# 和音性の計算法と曲線の描き方
## —不協和度・緊張度・モダリティ—

藤澤　隆史[*1]　　ノーマン D. クック[*2]

**要　旨**

和音（harmony）は，メロディ（melody），リズム（rhythm）とともに音楽を形作る重要な要素である．音楽の物理的な音響的特徴とその心理的な印象や感性との関連性について定量的に評価するために，和音性についての評価モデルを構築した．和音性は，(1)協和度（心地よい－わるい，澄んだ－濁った），(2)緊張度（緊張した－落ちついた），さらに長調か短調かといった性質を決定する，(3)モダリティ（明るい－暗い，うれしい－悲しい）から構成される．本資料では，Cook & Fujisawa (2006) において議論されている和音性知覚モデルの詳細について補足的な説明を行う．具体的には，それぞれの和音性曲線（不協和度，緊張度，モダリティ）を得るための心理数理モデルの詳細と，実際に曲線を描くための計算プログラム（C 言語・MATLAB）について解説する．

# How to Calculate "Harmoniousness" and Draw the Curves: Dissonance, Tension and Modality

Takashi X. FUJISAWA[*1]　　Norman D. COOK[*2]

**Abstract**

In this paper, we provide further details on the harmony model discussed in Cook & Fujisawa (2006). The model curves for calculating the total "harmoniousness" (dissonance, tension and modality) of chords are described and the program (in the C language and MATLAB) needed for drawing the theoretical curves is presented.

[*1]関西学院大学ヒューマンメディア研究センター　　[*2]関西大学総合情報学部

## はじめに

音楽心理学の分野において和音は非常に重要な現象であるにもかかわらず，これまで心理物理に基づいた明瞭なモデルは構築されてこなかった．もちろん，古典的な和音理論で記述されたように現象としてはよく知られているし，和音が協和的か不協和的かといった点では，それは主に2音のピッチ差（音程）に基づいた心理物理モデルが構築されてきた．しかしながら，長調の和音は明るく・楽しい感じで聴こえ，また短調の和音は暗く・さみしい感じで聴こえるといったように，私たちは和音から豊かな主観的印象を受けるにもかかわらず，これまでの先行研究において三和音についての知覚現象を定量的にモデル化したものは皆無であった．そこで，Cook & Fujisawa (2006) [1] は三和音の知覚現象を定量的に評価するための心理物理モデルを構築し，またそのモデルを用いて他の音響現象を評価するという試みも行ってきた [2], [3]．本資料は，Cook & Fujisawa (2006) において議論された和音性知覚の定量化モデルの詳細について補足的に説明するものであり，提案された3つのモデル（不協和度，緊張度，モダリティ）とそれぞれの知覚曲線を実際に描くための計算プログラム（C言語，MATLAB）について解説するものである．

### 1. 周波数と音程

ある2つの純音がもつピッチの距離（ピッチ距離）には様々な算出法が考えられうる．純音のピッチは，ほぼその音の周波数に対応すると考えてよいから，ピッチ距離の定義として最も単純なものとして，ある2つの純音の周波数 $f_1$, $f_2 (f_1 < f_2)$ の差（周波数差：$f_2 - f_1$）を挙げることができるだろう．ただし物理的な性質から定義される周波数差と，そこから得られる心理的な感覚差は一義的に対応しないことが分かっている．例えばピアノの鍵盤上で順に $C_3$ と $G_3$, $C_4$ と $G_4$ はともに5全音（7半音）であり，同じぐらいの距離であると感じられる．それぞれの周波数差を計算してみると $C_3$ (130.81 *Hz*) と $G_4$ (196.00 *Hz*) の差は65.19 *Hz* であり，$C_4$ (261.63 *Hz*) と $G_4$ (392.00 *Hz*) の差は130.37 *Hz* であることが分かる．

私たちが感じる音の等間隔性が音程（interval）と呼ばれるものである．ある2つの音が純音である場合，一般的にその音程はその周波数の比を対数変換したものに対応することが分かっている．したがって，ある2つの周波数 $f_1$, $f_2 (f_1 < f_2)$ が与えられた場合，その音程 ($x_{12}$) は以下の式で表現される．

$$x_{12} = 12 \log_2\left(\frac{f_2}{f_1}\right) \approx 39.863 \log_{10}\left(\frac{f_2}{f_1}\right) \tag{1}$$

式(1)によって定義される音程（ピッチ距離）の単位を12平均律半音，または単に半音（*semitones*）と呼ぶ．例えばMiddle C (261.63 *Hz*) とC# (277.18 *Hz*) の音程を計算みると1.00半音となることが分かる．本資料内では特に断らない限り，基本的に音程単位はこの半音（*semitones*）を用いる．

Ellisによってもたらされた音程単位である*cents*は半音基準を100倍したものである．したがってCとC#（例：$C_4$と$C_{4\#}$）の場合，そのピッチ距離は1.00半音，100*cents*となり，オクターブ関係にあるCとC'（例：$C_4$と$C_5$）の場合，そのピッチ距離は12.00半音，1200*cents*という互換関係がある．

## 2. 不協和度曲線

### 2.1　不協和度の定義と基本曲線の性質

本研究において，2つの音の不協和度を決定する式は，基本的にはPlomp & Levelt (1965)[4]のものに基づいており，その見解に基づいたKameoka & Kuriyagawa (1969)[5]，Sethares (1993; 1999)[6],[7]らで用いられているものとも，ほぼ同一のものである．

$$d = v_{12}\alpha_3[\exp(-\alpha_1 x_{12}{}^{\beta}) - \exp(-\alpha_2 x_{12}{}^{\beta})] \tag{2}$$

ここで，$x_{12}$は2音の周波数$f_1$, $f_2$($f_1 < f_2$)から式(1)から導かれる音程（ピッチ距離）であり，$v_{12}$は2音の音量$v_1$, $v_2$から定義される値である．本資料ではSethares (1993)[6]にしたがい$v_1$, $v_2$の積としている(1)(2)．例えば音1の音量が0.8で，音2の音量がその半分の0.5である場合，その音の不協和度の強さは0.4倍される．またどちらかの音量が非常に小さい場合（ほとんど0である場合），積をとることで，その音のペアが有する不協和度も限りなく0に近づく．

