〔非公開〕

ＴＲ－Ｍ－００１４

音楽創作活動支援を目的とするフレーズへの自動キーワード付与手法の検討

照井　一由　Kazuyoshi TERUI
西本　一志　Kazushi NISHIMOTO

１９９７．２．２７

ＡＴＲ知能映像通信研究所

音楽創作活動支援を目的とするフレーズへの自動キーワード付与手法の検討

照井一由 * , 西本一志 **

* 豊橋技術科学大学　情報工学科　　**(株)ATR 知能映像通信研究所

概要

我々は現在音楽の創作活動を支援するための手法ならびにシステムの研究を進めている．音楽の創作活動においては，楽曲内のフレーズの関係を把握すること，そして他者と意思の疎通を十分にはかり演奏意図を的確に汲み取ることが重要となる．これは，自然言語によって対話を行なう際の状況と酷似している．我々はこれまでに自然言語による対話を対象とし，対話構造を統計的手法を用いて空間構造として提示することによって，各発言の相互関係の把握と相互の意思疎通の円滑化を支援する対話環境 AIDE(Augmented Infomative Discussino Environment) を開発している．そこで，このシステムを音楽情報に対して適用できるようにすれば，音楽創作活動を有効に支援することができるようになることが期待できる．この音楽創作活動支援用 AIDE の実現のためには，フレーズにキーワードを自動的に付与する技術を確立することが必要である．フレーズのキーワードとしては、フレーズの輪郭情報に基づくものとフレーズの色彩情報に基づくものの２つが考えられる．本稿では，フレーズをデジタル信号処理の手法によって処理することにより，輪郭情報に基づくフレーズへの自動キーワード付与を行なった．

1 はじめに

我々は現在音楽の創作活動を支援するための手法ならびにシステムの研究を進めている（例えば[2]）．この研究の一環として，本稿では，楽曲中におけるまとまり感のある旋律断片（以下これをフレーズと呼ぶ）に対して，そのフレーズの特徴を表現するための各種属性の自動抽出手法について述べる．なお，自然言語で記述されたあるテキストの特徴を表現するものとしてキーワードが使用されることとの類似性に基づき，本稿ではこのフレーズの特徴を表現する属性のことを便宜的にキーワードと呼ぶことにする．

音楽創作活動は，大きく作曲活動と演奏活動とに分類できる．また，その両者の中間にある行為として，ジャズなどにおける即興演奏が存在する．楽曲はフレーズの時系列的連鎖で構成されるのであるから，作曲活動はどのようなフレーズをどう時系列的に並べるかを考案することであり，また演奏活動はすでに並べられている既存のフレーズをどう演奏するかを考察し実行することであると言うことができる（即興演奏はこの二つを同時に行なうものである）．したがってこれらの創作活動においては，個々のフレーズが相互にどういう関係にあるかを考慮することが重要となると考えられる．

一方音楽創作活動を別の視点から見ると，複数の創作者によるコラボレーションとしての側面があることがわかる．これは特に演奏活動において顕著である．たとえばオーケストラの演奏においては，指揮者と各パートの奏者あるいは各奏者同士が，互いの演奏を聞きあるいは身振りや表情などを互いに見あうことによって，全体として一貫した演奏表現をすることができるように協調的動作を行なっている．さらに，ジャズの即興演奏，特に"4 bars"[1]と呼ばれる形態の即興演奏では，オーケストラの場合と同様，相互の意思疎通によって演奏するフレーズへ表情づけを行なうのに加え，他者の演奏したフレーズや伴奏のフレーズとの関係を考慮しながら自分が演奏するフレーズを生成（作曲）していくことも必要となる．

このように音楽創作活動においては，楽曲内のフレーズの関係を把握すること，そして他者と意思の疎通を十分にはかり演奏意図を的確に汲み取ることが重要となる．この状況は一般的な自然言語による対話における状況と酷似している．すなわち対話を進めるにあたって，各対話参加者は対話中の各発言相互の関係を把握し，相互に意思疎通を十分にはかりながらこれらの情報に基づき自分の次の発言を構成していかねばならないのと同じ状況であるとみなせる．

我々はこれまでに自然言語による対話を対象とし，対話構造を統計的手法を用いて空間構造として提示することによって，各発言の相互関係の把握と相互の意思疎通の円滑化を支援する対話環境AIDE(Augmented Infomative Discussino Environment)[1]を開発している．そこで，このシステムを音楽情報に対して適用できるようにすれば，音楽創作活動を有効に支援することができるようになることが期待できる．すなわち，

- 複数演奏者による音楽演奏におけるインタラクション支援：

  通常使用される，音，身ぶり，表情などのモーダルに，さらにもう一つの人工的なモーダルを追加することにより，意思疎通を円滑化できる．

- 即興演奏におけるフレーズ生成支援：

  即興演奏という実時間的作曲行為を支援する．アドリブ・ソロ演奏，あるいは複数演奏者による4 bars形式などでの即興演奏において，演奏者に対しフレーズ生成のためのある指針を与える．

[1]複数の演奏者がそれぞれ4小節の短いアドリブ・ソロを順番に演奏していく形態の即興演奏

- 楽曲分析支援：

  例えば作曲にあたり，楽曲を構成するフレーズ相互の関係を分析することによって楽曲全体がどういう構造となるかを把握できるようになる．またたとえば即興演奏の学習において，プロ奏者のアドリブソロの構造を分析することにより，どのようなフレーズの連鎖が良い結果を産むのかを調べることを手助けできる．

などの応用が考えられる．

AIDE では個々の発言オブジェクトがどのようなキーワードを共有しているか，およびある一つの発言オブジェクトの中でどのようなキーワードが共起しているかの情報に基づき，各発言オブジェクト間の関係を定量化している．したがって，音楽においてこの発言オブジェクトにあたるものが個々のフレーズであるとみなすならば，音楽創作活動支援用 AIDE （以下 Music-AIDE と呼ぶ）の実現のためには，フレーズにキーワードを自動的に付与する技術を確立することが必要である．

音楽のフレーズには，自然言語における単語のような明確な意味を持つ単位が存在しない．音楽の最小単位としての一つ一つの音は自然言語におけるアルファベットに相当するものであるゆえ，それらを個々に取り出しても意味のある属性として扱うことはできない．したがって，個々の音よりは大きな単位で，対象フレーズの特徴を表現することができる程度の情報を抽出することが必要である．このようなフレーズを特徴づける属性としては，「フレーズの輪郭情報」と「フレーズの色彩情報」の 2 種が考えられる．「フレーズの輪郭情報」とはフレーズを構成する音のたどる軌跡に注目した場合の情報であり，「フレーズの色彩情報」とは，フレーズを構成する音の持つ機能に注目した場合の情報である．本稿ではこのうちフレーズの輪郭情報に属する属性情報の抽出によるキーワード付与手法について検討する．

2　システムの構成

図 1に Music-AIDE の全体構成を示す. 機器構成は, 一台の入力用インタフェースと,SGI の Indy グラフィックワークステーション上で稼働する Music-AIDE ソフトウエアで構成され, これらが MIDI で統合されている. Music-AIDE には,MIDI を通じて入力用インタフェースからのタイミングクロック, ノート情報を主とする各種 MIDI 情報が入力される. 演奏中 Music-AIDE は, 入力用インタフェースからのクロック情報に同期して処理を進める. 今回の実習では, 入力用インタフェースの代わりにシーケンサを使用し, あらかじめプログラムしたフレーズを使用して実験を行った.

