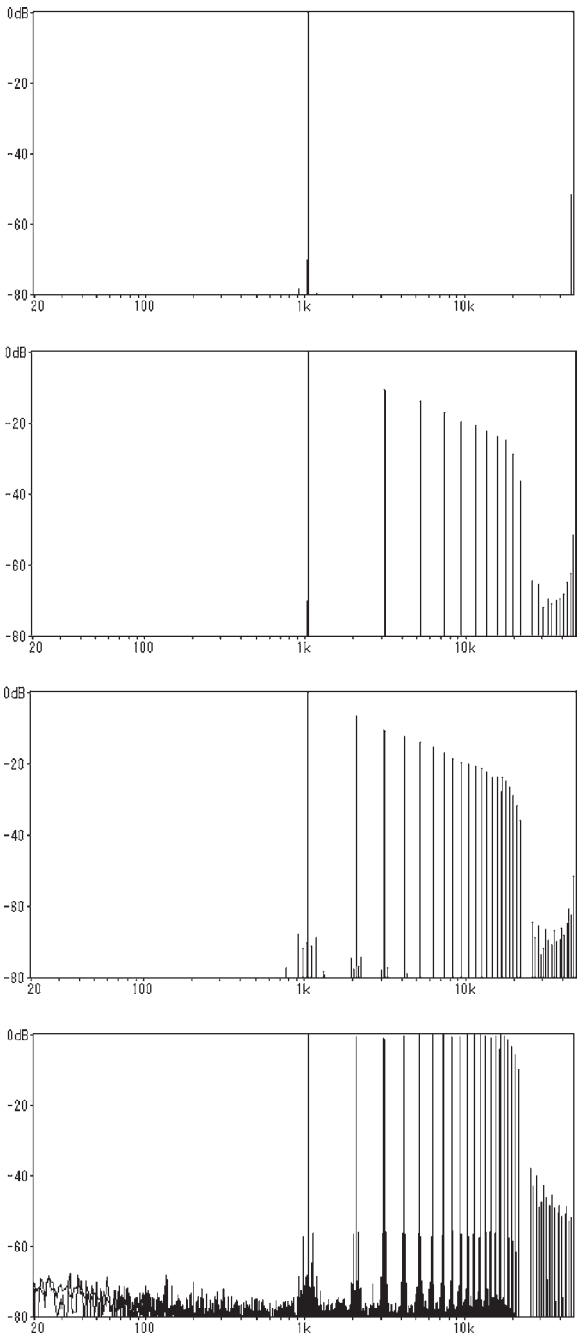

Fig. 7－2　DA-AD変換後のフーリエ解析(1046.5Hz)