### 2.2　倍音の構造と不協和度曲線

本節では，倍音を含めた場合における不協和度について検討していく．ある音の基本周波数を$f_0$とした場合，その$i$番目の高次周波数$f_i$（倍音）は，以下のように定式化できる．

$$f_i = f_0 \cdot (i+1) \qquad (\text{ただし，} i < n) \tag{3}$$

ここで$n$は，計算のために含められる倍音の数である．また$i$番目の高次周波数の相対的音量$v_i$は，基本周波数の音量を$v_0$とした場合，以下の式にしたがって減衰するものとする[6]．

$$v_{i+1} = 0.88 \cdot v_i \qquad (\text{ただし，} i < n) \tag{4}$$

---

(1) その他にも，$v_1$と$v_2$の平均$ave(v_1, v_2)$や平方根$sqrt(v_1, v_2)$をとる方法，最小値$min(v_1, v_2)$や最大値$max(v_1, v_2)$，低い音の音量に合わせる（$v_{12} = v_1$），音量効果は考慮しない（$v_{12} = 1.00$），など様々な方法が考えられるが，選択について決め手となる理論的および経験的根拠はない．以下の2点に注意して最適なモデル選択を行うべきである．曲線形は，1．どの関数を選択しても$x$軸方向の値（山や谷の位置）は大きく変化することはない．2．選択する関数によって$y$軸方向の値（不協和度の大きさ）はいくぶん異なってくる．

(2) 例えば$min(v_1, v_2)$の場合，$n$倍音の不協和度への貢献は最低限へと制限する，というモデルを選択したことに等しい．その結果，具体的には11半音（例：$C_4$と$B_4$）の不協和度は1半音（例：$C_4$と$C^{\#}_4$）の1/3程度となる．種々の心理実験では11半音程は比較的不協和度が高いという結果が得られていることから，実験データと整合性のある不協和度曲線を構築するためには，この音量関数の選択は1つのポイントである．

音量の減衰関数には式(4)以外にも様々なもの存在するが[3]，いずれにしろ重要な点は，関数の減衰形が不協和度曲線の形を大きく左右することはない．式(4)により，倍音の相対的音量は1.00($F_0$)，0.88($F_1$)，0.78($F_2$)… と徐々に倍音の音量が減衰していく．

倍音成分が以上のように定義されたとき，倍音成分を含めた2音の組み合わせによって構成される音程の不協和度が計算できる．音1におけるi倍音の周波数を$f_{1i}$，音2におけるj倍音の周波数を$f_{2j}$とし，またそれぞれの音量を$v_{1i}$, $v_{2j}$とした場合，複合音の不協和度（$D$）は以下のように定式化できる．

$$\begin{aligned} D &= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} d(f_{1i},\ f_{2j},\ v_{1i},\ v_{2j}) \\ &= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} d(x_{ij},\ v_{ij}) \end{aligned} \tag{5}$$

ここで$x_{ij}$は$f_{1i}$と$f_{2j}$によって定義される音程を表し，$v_{ij}$は$v_{1i}$と$v_{2j}$によって定義される値である．

### 2.3 不協和度曲線を描く計算プログラム

以上の節までに述べてきた不協和度の計算方法とパラメータ設定による計算プログラムを2種類以下に示す．1つ目はC言語を用いたもので，プログラムを実行すると，図1に示したように，音程の値とその音程から計算される不協和度の値が得られる．$\alpha_1$, $\alpha_2$, $\alpha_3$, $\beta$の各種パラメータは，順に定数a，b，c，dで表現されており，パラメータの値を変更するためにはプログラム内で変更する必要がある．音程のステップは0.1半音ずつ進行する設定となっているが，これもプログラム内のinc変数の値によって変更することができる．例えば1.0半音刻みで不協和度の値を得たい場合には，inc=1.0と設定しなおせばよい（図2）．2つ目はMATLAB[9]を用いたもので（図3），実行すると$F_0$から$F_5$を含めたときの不協和度曲線が描かれる（図4）．

| Interval | Tab | Dissonance |
|---|---|---|
| 0.0 | Tab | 0.000 |
| 0.1 | | 0.252 |
| 0.2 | | 0.454 |
| 0.3 | | 0.614 |
| : | | : |

図1　プログラム実行後の計算結果の例．

(3) 倍音の音量減数関数についても，またいくつかのモデルが仮定できる．音量は減衰しない（$v_i = 1.00$）とする仮定，$i$+1倍音の音量は常に$i$倍音の半分となる（$v_i = v_0 * \{1/(i+1)\}$）とする仮定[8]などがある．

```
/* d_curve.c: How to Draw Dissonance Curve */

#include <stdio.h>
#include <math.h>

#define N 5                         /* F0-F5 までを含む */

#define a 0.70      /* 不協和度計算式のパラメータ */
#define b 1.40
#define c 4.00
#define d 1.25

float freq1[N],freq2[N];
float amp1[N],amp2[N];
char br;

void main(){

  int i,j;
  float loop;
  float inc=0.1;     /* 12 半音まで 0.1 半音刻み */
  float x12,v12;
  float d12,Dtotal;
  float sum1,sum2;

  printf("Interval¥tDissonance¥n");
  for(loop=0;loop<=12.1;loop=loop+inc){

    freq1[0]=261.63;     /* 低音の周波数（固定，Middle C） */
    freq2[0]=pow(10,loop/39.863)*freq1[0];     /* 高音は loop 半音離れた周波数 */

    for(i=0;i<N;i++){
      /* 倍音の周波数と音量の設定 */
      freq1[i]=freq1[0]*(i+1); freq2[i]=freq2[0]*(i+1);
      amp1[i]=pow(0.88,i); amp2[i]=pow(0.88,i);
    }

    sum2=0.0;Dtotal=0.0;
    for(i=0;i<N;i++){
      sum1=0.0;
      for(j=0;j<N;j++){
        x12=fabs(39.863*log10(freq2[j]/freq1[i]));     /* 周波数から音程へ */
        v12=amp1[i]*amp2[j];     /* 音量の設定関数 */
        d12=v12*c*(exp(−a*pow(x12,d))−exp(−b*pow(x12,d)));     /* 不協和度の計算式 */
        sum1=sum1+d12;
      }
      sum2=sum2+sum1;
    }
    Dtotal=sum2/2.0;
    printf("%4.1f¥t%6.3f¥n",loop,Dtotal);
  }

}
```