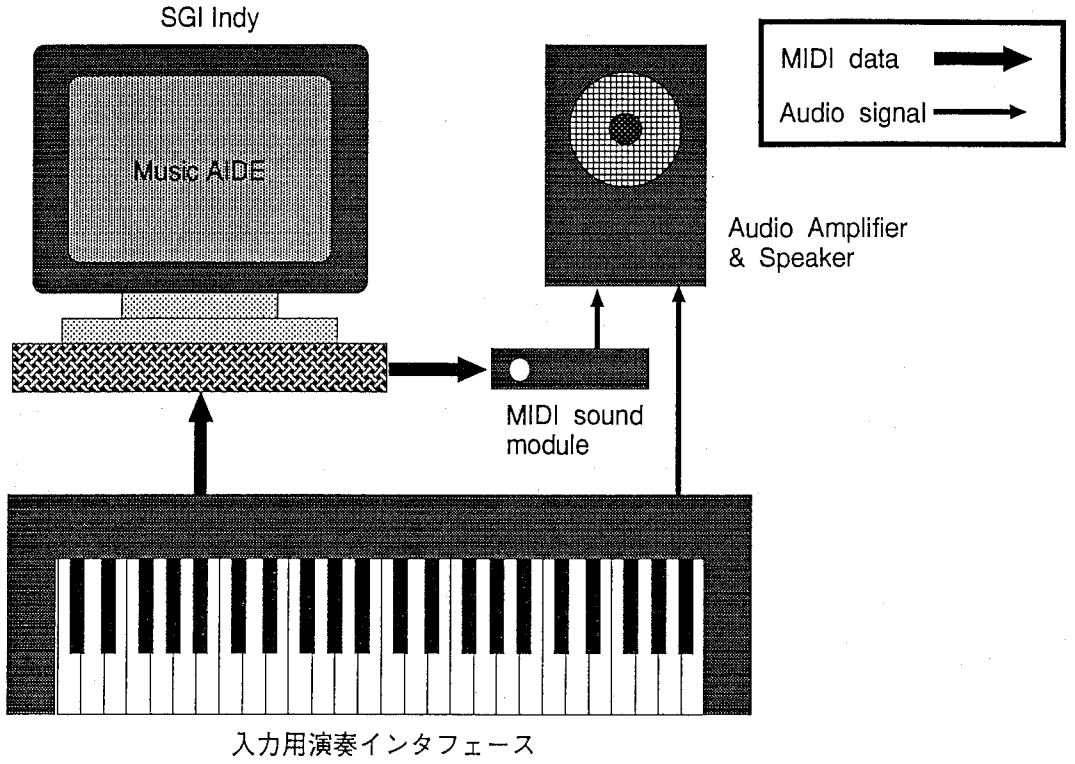


図 1: システムの構成

3　キーワード付与方法

MIDI で入力される演奏データは, 音の Pitch とその長さ（拍数である）. これは, 楽譜として記述されている楽曲のデータと等価であるので, 以下では, 楽譜データを用いて説明を進める.

また, 本来フレーズとは特に定まった長さを持つものではなく, その音楽的内容によって, どこからどこまでがひとまとまりのフレーズになるかを判断すべきである. しかし, 機械的にこの判断を行うことは困難であるため, 今回は,4 小節をひとまとまりのフレーズと見なし処理を行うことにした.

本システムで, 音の輪郭情報に対して, キーワードを与えるために, ディジタル信号処理の技術を導入した. 楽譜を追っていくと, その音符の動きが, 信号処理で言う不等間隔サンプ

リングデータに類似していることが分かる. 今回, これをディジタル処理するために, 補間を用いて等間隔サンプリングデータを生成し,

- 平均, 標準偏差
- 自己相関関数
- スペクトル解析

を求めることにより, 得られたデータを「フレーズの輪郭情報に基くインデックス情報」とした.

また, 信号処理的な手法ではないが, 対象となるフレーズの音符の数もキーワードとした.

3.1 平均, 標準偏差

対象となるフレーズを楽譜のような離散データと考え, 平均, 標準偏差を求める.N 個のサンプリング系列 x_p(p=0,1,2, $\cdots$,N-1) の平均 $\bar{x}$, 標準偏差 σ_x は次式により求まる.

$$\bar{x} = \frac{1}{N}\sum_{p=0}^{N-1} x_p \tag{1}$$

$$\sigma_x = \sqrt{\frac{1}{N}\sum_{p=0}^{N-1}(x_p - \bar{x})^2} \tag{2}$$

この時, 各音符の持つ, 時間的な長さは一定ではないため, サンプリングは不等間隔となる. フレーズの平均及び標準偏差を求めるには, このデータ列を補間し, 等間隔データを生成しなければならない. 補間の方法はいくつか考えられるが, 今回は音が次の音へと変化する間, 同一の音であるとして補間を行った. この補間方法は実際の音の変化と等価なものであると考えられる.

このような方法で生成された等間隔データから平均を求めることによって, 対象となるフレーズの全体のキーについての情報が得られる. また標準偏差を求めることによって, 音のばらつきについての情報が得られる.