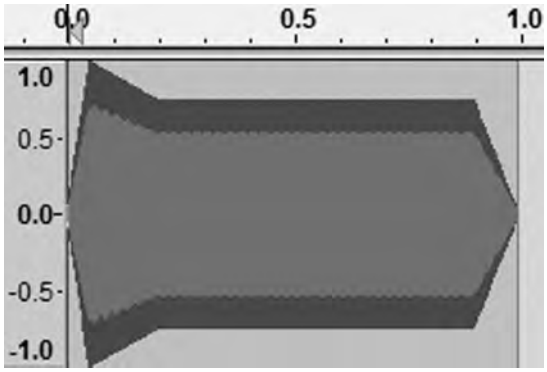

Fig. 8　作成されたADSRの典型例

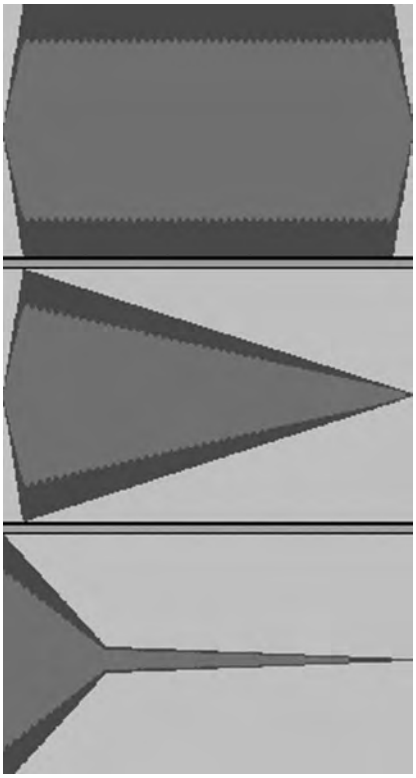

Fig. 9　エンベロープの作成例

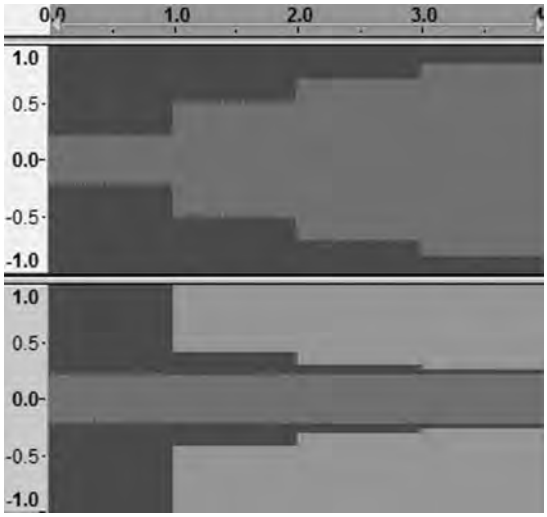

Fig.10　音量補正前(上)と補正後(下)のエンベロープ

Table 1　音量補正のための係数表

項目	波種	サイン波	矩形波	鋸歯状波	「等倍音」波
絶対値平均	理想波	0.637	1.000	0.500	0.000
	21倍音波	-----	0.832	0.433	0.095
実効値	理想波	0.707	1.000	0.577	0.000
	21倍音波	-----	0.840	0.502	0.208
修正係数1	21倍音波	0.294	0.248	0.415	1.000
修正係数2	21倍音波	0.710	0.598	1.000	non-use
修正係数3	21倍音波	1.000	0.842	non-use	non-use

は必要かつ十分であること、(2)目視の範囲で確認できる波形に問題はないこと、(3)DA変換前のスペクトルは設計通りであること、(4)DA変換後に再AD変換を行ったもののスペクトルが実用上は問題ないと判断できる範囲内であったこと、(5)実装したエンベロープ・ジェネレータは想定通り機能していたこと、(6)波形に違いに対する音量補正は適切に行えること、などが明らかになった。これらをもって、本研究で提案した人工音のWAVファイル作成法は、音律研究に問題なく使用できるものとの結論に至った。

5．要約

音律に関する心理学的研究で使用することを前提に、研究目的に耐えうる精度をもった音響刺激を作成する方法の一つとして、波形情報を数値計算で定めてから、それをWAVファイル化する方法、および作成されたデータの適切さに関する検証結果について報告した。開発環境はWindows上のPadre IDE+Strawberry Perlであり、さらにCPANモジュールのWin32::Soundを使用した。その結果、人工音であるサイン波、矩形波、鋸歯状波、「等倍音」波に関しては、エンベロープ(ADSR)設定も含めて、要求される制度を満たすものが、比較的容易に作成可能であることが明らかになった。

6．参考文献

菅　千索　2011、音楽心理学研究における異なる音律の利用について－ピタゴラス音律、中全音律、平均律－、和歌山大学教育学部紀要－教育科学－、第61集、1～9頁。

菅　千索　2012a、音楽心理学研究における異なる音律の利用について(2)－純正律とウェル・テンペラメント－、和歌山大学教育学部紀要－教育科学－、第62集、9～16頁。

菅　千索　2012b、〔解説〕音律について、音楽知覚認知研究、第18巻第1号・第2号合併号、53～88頁。

菅　千索　2013a、数値計算による音響刺激WAVファイルの作成法、音楽知覚認知学会平成25年度春季研究発表会(岡山大学)。

菅　千索　2013b、数値計算による音響刺激WAVファイルの作成法(2)、音楽知覚認知学会平成25年度秋季研究発表会(東京情報大学)。

田辺義和　2001、Windowsサウンドプログラミング、翔泳社。

注：本論文は菅(2013a)および菅(2013b)において口頭発表した内容をもとに作成されたものである。