図 2　不協和度計算プログラム（C 言語）.

```
01 % d_curve.m: How to Draw Dissonance Curve
02 for n = 1:6; for k = 0:0.01:12;
03 LF0 = 261.63;
04 HF0 = 2^(k/12)*261.63;
05 freqL = LF0.*[1:1:n];   freqH = HF0.*[1:1:n];
06 ampL = 0.88.^[0:1:n-1]; ampH = 0.88.^[0:1:n-1];
07 X = 12*log2((1./freqL)'* freqH);
08 V = ampL'*ampH;
09 D=zeros(n); D=V.*4.0.*(exp(-0.7.*abs(X).^1.33)-exp(-1.4.*abs(X).^1.33));
10 hold on; plot(k,1/2*sum(sum(D)),'k')
11 end; end;
12 xlabel('Interval(semitones)'); ylabel('Dissonance(D)'); grid on;
```

図3　不協和度計算プログラム（MATLAB）.

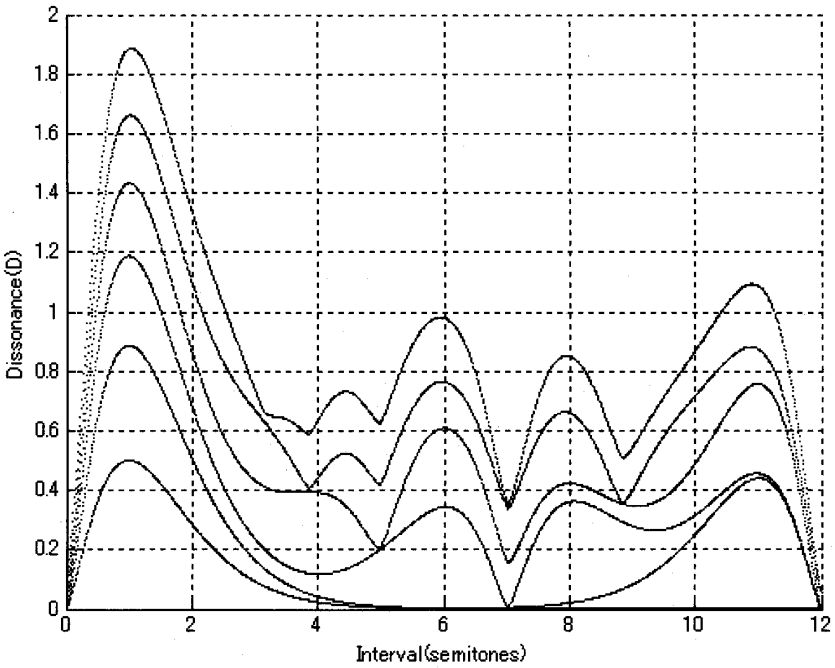


図4　プログラムの実行結果（MATLAB）.

## 3. 緊張度曲線

### 3.1　音程の等しさと緊張度曲線

和音性にかかわる新たな性質として，3音のピッチの相対的音程構造から定義される“緊張度”や“モダリティ”が存在することについてはCook & Fujisawa (2006)，または他の箇所において議論した[10], [11]．本節では，緊張度やモダリティの定義にかかわる“音程の等しさ(intervallic equivalence)”という概念を再確認した後に緊張度の定義を行い，最後に緊張度の値と理論曲線を得るためのプログラムについての解説を行う．

“緊張度”や“モダリティ”という概念は，三和音がなぜ明るかったり，暗かったり，緊張的なものであったりという知覚的性質をもつのかについて定量的に説明するためのものであり，

その知覚的性質は三和音におけるピッチの相対的構造に基づいている．まず，和音がもつ緊張感はMeyer（1969）の古典的知見に基づいたもので[12]，3音のそれぞれのピッチ距離（音程）がどれほど等しいかによって決定される．最もピッチの低いものと中間のものから構成される音程を第1音程，中間のものと最もピッチが高いものから構成される音程を第2音程とした場合，その構造は図5のように図示できる．

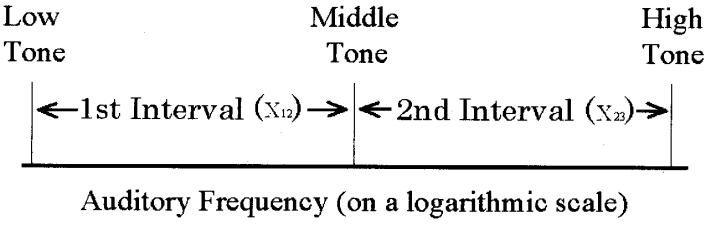


図5　第1音程（x12）と第2音程（x23）．

Cook & Fujisawa (2006)[1]において議論したように，第1音程と第2音程の値がそれぞれ等しい場合，それらの音は最も“緊張性”の高い，未解決的な響きをもつ和音として知覚される．逆に，それぞれの音程の値が等しくない場合には，“緊張性”の低い，解決的なものとして知覚される．

純音である，ある3つの音の周波数$f_1$, $f_2$, $f_3$($f_1 < f_2 < f_3$)において，$f_1$と$f_2$から定義される音程を$x_{12}$, $f_2$と$f_3$から定義される音程を$x_{23}$とし，またそれぞれの音量を$v_1$, $v_2$, $v_3$とした場合，緊張度（$t$）は以下のように定式化される．

$$t = v_{123}\exp\left[-\left(\frac{x_{23}-x_{12}}{\gamma}\right)^2\right] \tag{6}$$

ここで，$v_{123}$は3音の音量$v_1$, $v_2$, $v_3$の積である．また$\gamma$は定数で0.60である．3音の音量をそれぞれ1とした場合（$v_1 = v_2 = v_3 = 1$），式(6)によって定義される緊張度（$t$）の理論曲線を図6に示す．