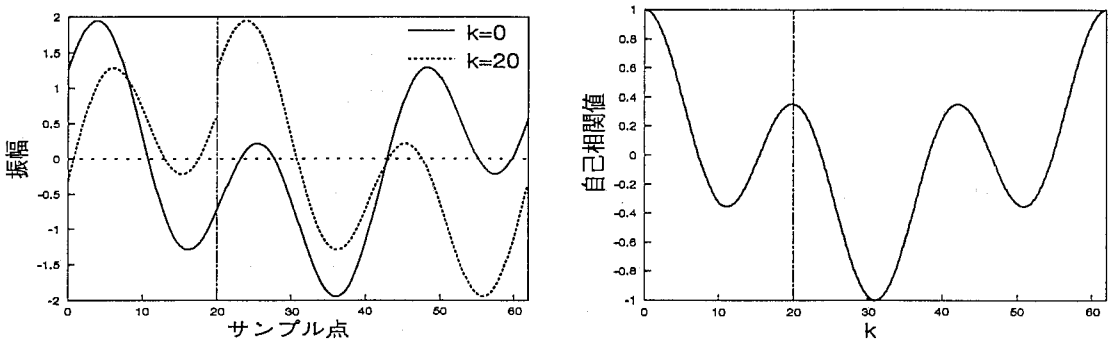
3.2 自己相関関数

対象となるフレーズの自己相関関数を求め, フレーズの構成についての情報をえる.N 個のサンプリング系列 x_p(p=0,1,2, $\cdots$,N-1) の自己相関関数 r_{xx} は以下の式により定義される.

$$r_{xx} = \frac{1}{N\sigma_x^2}\sum_{p=0}^{N-1}(x_p - \bar{x})(x_{p+k} - \bar{x}) \qquad p+k \geq N \ \ then \ \ x_{p+k} = x_{p+k-N} \tag{3}$$

$\bar{x}$ は x_p の平均値で σ_x はその標準偏差である. 等間隔データの生成は 3.1節と同様の方法で行った. 図 2に図示したのは, ある対象波形の例と, その波形についての相関関数の変化である. この対象波形は 4 小節分のフレーズを補間し, サンプリング数 N =64 の等間隔サンプリングデータを生成したものである. 図 2の (b) の変化に注目すると,$k = 0$ の時点で最大であった値が,k の増加と共に徐々に減少する. 再び値が増加するのは $k = 12$ を過ぎた点からで,$k = 20$ で極大をとった後, 再び減少を始める. ここで図 2の (a) をみると $k = 20$ の点で $k = 0$ の波形とその形が類似していることが分る. この時の $k = 20$ は,4 分 4 拍子では 5 拍

に相当する. つまりこの対象波形は,5 拍めで反復されていることになる. このように, 相関関数のピークを検出することによってフレーズがどの程度のスパンで繰り返しパターンを含んでいるかを読み取ることが可能である. 本稿では, フレーズの繰返しパターンの情報を得るキーワードとして, このようなフレーズの自己相関関数のピークにおける, 時間的なずれ幅をキーワードとした. また, ピークが相関関数に複数含まれている場合には, ピークでの相関値の大きいものをキーワードとした.



(a) 対象波形　　(b) 自己相関関数

図 2: 自己相関関数の増減

3.3 スペクトル解析

フレーズをフーリエ解析することにより, フレーズ波形の動きの情報を得ることができる. フーリエ解析をする際には, スプライン補間によって, 等間隔なサンプリング系列を得る. またあらかじめ, このサンプリング系列を, 平均, 標準偏差を用いて正規化することによって, フーリエ解析した場合の, スペクトルの総量を, フレーズ間で統一した. このようにして得られたデータに対してフーリエ変換を行い, パワースペクトルを算出する. スペクトルを解析し, キーワードを得るために, 図 3のようにパワースペクトルを周波数に応じて, 複数の成分に分け, その各成分それぞれの総量をキーワードとした.

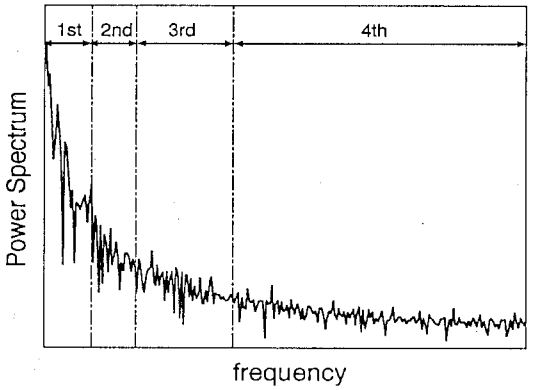


図 3: フレーズのスペクトル解析

各成分の総量を, フレーズ間で比較, 検討することにより, 大きな動きのフレーズか, 細かな動きのフレーズかといったフレーズの動きの特徴を検出する.

3.4 音符の数による解析

フレーズ中に含まれる音符の数がキーワードとなる. これは, フレーズ中で使用される音の長さの, 代表値の逆数と見なせる. これにより, そのフレーズがどのようなテンポで進行しているかを知ることが可能である.

4 キーワード付与機能の実装

この章では本稿の末巻に記載されたプログラムリストを参照しながら説明を進める. また, 今回解析の解析では、音楽データベースを 4 小節に区切り, 各 4 小節に対し解析を行った.

本システムは, 次に示す 7 つのプログラムにより構成される. 各ソースの概要を表 1 に示す.

ソースプログラム	概要
analysys.c	フレーズ解析を行う
initMidi.c	MIDI の初期化を行う
Midi.c	フレーズの採取を行う
FFT.c	高速フーリエ変換を行う
correlation.c	自己相関関数を求める
spl.c	スプライン補間を行う
step.c	階段状補間を行う

表 1: プログラム概要

このシステムは, フレーズ採取プロセスとフレーズ解析プロセスとで構成されている. 処理の概要を図 4に示す. フレーズ採取プロセスは, 入力用演奏インタフェースより入力された MIDI 情報から, ノートナンバーと, 演奏開始点からの相対的な時間情報を, 演奏データとしてファイルに出力する. 出力用のファイルは 2 つ用意し, それぞれのファイルを 4 小節ごとに交互に使用する. フレーズ解析プロセスでは, 採取プロセスによってファイルに書き込まれた演奏データを読み込み, 処理を進める. この時, フレーズ解析プロセスが, 採取プロセスによって書き込み中のファイルにアクセスするのを防ぐために, プロセス間では, セマフォを用い排他的処理を行っている. この様子を示したのが図 5である. 親プロセスである解析プロセスは, 初め, 自らのプロセスをロックする. 子プロセスである採取プロセスは,MIDI 情報を 4 小節取り込んだ時点で解析プロセスをアンロックし, 同時に次の 4 小節からの演奏データの収集を開始して、用意されたもう一方のファイルに対し出力する. アンロックされた解析プロセスは, 採取プロセスによって得られた, フレーズ輪郭情報の入ったファイルにアクセスし, 解析を始める. 解析が終了した時点で, このプロセスは, 再び自らをロックする. このような動作を連続的に行うことによって, リアルタイムでの処理が可能となる.

解析された情報は Tck/TK を利用し, ディスプレ以上にグラフとして, リアルタイムに出力される.

4.1 analysys.c

'analysys.c' は, 本システムのメインプログラムである. このプログラムの流れを図 6のフローチャートに示す.