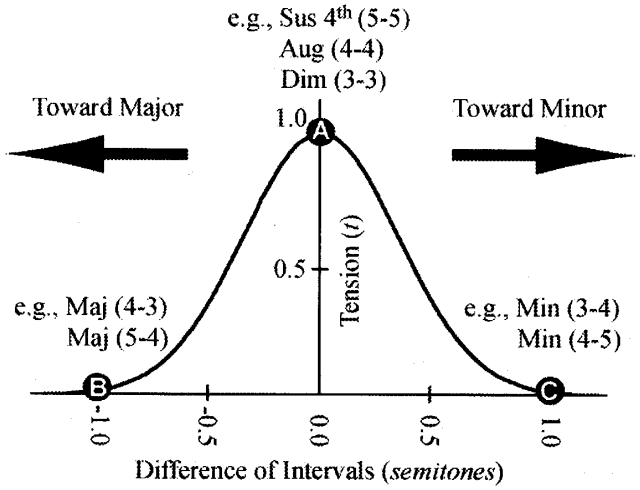


図6　緊張度（$t$）の理論曲線．

また，三和音を構成するそれぞれの音には倍音成分が含まれることから，それぞれの倍音の周波数が式(3)によって定義され，またそれぞれの音量が式(4)によって定義された場合，複合音三和音の緊張度（$T$）の計算が可能となる．音1のi次周波数を$f_{1i}$，音2のj次周波数を$f_{2j}$，音3のk次周波数を$f_{3k}$とし，またそれぞれの音量を$v_{1i}$，$v_{2j}$，$v_{3k}$とした場合，その緊張度（$T$）は以下のように定式化される．

$$T = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} t(x_{ij},\ x_{jk},\ v_{ijk}) \tag{7}$$

ここで，$x_{ij}$は周波数$f_{1i}$と$f_{2j}$によって定義される音程（第1音程），$x_{jk}$は$f_{2j}$と$f_{3k}$によって定義される音程（第2音程）である．また$v_{ijk}$は音量$v_{1i}$，$v_{2j}$，$v_{3k}$の積である．

### 3.2 緊張度計算プログラム

3.1節で述べた緊張度（$T$）の計算方法とパラメータ設定による計算プログラム2種類を以下に示す．1つ目はC言語を用いたものである（図7）．緊張度およびモダリティは，第1音程と第2音程の2変数から定義される特性であることから，すべての変数間関係（第1音程，第2音程，緊張度）を表示するためには3次元である必要がある．本プログラムでは第1音程をある固定の値とすることで，不協和度曲線を描いたときと同様に，第2音程の値を少しずつ変更したときの緊張度（$T$）を検討するものとなっている．プログラム中では，第1音程と第2音程の変数名はそれぞれlow_interval，high_intervalと表示されており，また第1音程の値は4.0と固定されている．含まれる倍音の数は6行目において，またパラメータ$\gamma$の値は48行目において直接0.60と設定されているので，変更の場合にはそれぞれ書きかえる必要がある．音程刻みの値は，不協和度プログラムと同様にinc変数の値を設定しなおすことで変更することができる．2つ目はMATLABを用いたもので（図8），実行すると$F_0$から$F_5$を含めたときの緊張度曲線が描かれる（図9）．

## 4. モダリティ曲線

### 4.1 モダリティの定量化

“モダリティ”は長短調の次元にかかわる知覚的性質であり，緊張度と同様に第1音程と第2音程の相対的構造から定義される和音性である．緊張度は3音それぞれのピッチの等間隔性に基づく知覚的性質であることは3.1節で述べた．モダリティの知覚的性質はこの等間隔性が崩壊したときに生じる知覚的性質であると仮定する．例えば，Cを根音とした増和音はC-E-G#から構成される．このとき第1音程と第2音程はそれぞれ4半音であり（4−4），またその音程差は0であることから，増和音独特の緊張感がもたらされる．次に，いちばん高いピッチであるG#を半音下げてGとしてみるとC-E-Gとなり，これは長調の和音の基本型であることが分かる．このとき第1音程は4，第2音程は3へと変化し（4−3），その音程差は＋1という値となる．それに対して，いちばん低いピッチであるCを半音上げてC#としてみると，

```
01  /* t_curve.c: How to Draw Tension Curves */
02
03  #include <stdio.h>
04  #include <math.h>
05
06  #define N 6      /* F0-F5 までを含む */
07
08  float freq1[N],freq2[N],freq3[N];
09  float amp1[N],amp2[N],amp3[N];
10
11  void main(){
12
13    int i,j,k;
14    float loop,inc=0.1;
15    float low_interval=4.0;
16    float hig_interval;
17    float oct=12.0−low_interval;
18    float temp,temp1,temp2,temp3;
19    float x12,x23,v123;
20    float sum,t123,Ttotal;
21
22    printf("Interval¥tTension¥n");
23    for(loop=0.0; loop<=oct; loop=loop+inc){
24      hig_interval=loop;      /* hig_interval は loop 半音 */
25      freq1[0]=261.63;      /* 最低音の周波数（固定，Middle C） */
26      freq2[0]=pow(10,low_interval/39.863)*freq1[0];  /* 中間音は low_interval 半音離れた周波数 */
27      freq3[0]=pow(10,hig_interval/39.863)*freq2[0];  /* 最高音は hig_interval 半音離れた周波数 */
28
29      for(i=0;i<N;i++){
30        /* 倍音の周波数と音量の設定 */
31        freq1[i]=freq1[0]*(i+1);freq2[i]=freq2[0]*(i+1);freq3[i]=freq3[0]*(i+1);
32        amp1[i]=pow(0.88,i);amp2[i]=pow(0.88,i);amp3[i]=pow(0.88,i);
33      }
34
35      sum=0.0;
36      for(i=0;i<N;i++){
37        for(j=0;j<N;j++){
38          for(k=0;k<N;k++){
39            /* 周波数の値を昇順に並び替え */
40            temp1=freq1[i];temp2=freq2[j];temp3=freq3[k];
41            if(temp1>temp2){temp=temp1;temp1=temp2;temp2=temp;}
42            if(temp1>temp3){temp=temp1;temp1=temp3;temp3=temp;}
43            if(temp2>temp3){temp=temp2;temp2=temp3;temp3=temp;}
44            /* ３つの周波数から第１音程(x12)と第２音程(x23)の算出 */
45            x12=fabs(39.863*log10(temp2/temp1));
46            x23=fabs(39.863*log10(temp3/temp2));
47            v123=amp1[i]*amp2[j]*amp3[k];      /* 音量の設定関数 */
48            t123=v123*exp(−pow((x23−x12)/0.60,2));      /* 緊張度の計算式 */
49            sum=t123+sum;
50          }
51        }
52    }
53    Ttotal=sum/6;
54    printf("%4.1f¥t%6.3f¥n",loop,Ttotal);
55
56    }
57  }
```