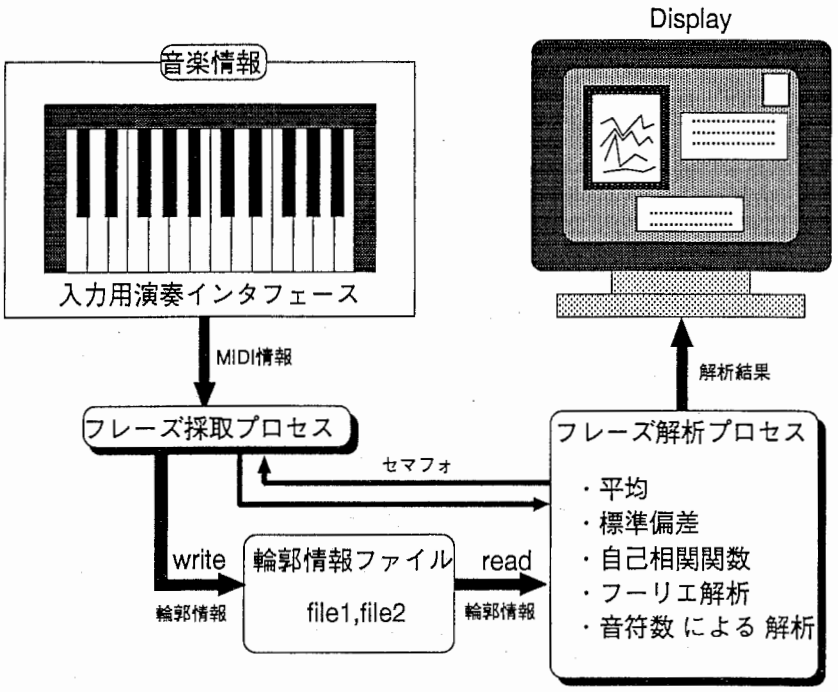


図 4: プロセス間の情報の伝達

フレーズ解析プロセス (親プロセス)		フレーズ採取プロセス (子プロセス)	
プロセスの 動作	使用ファイル	プロセスの 動作	使用ファイル
lock wait ↓		get MIDI infomation ↓ 4verse finish unlock	file1
analysys ↓ lock	file1	get MIDI infomation ↓ 4verse finish unlock	file2
wait			
⋮	⋮	⋮	⋮
analysys ↓ lock	file1	get MIDI infomation ↓ 4verse finish unlock	file2
wait			
analysys ↓ lock	file2	exit	
wait			

図 5: セマフォによる排他的処理

各解析を行う場合, 採取プロセスからのセマフォ値を確認し, ロックされていなければ, 平均, 標準偏差, 自己相関関数, フーリエ解析を実行する. 解析が終了すると, 自プロセスをロックし, フレーズ採取プロセスがアンロックするまでプロセスを停止する.

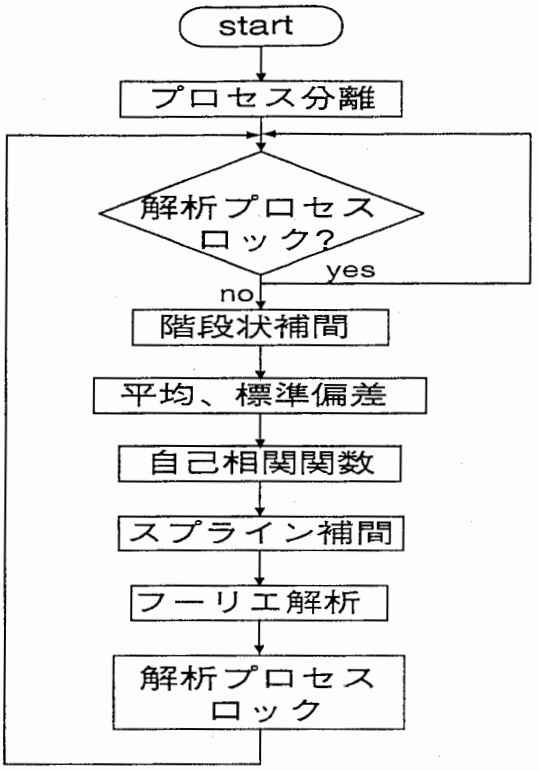


図 6: 'analysys.c' のフローチャート

4.2 Midi.c

'Midi.c' はフレーズの採取プログラムである.analysys.c によって分割されたフレーズ採取プロセスは, このプログラム上で動作する. このプログラムの流れを図7のフローチャートに示す.

このプログラムは, 入力用演奏インタフェースから MIDI を通じて, タイミングクロック, ノート情報を主とする, 各種 MIDI 情報を採取する. MIDI 情報はプログラムにより, タイミングクロック, ノート情報, 演奏終了情報のいずれであるかを判定され, それに対応した処理を行う. タイミングクロックの場合, クロックカウンタをカウントし, もし4小節の処理が終了していれば, 解析プロセスをアンロックする. ノート情報の場合, ノートナンバーと, 相対的な時間情報をファイルに出力する. 終了情報の場合, セマフォを消去しプロセスを終了する.

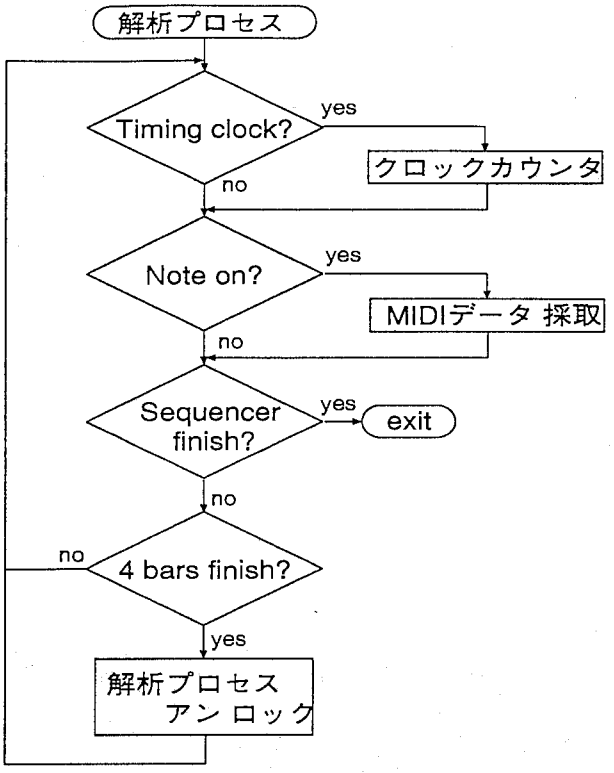


図 7: 'Midi.c' のフローチャート

4.3 FFT.c

'FFT.c' は高速フーリエ変換を行い, スペクトル解析をするプログラムである. フーリエ変換するためのサンプリング数 N は, ナイキストのサンプリング定理[2]を十分考慮し $N = 64$ とした. フーリエ解析により得られたパワースペクトルは,3.3節で述べたように, 周波数に応じ 4 つの成分に分けられる. 図 3に示す'3rd' と'4th' とを, 分割する境界は, 図 8に示すように,16 分音符の動きを示す 16 番目のスペクトルとした. 同様に'2nd' と'3rd' とを分割する境界は,8 分音符の動きを示す 8 番目のスペクトル, '1st' と'2nd' とを, 分割する境界は,4 分音符の動きを示す 4 番目のスペクトルとした.