図７　緊張度計算プログラム（C 言語）.

```
% t_curve.m: How to Draw Tension Curves
clear all; for n = 1:6; m = 3.0; for l = 0:0.01:12.0-m;
LF0 = 261.63; MF0 = 2^(m/12)*LF0; HF0 = 2^(l/12)*MF0;
for s = 1:n^3;
    i = round(mod(s-1,n)+1); j = round(mod((s-i)/n,n)+1); k = round((s-n*(j-1)-i)/(n^2)+1);
    freq(s,1) = LF0.*i; freq(s,2) = MF0.*j; freq(s,3) = HF0.*k; freq(s,:) = sort(freq(s,:));
    amp(s,1) = 0.88.^(i-1); amp(s,2) = 0.88.^(j-1); amp(s,3) = 0.88.^(k-1);
end;
DI = 12*log2((freq(:,2).^2).*(1./(freq(:,1).*freq(:,3))));
V = amp(:,1).*amp(:,2).*amp(:,3);
T = V.*exp(-((DI./0.6).^2));
hold on; plot(l,sum(T),'k')
end; end;
xlabel('2nd Interval(semitones)'); ylabel('Tension(T)'); grid on;
```

図8　緊張度計算プログラム（MATLAB）.

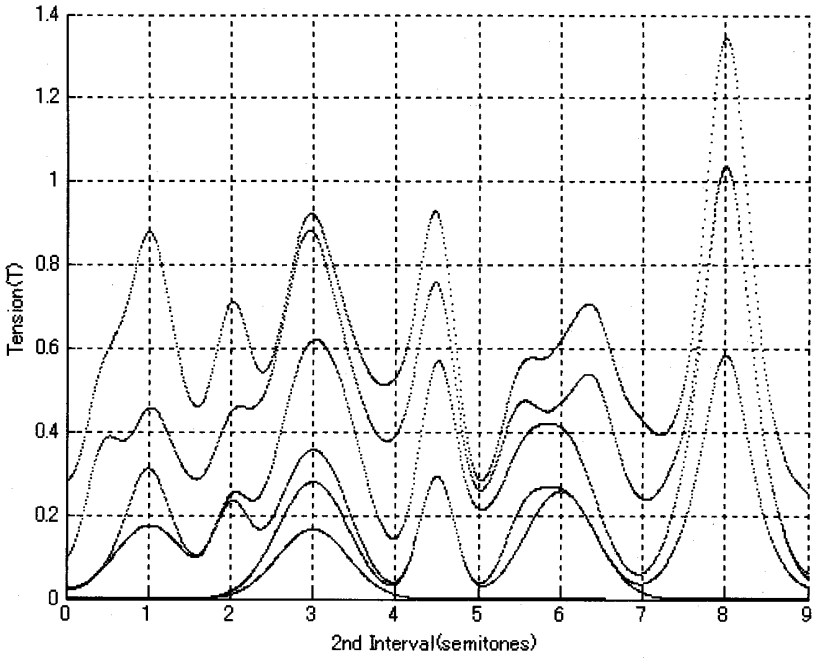


図9　緊張度計算プログラム（MATLAB）の実行結果例.

C#-E-G#となり，これは短調の和音の基本型であることが分かる．このときの第1音程と第2音程の値はそれぞれ3と4であり（3−4），その音程差は−1という値となる．緊張的和音と長短調の和音にみられるこの法則性を利用することで，明るさ／暗さ，楽しい感じ／寂しい感じというモダリティに関する定式化が可能となる．

まず，倍音成分を含まないモダリティ（$m$）の基本関数は，ある3音それぞれのピッチの基本周波数を$f_1$, $f_2$, $f_3$($f_1 < f_2 < f_3$)，またそれぞれの音量を$v_1$, $v_2$, $v_3$とした場合，以下のように定式化できる．

$$m = -v_{123}\left[\frac{2(x_{23}-x_{12})}{\varepsilon}\right]\exp\left\{-\left[\frac{-(x_{23}-x_{12})^4}{4}\right]\right\} \tag{8}$$

緊張度（$t$）の場合と同様に $v_{123}$ は $v_1$, $v_2$, $v_3$ の積である．また $e$ は定数で $e = 1.56$ である．3音の音量が等しい場合（$v_1 = v_2 = v_3$）のモダリティ（$m$）の理論曲線を図10に示す．

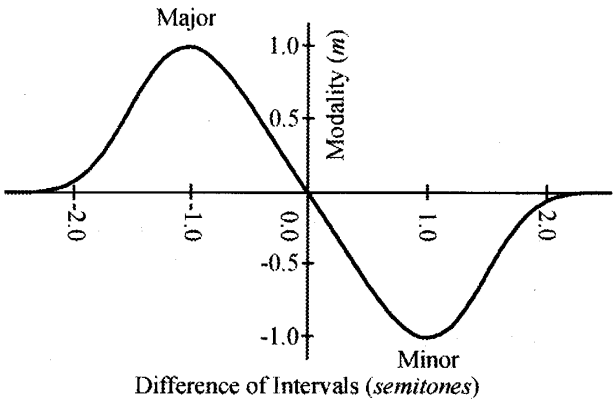


図10　モダリティ（$m$）の理論曲線．

また倍音の周波数が式(3)によって定義され，またそれぞれの音量が式(4)によって定義された場合，複合音三和音のモダリティ（$M$）の計算が可能となる．音1のi次周波数を $f_{1i}$，音2のj次周波数を $f_{2j}$，音3のk次周波数を $f_{3k}$ とし，またそれぞれの音量を $v_{1i}$, $v_{2j}$, $v_{3k}$ とした場合，そのモダリティ（$M$）は以下のように定式化される．

$$M = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\sum_{k=0}^{n-1} m(x_{ij}, x_{jk}, v_{ijk}) \tag{9}$$

ここで，$x_{ij}$ は周波数 $f_{1i}$ と $f_{2j}$ によって定義される音程（第1音程），$x_{jk}$ は $f_{2j}$ と $f_{3k}$ によって定義される音程（第2音程）である．また $v_{ijk}$ は音量 $v_{1i}$, $v_{2j}$, $v_{3k}$ の積である．

### 4.2　モダリティ計算プログラム

4.1節で述べたモダリティ（$M$）の計算方法とパラメータ設定によるプログラムを2種類以下に示す．モダリティの値は緊張度と同様に音程差を基にして算出されるので，その計算プログラムの構造は基本的に同じものである．したがって，本節のプログラムは緊張度計算プログラムと表示が異なる行についてのみ記載する．図11において，緊張度の計算プログラムと大きく異なる箇所は48行目にあるモダリティ計算式であり，あとはそれに伴って宣言された変数名（m123, Mtotal）が異なっている．パラメータeの値は同じく48行目に直接1.56と設定されているので，変更の場合にはそれぞれ書きかえる必要がある．また他のパラメータ（第1音程，倍音数，音程刻み）についても緊張度の場合と同様である．計算プログラムは，1つ目はC言語を用いたもので（図11）2つ目はMATLABを用いたものである（図12）．MATLABの

ものを実行すると$F_0$から$F_5$を含めたときのモダリティ曲線が描かれる（図13）.

```
01 /* m_curve.c: How to Draw Modality Curves */
:
20   float sum,m123,Mtotal;
21
22   printf("Interval¥tModality¥n");
:
48          m123=-v123*(2*(x2-x1)/1.56)*exp(-(pow(x2-x1,4))/4);    /* モダリティの計算式 */
49          sum=m123+sum;
50        }
51      }
52   }
53   Mtotal=sum/6;
54   printf("%4.1f¥t%6.3f¥n",loop,Mtotal);
```

図11　モダリティ計算プログラム（C言語）.

```
01 % m_curve.m: How to Draw Modality Curves
:
11 M = V.*((2.*DI./1.56).*exp((-(DI.^4)./4)));
12 hold on; plot(1,1/6*sum(M),'k')
:
14 xlabel('2nd Interval(semitones)'); ylabel('Modality(M)'); grid on;
```

図12　モダリティ計算プログラム（MATLAB）.

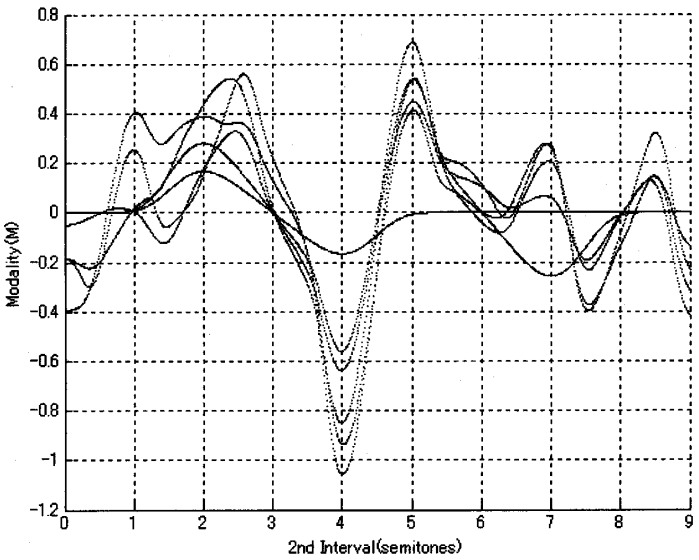


図13　モダリティ計算プログラム（MATLAB）の実行結果例.

## 5. 和音性の総合的評価

### 5.1　不安定度の定量化

不安定性は，ある3和音がどれほど“不安定”な印象で知覚されるかについての指標である.

不安定度（$I$）は不協和度（$D$）と緊張度（$T$）から定義される合成変数であり，以下の式によって定式化される．

$$I = \frac{1}{3}\sum_{i=1}^{3} D_i + \delta T \tag{10}$$

ここで，$\delta = 0.207$ であり，不安定度（$I$）に対する緊張度（$T$）の相対的影響度を決定する．パラメータ $\delta$ の値は主な3和音に対して得られた実験評価データ[13]に符合するように設定されたものである．

### 5.2　和音性の評価プログラム

最後に，これまで紹介してきたすべての和音性（不協和度・緊張度・不安定度・モダリティ）のそれぞれの値について計算するプログラム（C言語）を図14に示す．プログラムは，これまでの計算プログラムとは異なって理論曲線を描くタイプのようなものではなく，任意の三和音についてピッチ構造を指定すると，その三和音におけるそれぞれの和音性についての値を算出するというものである．ここでのプログラムは機能的にはこれまで解説してきたものとは異なるが，その構造については類似した部分が多いので解説については省略する．ただし，和音性の各種パラメータについての変数名は，プログラム内においても記述されているようにCook & Fujisawa (2006)[1]と対応させているので，その点については注意されたい．

プログラムを実行すると，任意の三和音についてのピッチ構造を特定するために，まずピッチの入力タイプ（周波数か音程か），次に周波数および音程の値，最後に音量スペクトルのタイプを選択する入力モードとなる（図15）．それぞれの値を入力しおわると，その入力パラメータの値に応じて，それぞれの和音性についての値が $F_0$ のみの場合から $F_5$ までを含めた場合まですべて算出される．