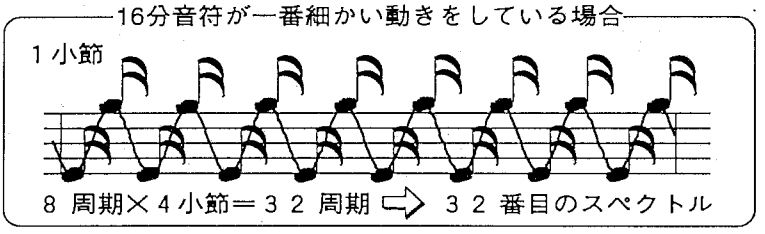


図 8: 音符の動きと周波数の関係

[2]ナイキストのサンプリング定理とはサンプリング周波数 f_s と最大周波数 f_m との関係が

$$f_s \geq 2f_m$$

を満たさなければならないというものである. この定理が満たされない場合, スペクトルの高周波成分に誤差を生ずる. またフーリエ解析によって得られたスペクトルは N/2 を境界とし対象性を有する. これにより, 今回解析したスペクトルは N/2 番目までである.

5　出力結果

各キーワードをリアルタイム出力したものを, 図 9に示す. 今回解析に使用した音楽データベースは,SONNY STITT による「ALL THE THINGS YOU ARE」のアドリブソロである.

出力は, 平均, 標準偏差, 自己相関値, 音符の数について行った. 各キーワードは,

- 平均　　: 三角形
- 標準偏差　　: 円
- 自己相関関数　: 菱形
- 音符の数　　: 四角形

でプロットされている. このグラフの横軸は, 各フレーズの順番で,4 小節ごとに配置されている. 縦軸は各キーワードの代表値である.

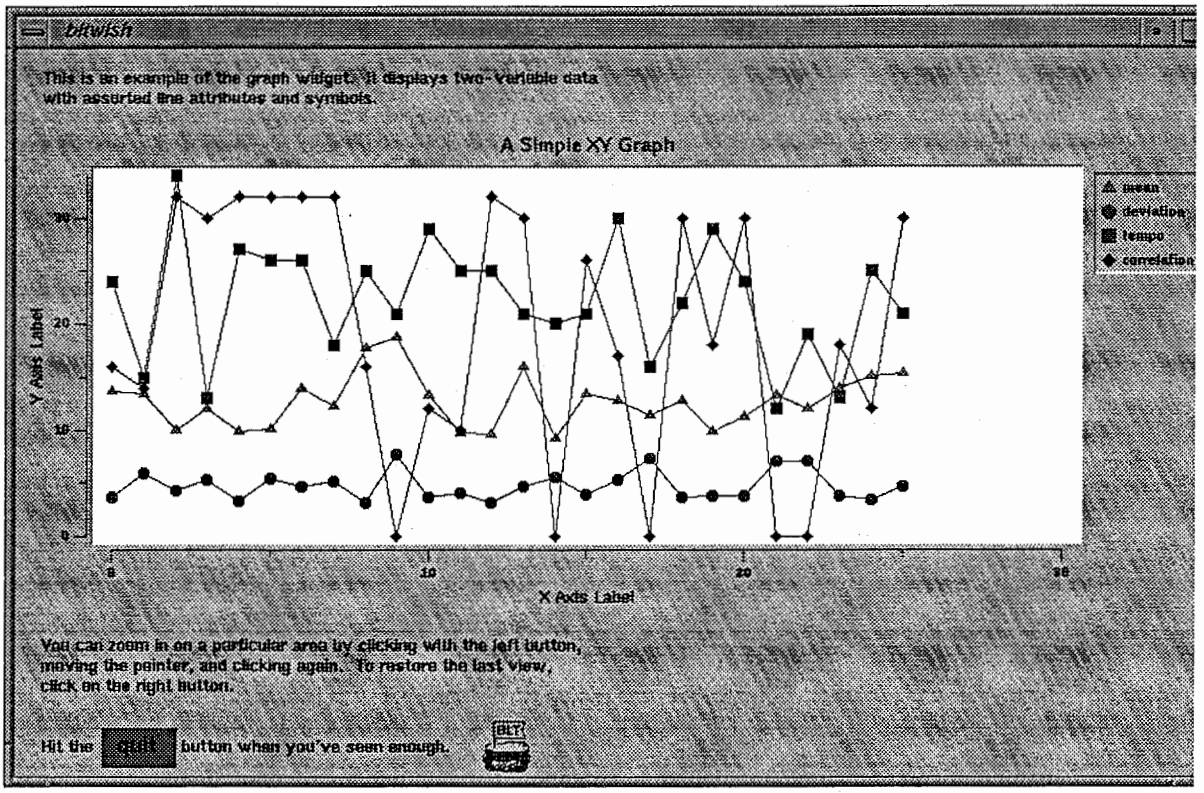


図 9: 各キーワードの出力結果

6　まとめ

我々は現在, 音楽の創作活動を支援することを目的とし, AIDE の枠組で音楽を取り扱うことを目指している. その実現のためにはフレーズの特徴を表現するキーワードの付与方法を確立する必要がある. 演奏者が奏でたフレーズをキーワード化する手段としては,「フレーズの輪郭情報に基づくインデックス抽出」と「フレーズの色彩情報に基づくインデックス抽出」の 2 つが考えられる. 本稿では, 前者について信号処理の技法を用いて, 検討をおこ

なった. この結果, 平均, 標準偏差, スペクトル解析, 自己相関関数を求めることにより, 4小節単位でほぼリアルタイムにキーワード獲得が可能となった. 今後は「フレーズの色彩情報に基ずくインデックス抽出」についてもキーワード付けを行い,Music-AIDE を実装し試用実験をすすめたい.

謝辞

実務訓練を行うにあたり, 多大なご指導いただいた知能映像研究所 第二研究室の方々に心から感謝いたします. また, 充実した実務訓練の場を与えて下さった, 知能映像通信研究所の方々にこの場を借りて厚く御礼を申し上げます.