## 6. おわりに

本資料では，和音を定量的に評価するためのモデルとして“和音性”という統合的な概念を提案し，その心理物理モデルについての詳細と計算プログラムについての解説を行った．従来の西洋音楽理論において，和音のもつ豊かな印象については経験的には知られてきたし，また定性的には把握されてきたけれども，その体系的な理解が容易であったとは言い難い．本研究における“和音性”の知覚モデルはその理解のための基盤となりうるものである．本研究において提案された知覚モデルは定量的評価が可能であることから，心理学的な観点から見れば，脳科学を含めた心理物理データとの対応関係を明らかにすることも可能である．また工学的な観点から見れば，複合音の音響的特性とその印象についてのモデルという特性を生かすことで，楽音についての印象評価，または他の音響現象（例えば音声の音楽性評価[2],[3]）といった応用が期待できる．

```
01 /* Harmony.c: How to Calculate Harmoniousness */
02 /* The calculations of dissonance, tension, instability and modality */
03 /* described here are discussed in detail in an article in */
04 /* Empirical Musicology Review (2006) (http://emusicology.org). */
05
06 #include <stdio.h>
07 #include <math.h>
08 /* The constants used in Equations 1-7 are defined here: */
09 #define NP 6        /* Define the number of partials, F0-F5 */
10 #define a  0.60     /* Define the steepness of the tension curve, Eq. 1. */
11 #define b1 -0.80    /* Define the parameters in the dissonance curve, Eq. 3 */
12 #define b2 -1.60    /* Define the parameters in the dissonance curve, Eq. 3 */
13 #define b3 4.00     /* Define the parameters in the dissonance curve, Eq. 3 */
14 #define g 1.25      /* Define the exponent in the dissonance curve, Eq. 3 */
15 #define d 0.207     /* Define the relative weight of dissonance and tension, Eq. 5 */
16 #define e 1.558     /* Define the parameter in the modality curve, Eq. 6 */
17
18 float interval1, interval2;
19 float freq1[NP], freq2[NP], freq3[NP];
20 float amp1[NP], amp2[NP], amp3[NP];
21
22 void freq_setting(){
23
24   int i;
25   int pitch_switch;
26
27   printf("===== Select the Mode for Pitch Input: =====¥n");
28   printf("0. freqency(Hz)¥n");
29   printf("1. interval(semitones)¥n");
30   scanf("%d",&pitch_switch);
31   switch(pitch_switch){
32   case 0:
33     printf("For example, the major chord in root position would be: 261.63, 329.63, 392.00. ¥n");
34     printf("1st Freqency (Hz) = "); scanf("%f",&freq1[0]);
35     printf("2nd Freqency (Hz) = "); scanf("%f",&freq2[0]);
36     printf("3rd Freqency (Hz) = "); scanf("%f",&freq3[0]);
37
38     for(i=0;i<NP;i++){
39       freq1[i]=freq1[0]*(i+1); freq2[i]=freq2[0]*(i+1); freq3[i]=freq3[0]*(i+1);
40     }
41   break;
42   case 1:
43     printf("For example, the major chord in root position would be: 4, 3. ¥n");
44     printf("1st Interval (in semitones) = "); scanf("%f",&interval1);
45     printf("2nd Interval (in semitones) = "); scanf("%f",&interval2);
46
47     freq1[0]=261.63;          /* Middle C */
48     freq2[0]=pow(10,interval1/39.863)*freq1[0];
49     freq3[0]=pow(10,interval2/39.863)*freq2[0];
50
51     for(i=0;i<NP;i++){
52       freq1[i]=freq1[0]*(i+1); freq2[i]=freq2[0]*(i+1); freq3[i]=freq3[0]*(i+1);
53     }
54   break;
55   }
56 }
57
58 void amp_setting(){
59
```

図14　和音性計算プログラム（C言語）.

```
60    int i;
61    int amp_switch;
62
63    printf("===== Select a Configuration for the Relative Amplitudes of Partials: =====¥n");
64    printf("0. 1.00, 1.00, 1.00...(default)¥n");
65    printf("1. 1.00, 0.50, 0.33...(1/n)¥n");
66    printf("2. 1.00, 0.88, 0.77...(0.88^n; Sethares,1993,1999)¥n");
67    scanf("%d",&amp_switch);
68    switch(amp_switch){
69      case 0:
70        for(i=0;i<NP;i++){ amp1[i]=1.00; amp2[i]=1.00; amp3[i]=1.00; }
71        break;
72      case 1:
73        for(i=0;i<NP;i++){ amp1[i]=(float)1/(i+1); amp2[i]=(float)1/(i+1); amp3[i]=(float)1/(i+1); }
74        break;
75      case 2:
76        for(i=0;i<NP;i++){ amp1[i]=pow(0.88,i); amp2[i]=pow(0.88,i); amp3[i]=pow(0.88,i); }
77        break;
78    }
79  }
80
81  void calculate(){
82
83    int i,j,k,n;
84    float x12, x23 ,x13 ,v12 ,v23 ,v13 ,v123;
85    float temp, temp1, temp2, temp3;
86    float sum_dissonance, sum_tension, sum_instability, sum_modality;
87    float d12, d23, d13, d123, t123, m123;
88
89    printf("Fn Dissonance(D)      Tension(T)   Instability(I)   Modality(M)¥n");
90    for(n=0; n<NP; n++){
91      sum_dissonance=0.0; sum_tension=0.0; sum_instability=0.0; sum_modality=0.0;
92      for(i=0;i<n;i++){   for(j=0;j<n;j++){   for(k=0;k<n;k++){
93        temp1=freq1[i];temp2=freq2[j];temp3=freq3[k];
94        if(temp1>temp2){temp=temp1;temp1=temp2;temp2=temp;}
95        if(temp1>temp3){temp=temp1;temp1=temp3;temp3=temp;}
96        if(temp2>temp3){temp=temp2;temp2=temp3;temp3=temp;}
97        x12=fabs(39.863*log10(temp2/temp1));
98        x23=fabs(39.863*log10(temp3/temp2));
99        x13=fabs(39.863*log10(temp3/temp1));
100       v12=amp1[i]*amp2[j]; v23=amp2[j]*amp3[k]; v13=amp1[i]*amp3[k];
101       d12=v12*b3*(exp(b1*pow(x12,g))-exp(b2*pow(x12,g)));
102       d23=v23*b3*(exp(b1*pow(x23,g))-exp(b2*pow(x23,g)));
103       d13=v13*b3*(exp(b1*pow(x13,g))-exp(b2*pow(x13,g)));
104       d123=(d12+d23+d13)/3; sum_dissonance=sum_dissonance+d123;
105       v123=amp1[i]*amp2[j]*amp3[k];
106       t123=v123*exp(-((x23-x12)/a)*((x23-x12)/a)); sum_tension=sum_tension+t123;
107       sum_instability= sum_dissonance + d*sum_tension;
108       m123=v123*-(2*(x23-x12)/e)*exp(-pow((x23-x12),4)/4); sum_modality=sum_modality+m123;
109     }  }          }
110     printf("%2d%14.3f%14.3f%14.3f%14.3f¥n",n,sum_dissonance,sum_tension,sum_instability,sum_modality);
111   }
112 }
113
114 void main(){
115   freq_setting();
116   amp_setting();
117   calculate();
118 }
```

図14　和音性計算プログラム（C 言語）（続き）.

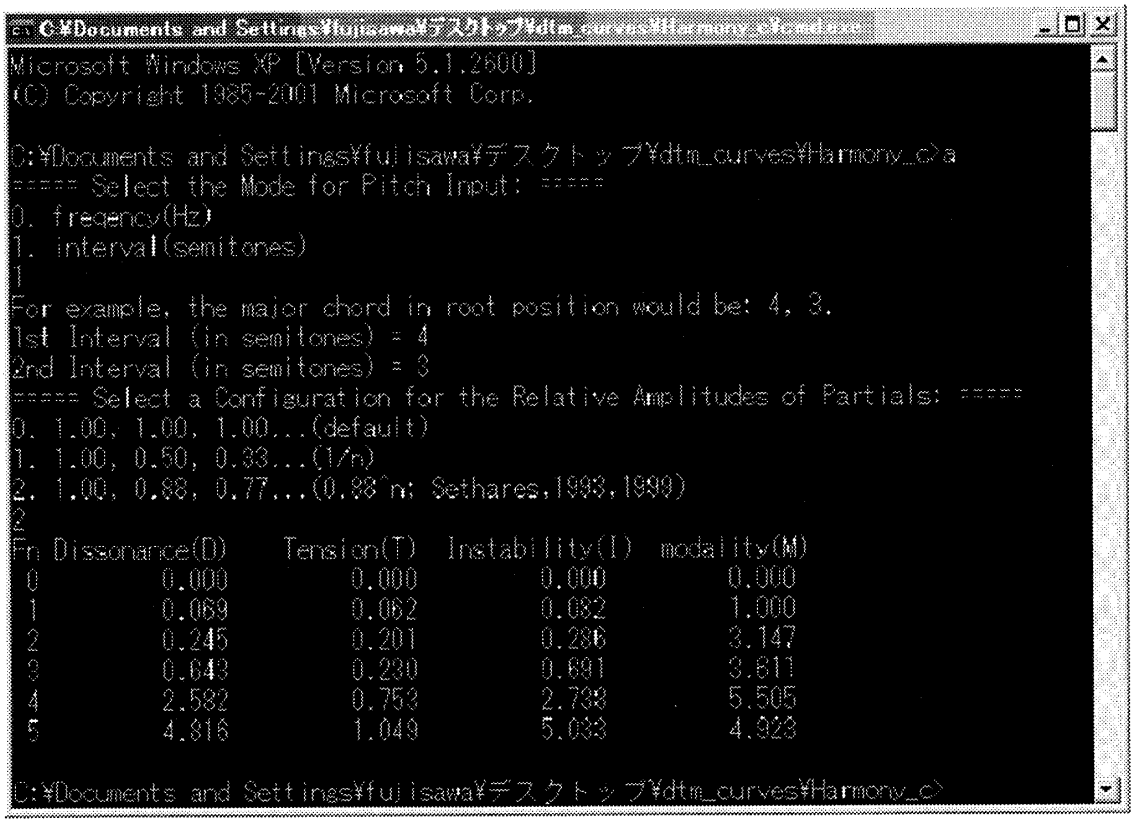


図15　和音性計算プログラムの実行結果例.

## 参考文献

[1] Cook, N. D. & Fujisawa, T. X. (2006) The Psychophysics of Harmony Perception: Harmony Is a Three-Tone Phenomenon. *Empirical Musicology Review*, 1 (2), 106-126. (http://emusicology.org/)

[2] 藤澤隆史・高見和彰・Cook, N. D. (2004) 感情的発話における音楽性：基本周波数を用いた和音性の定量化について. 認知心理学研究, 1 (1), 25-34.

[3] Cook, N. D., Fujisawa, T. X. & Takami, K. (2006) Evaluation of the Affective Valence of Normal Speech Using Pitch Substracture. *IEEE Transactions on Audio, Speech, and Language Processing*, 14 (1) pp. 142-151.

[4] Plomp, R., & Levelt, W. J. M. (1965) Tonal consonances and critical bandwidth. *Journal of the Acoustical Society of America*, 38, 548-560.

[5] Kameoka, A., & Kuriyagawa, M. (1969) Consonance theory: Parts I and II. *Journal of the Acoustical Society of America*, 45, 1451-1469.

[6] Sethares, W. A. (1993) Local consonance and the relationship between timbre and scale. *Journal of the Acoustical Society of America*, 94, 1218-1228.

[7] Sethares, W. A. (1999) *Tuning, timbre, spectrum, scale* (*2nd edition*). Berlin: Springer.

[8] Parncutt, R. (1989) *Harmony: A psychoacoustical approach.* Berlin: Springer.

[9] 上坂吉則 (2000) MATLAB プログラミング入門. 牧野書店.

[10] Cook, N. D. (2002) *Tone of Voice and Mind.* Amsterdam: John Benjamins.

[11] 藤澤隆史 (2004) 音声と音楽の感情表現について—和音性に関する音響心理学モデルの発話イントネーションへの応用—. 関西学院大学出版会 (関西大学大学院博士論文).

[12] Meyer, L. B. (1956) *Emotion and Meaning in Music.* Chicago: University of Chicago Press.

[13] Roberts, L. (1986). Consonant judgments of musical chords by musicians and untrained listeners. *Acustica*, 62, 163-171.