参考文献

[1] 西本, 角, 間瀬：”一般参加者として対話に加わる対話活性化エージェント”, 信学技報,TL96-7(1996-11)

[2] 西本, 渡辺, 馬田, 間瀬：”旋律を奏でられる即興演奏用リズム楽器の提案”, 情処研報, 音楽情報科学,18-5(1996-12)

[3] 宮川, 城戸ほか著：”ディジダル信号処理”, コロナ社, 1975

```
#!../bltwish -f

source bltDemo.tcl

image create photo bgTexture -file ./bitmaps/rain.gif

option add *Graph.Tile                  bgTexture
option add *Label.Tile                  bgTexture
option add *Frame.Tile                  bgTexture
option add *Htext.Tile                  bgTexture
option add *TileOffset                  0
option add *HighlightThickness          0
option add *takeFocus                   1

option add *graph.elemActiveColor       yellow4
option add *graph.elemActiveFill        yellow
option add *graph.elemPixels            6
option add *graph.elemScaleSymbols      1

set visual [winfo screenvisual .]
if { $visual != "staticgray" } {
    option add *print.background yellow
    option add *quit.background red
}

proc FormatLabel { w value } {
    puts stderr "tick is $value"
    return $value
}


frame .f
set remote {}
set graph .graph

htext .header -text {\

This is an example of the graph widget.  It displays two-variable data
with assorted line attributes and symbols.

}

graph $graph -title "A Simple XY Graph"
$graph xaxis configure \
        -loose 1 \
        -title "X Axis Label"
$graph yaxis configure \
        -title "Y Axis Label"
$graph legend configure \
        -activerelief sunken \
        -background ""

htext .footer -text {\

You can zoom in on a particular area by clicking with the left button,
moving the pointer, and clicking again.  To restore the last view,
click on the right button.

Hit the %%
button $htext(widget).quit -text quit -command {
    catch "send GraphConfig after 1 exit"
    exit
}
$htext(widget) append $htext(widget).quit
%% button when you've seen enough.%%
label $htext(widget).logo -bitmap BLT
```



```
$htext(widget) append $htext(widget).logo -padx 20
%%}


for {set i 0} 1 {incr i} {

        update

        if { ![file readable /home/terui/study2/data/count.dat] } {
                continue
        }
        set file_x  [open /home/terui/study2/data/count.dat r]
        while {![eof $file_x]} {
        gets $file_x x
        }
        close $file_x


        if { ![file readable /home/terui/study2/data/mean.dat] } {
                continue
        }
        set file_y1 [open /home/terui/study2/data/mean.dat r]
        while { ![eof $file_y1] } {
        gets $file_y1 y1
        }
        close $file_y1


        if { ![file readable /home/terui/study2/data/deviation.dat] } {
                continue
        }
        set file_y2 [open /home/terui/study2/data/deviation.dat r]
        while { ![eof $file_y2] } {
        gets $file_y2 y2
        }
        close $file_y2


        if { ![file readable /home/terui/study2/data/tempo.dat] } {
                continue
        }
        set file_y3 [open /home/terui/study2/data/tempo.dat r]
        while { ![eof $file_y3] } {
        gets $file_y3 y3
        }
        close $file_y3

        if { ![file readable /home/terui/study2/data/correlation_s.dat] } {
                continue
        }
        set file_y4 [open /home/terui/study2/data/correlation_s.dat r]
        while { ![eof $file_y4] } {
        gets $file_y4 y4
        }
        close $file_y4

        if {$i==0} {
                vector X Y1 Y2 Y3 Y4
                X  set  " $x "
                Y1 set  " $y1 "
                Y2 set  " $y2 "
                Y3 set  " $y3 "
                Y4 set  " $y4 "

                $graph element create mean -xdata X -ydata Y1 \
                -symbol triangle -color green4 -fill green1
```

```
                $graph element create deviation -xdata X -ydata Y2 \
                -symbol circle -color red4 -fill red1
                $graph element create tempo -xdata X -ydata Y3 \
                -symbol square -color purple4 -fill purple1
                $graph element create correlation -xdata X -ydata Y4 \
                -symbol diamond -color blue4 -fill blue1

        } elseif { $tmp_y2 != $y2 } {
                unset X
                unset Y1
                unset Y2
                unset Y3
                unset Y4

                vector X Y1 Y3 Y2 Y4

                X  set  " $x "
                Y1 set  " $y1 "
                Y2 set  " $y2 "
                Y3 set  " $y3 "
                Y4 set  " $y4 "

                $graph element  configure mean -xdata X -ydata Y1 \
                -symbol circle -color red4 -fill red1
                $graph element configure  deviation -xdata X -ydata Y2 \
                -symbol triangle -color green4 -fill green1
                $graph element configure tempo -xdata X -ydata Y3 \
                -symbol square -color purple4 -fill purple1
                $graph element configure correlation -xdata X -ydata Y4 \
                -symbol diamond -color blue4 -fill blue1

        } else {
                continue
                set i 1
        }

        set tmp_x  $x
        set tmp_y1 $y1
        set tmp_y2 $y2
        set tmp_y3 $y3
        set tmp_y4 $y4

        table .f \
         .header 0,0 -padx 20  \
         .graph 1,0  \
         .footer 2,0 -padx 20

        table configure .f .header .graph .footer -fill both

        table . .f -fill both
        wm min . 0 0

        Blt_ZoomStack $graph
        Blt_Crosshairs $graph
        Blt_ActiveLegend $graph
        Blt_ClosestPoint $graph

        puts stdout "."
        set i 1
}
```

```
#define CMAX        2050            /* soundの配列の最大値                          */
#define Mine_Time   6.0             /* 1/4拍 (1拍:24)                               */
#define Mine_Time   6.0             /* 1/4拍 (1拍:24)                               */
#define BAR_4       4               /* 何小節読み込むか                             */
#define Max_Time    4.0*4.0*24.0    /* " BAR_4" 小節でのMidiのタイミングクロック数 */
#define SAMPLING    64              /* 補間時のサンプリング数                       */
#define FILTER_ODER 10              /* ローパスフィルターの次数                     */
#define PI          3.141592654     /* 円周率                                       */

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

/* セフォマ */
#include <sys/ipc.h>
#include <sys/sem.h>

/* Midi */
#include <dmedia/midi.h>
#include <getopt.h>
#include <sys/syssgi.h>

#include <unistd.h>
#include <sys/types.h>
#include <bstring.h>
#include <sys/time.h>

#include <limits.h>
#include <sys/prctl.h>
#include <sys/schedctl.h>


/* 構造体の定義 */
typedef struct{                 /* フレーズ用構造体 */
  double      amp[CMAX];
  double     time[CMAX];
  int        lastNum   ;
} phrase_def;

union {                         /* semctlの引数*/
  int val;
  struct semid_ds *buf;
  ushort *array;
} semnum;


#include "signal_prosess.h"     /* 信号処理演算用ヘッダ */
#include "liner.h"              /* 線形補間用ヘッダ */
#include "step.h"               /* 階段状補間用ヘッダ */
#include "nrutil.h"
#include "nrutil2.h"
#include "spline.h"
#include "splint.h"
#include "spl.h"                /* スプライン補間用ヘッダ */
#include "FFT.h"                /* 高速フーリエ変換用ヘッダ */
#include "correlation.h"        /* 自己相関関数用ヘッダ */
#include "Midi.h"               /* フレーズ採取用ヘッダ */


main(){

  int         i, k, count=0;        /* カウンタ */
  int         N;                    /* 補間サンプリング点数 (相関関数) */
  int         fileFlag=0;           /* fileFlag->0:ampl.dat ,fileFlag->1:amp2.dat*/
```



```
  int         take;                 /* */
  int         stat;                 /* 子プロセスの終了状態*/
  int         semid;                /* セフォマID */
  double      sigma, u;             /* 標準偏差、平均 */
  double      re[CMAX], im[CMAX];
  extern int  errno;
  phrase_def  phrase;               /* フレーズの構造体 */
  FILE *fp;                         /* */
  FILE *fp_x;                       /* */
  FILE *fp_m;                       /* 平均出力ファイル*/
  FILE *fp_h;                       /* 標準偏差出力ファイル*/

  /* ファイルの初期化 */
  if((fp_x = fopen("data/count.dat","w")) == NULL){
    printf("can't open file scores count.dat\n"); exit(0);}
  fclose(fp_x);
  if((fp_m = fopen("data/mean.dat","w")) == NULL){
    printf("can't open file scores mean.dat\n"); exit(0);}
  fclose(fp_m);
  if((fp_h = fopen("data/deviation.dat","w")) == NULL){
    printf("can't open file scores deviation.dat\n"); exit(0);}
  fclose(fp_h);
  if ( ( fp = fopen("data/ampl.dat","w") ) == NULL ){
    printf("can't open file scores ampl.dat\n");exit(0);}
  fclose(fp);
  if ( ( fp = fopen("data/amp2.dat","w") ) == NULL ){
    printf("can't open file scores ampl.dat\n");exit(0);}
  fclose(fp);
  if((fp = fopen("data/correlation_s.dat","w")) == NULL){
    printf("can't open file scores correlation_s.dat\n"); exit(0);}
  fclose(fp);

  /* Get Sempho ID */
  if( ( semid = semget( IPC_PRIVATE, 2, 0666 ) )==EOF){
    printf("semGet Error %d\n",errno);
    exit(1);
  }

  /* Set Semaph Value => 0 */
  semnum.val = 0;
  if( semctl( semid, 0, SETVAL, semnum )==EOF ){
    printf("SemCtl Error %d\n");
    exit(1);
  }
  if( semctl( semid, 0, SETVAL, semnum )==EOF ){
    printf("SemCtl Error %d\n");
    exit(1);
  }

 /*** 音楽波形の採取プロセス ***/
  if( fork()==0 ){
    Midi( semid );
    exit(0);
  }


  /*** 音楽波形解析プロセス ***/
  if( fork()==0 ){
    if( fileFlag==0 ){                  /* 採取プロセスとの同期 */
      semlock( semid, ON, fileFlag );
    }
    else{
      printf("error\n");exit(1);
    }

    take = 0;
```

```
    while(1){
      if( fileFlag==0 ){                    /* 入力ファイルの選択 */
        semlock( semid, OFF,  fileFlag+1 );
        if ( ( fp = fopen("data/amp1.dat","r") ) == NULL )
          {printf("can't open file scores amp1.dat\n");exit(1);}
      }
      else if( fileFlag==1 ){
        semlock( semid , OFF,  fileFlag-1 );
        if ( ( fp = fopen("data/amp2.dat","r" ) ) == NULL)
          {printf("can't open file scores amp2.dat\n");exit(1);}
      }

      if( ( fp_x = fopen("data/count.dat","a") ) == NULL ){
        printf("can't open file scores count.dat\n"); exit(1);}
      fprintf( fp_x, "%d ", take);
      fclose(fp_x);
      take ++;

      count=0;                              /* フレーズの取り込み */

      while( ( fscanf( fp, "%lf %lf\n", &phrase.time[count], &phrase.amp[count] ) ) !
= EOF ){
        count++;
      }
      phrase.lastNum = count;
      fclose(fp);

      /* スプライン補間 (FFT用) */
      N=SAMPLING;
      if( ( fp = fopen("data/phrase.dat","w") ) == NULL )
        {printf("can't open file scores phrase.dat\n"); exit(1);}
      if( round_int( pow( 2.0, log2( (double)N ) ) ) != N ){
        printf("but input 'N' %d\n",round_int( pow( 2.0, log2( (double)N ) ) ) );
        printf("log2(N) %d\n",round_int( log2( (double)N ) ) );
        exit(0);}
      spl( N, re, &phrase, 1, fp);
      fclose(fp);

      /* 平均と標準偏差 */
      if( ( fp_h = fopen("data/deviation.dat","a") ) == NULL ){
        printf("can't open file scores deviation.dat\n"); exit(1);}
      if( (fp_m = fopen("data/mean.dat","a") ) == NULL ){
        printf("can't open file scores mean.dat\n");  exit(1);}
      u     = mean( re, 0, N );
      sigma = deviation( u, re, 0, N );
      fprintf( fp_m, "%lf  ", u-60 );    /* (平均)ー 60 */
      fprintf( fp_h, "%lf  ", sigma );
      fclose(fp_m);
      fclose(fp_h);

      /* フーリエ変換 */
      for( k=0; k < N; k++ ){
        re[k] = (re[k]-u)/sigma;
        im[k]=0.0;
      }
      FFT( re, im, round_int(log2( (double)N )), 1);

    /* 自己相関関数 (階段状補間) */
      if( ( fp = fopen("data/correlation_s.dat","a") ) == NULL ){
        printf("can't open file scores correlation_s.dat\n"); exit(0);}
      N=SAMPLING;
      step( N, re, &phrase, -1, fp );
      correlation( N, re, fp );
      fclose(fp);

      if( fileFlag==0 ){
```



```
        semlock( semid, ON, fileFlag+1 );
        fileFlag = 1;
      }
      else if( fileFlag==1 ){
        semlock( semid, ON, fileFlag-1 );
        fileFlag = 0;
      }

    }
    exit(0);
  }
  wait(&stat);
  wait(&stat);

  semnum.val=0;
  if( semctl(semid, 0 , IPC_RMID , semnum )==EOF){  /* セマフォの消去 */
    printf("SemCtl Error  %d\n", errno);
    exit(1);
  }


}
```

```
#include <dmedia/midi.h>
#include <getopt.h>
#include <stdio.h>
#include <sys/syssgi.h>

#include <unistd.h>
#include <sys/types.h>
#include <bstring.h>
#include <sys/time.h>

extern    MDport *mdo;        /* open as many inports as there are */
extern    MDport *mdi;        /* these are out ports */

int initMidi()
{

    int i,x;

        i = mdInit();
        if (i < 0) {
                fprintf(stderr,"MIDI not intialized\n");
             return(-1);
        }

        printf("%d devices available\n", i);
        mdo = (MDport *) calloc(i, sizeof(MDport));
        mdi = (MDport *) calloc(i, sizeof(MDport));

        if (!mdo) {
                fprintf(stderr,"couldn't allocate array of out ports\n");
             return(-1);
        }

        if (!mdi) {
                fprintf(stderr,"couldn't allocate array of in ports\n");
             return(-1);
        }

        for (x = 0; x < i; x++) {
                printf("named %s\n", mdGetName(x));
                mdo[x] = mdOpenOutPort(mdGetName(x));
                if (mdo[x] == NULL) {
                        fprintf(stderr,"Port open failed\n");
                     return(-1);
                }
                mdSetStampMode(mdo[x],MD_DELTATICKS);
                mdSetTemporalBuffering(mdo[x],0);
        }

        for (x = 0; x < i; x++) {
                printf("named %s\n", mdGetName(x));
                mdi[x] = mdOpenInPort(mdGetName(x));
                if (mdi[x] == NULL)  {
                        fprintf(stderr,"Port open failed\n");
                     return(-1);
                }
        }


     return(i);
}
```

```
MDport   *mdo;          /* open as many inports as there are */
MDport   *mdi;          /* these are out ports               */

#define  ON  1          /* セマフォ値 -- */
#define  OFF 0          /* セマフォ値 ++ */

void Midi( semid )
     int semid;         /* semaphoID */
{

  int         count=0, num=0;
  int         flag = 0;
  int         timingClock = 0; /* Timing clock */
  int         maxCount=0;
  int         fileFlag=0;      /* fileFlag->0:ampl.dat ,fileFlag->1:amp2.dat*/
  int         tmp;
  phrase_def  phrase;          /* フレーズの構造体 */
  MDevent     mdv;

  FILE       *fp;              /* 旋律波形の出力ファイル */

  FILE       *fp_t;            /* テンポの出力ファイル   */

  if ( ( fp_t = fopen("data/tempo.dat","w") ) == NULL ){ /* ファイルの初期化 */
    printf("can't open file scores temp.dat\n"); exit(1);}
  fclose(fp_t);

  initMidi();

  mdv.msg[0] = (char)0xfa;                    /* Start */
  mdv.msglen = 1;
  mdSend( mdo[0], &mdv, 1);

  if( fileFlag==0 ){
    if ( ( fp = fopen("data/ampl.dat","w") ) == NULL ){
      printf("can't open file scores\n"); exit(1);}
   }
  else if( fileFlag==1 ){
    if ( ( fp = fopen("data/amp2.dat","w" ) ) == NULL ){
      printf("can't open file scores\n"); exit(1);}
  }

  printf("\n");
  while(1){
    mdReceive( mdi[0], &mdv, 1 );
    if( mdv.msg[0] == (char)0xf8 ){           /* Timing clock */
      timingClock ++;
      if( ( ( timingClock)%(4*24) == 0 ) && (flag) ){
        num ++;
      }

    }
    else if( ( mdv.msg[0] & (char)0xf0) == (char)0x80 ){ /* Channel 1 Note off */
    }
    else if( ( mdv.msg[0] & (char)0xf0) == (char)0x90 ){ /* Channel 2 Note on  */
      if( flag == 0 ){ /* 最初の場合の処理 */
        flag = 1;
        timingClock = 0;
      }
      if( num == BAR_4 ){
        phrase.amp[count]  = phrase.amp[count-1];
        phrase.time[count] = Max_Time;
        maxCount = count;
        if ( ( fp_t = fopen("data/tempo.dat","a") ) == NULL ){
          printf("can't open file scores temp.dat\n"); exit(1);}
        fprintf(fp_t,"%d ",maxCount); /* テンポの出力 */
```



```
        fclose(fp_t);

        for( count=0; count <= maxCount; count++ ){ /* フレーズのファイル出力 */
          fprintf(fp,"%lf %lf\n",phrase.time[count],phrase.amp[count]);
        }
        fclose( fp );

        if( fileFlag==0 ){
          semlock( semid , OFF,  fileFlag );
          semlock( semid , ON, fileFlag+1 );
          if ( ( fp = fopen("data/amp2.dat","w") ) == NULL )
            {printf("can't open file scores\n"); exit(0);}
          fileFlag = 1;
        }
        else if( fileFlag==1 ){
          semlock( semid , OFF,  fileFlag );
          semlock( semid , ON, fileFlag-1 );
          if ( ( fp = fopen("data/ampl.dat","w" ) ) == NULL)
            {printf("can't open file scores\n"); exit(0);}
          fileFlag = 0;
        }

        count = num = timingClock = 0;
      }
      phrase.amp[count]  = (double)mdv.msg[1];
      phrase.time[count] = (double)timingClock;
      count ++;
    }
    else if( mdv.msg[0] == (char)0xfc ){ /* END */
      fclose(fp);
      break;
    }
  }

  semnum.val=0;
  semlock( semid, OFF, 0 );
  semlock( semid, OFF, 1 );
  if( semctl( semid, 0 , IPC_RMID , semnum )==EOF){ /* セマフォの消去 */
    printf("SemCtl Error  %d\n",errno);
    exit(1);
  }
  if( semctl( semid, 1 , IPC_RMID , semnum )==EOF){
    printf("SemCtl Error  %d\n",errno);
    exit(1);
  }

}

semlock( semid, f, no )
     int f;     /* ロック、アンロック */
     int semid; /* semaphoID */
     int no;    /* 実行するセマフォ番号 */
{
  struct sembuf sb[2];
  extern int errno;

  sb[0].sem_num = no;

  if( f==ON ){
    sb[0].sem_op = -1;
  }
  else{
    sb[0].sem_op =  1;
  }
  sb[0].sem_flg = 0;
  if( semop( semid, sb, 1 ) == EOF ){
```

```
    printf("SemOp' Error %d\n",errno);
    exit(1);
  }
}
```

```
/* スプライン補間 */
void spl( int NN, double re[], phrase_def *phrase, int pr, FILE *file )

{
  int     n , N;
  double  t = 0.0;
  double  *Time, *Amp;
  double  y2[CMAX] , y;

  N    = phrase->lastNum;
  Time = phrase->time - 1;
  Amp  = phrase->amp  - 1;

  spline( Time, Amp, N, (Amp[2]-Amp[1])/(Time[2]-Time[1]),
         (Amp[N]-Amp[N-1])/(Time[N]-Time[N-1]), y2);

   for(n=0, t=0.0; n<NN; n++, t+=Max_Time/(double)NN){
     splint( Time, Amp, y2, N, t, &y);
     re[n] = y;
     if( pr==1 ){
       fprintf(file,"%lf %lf\n", t, re[n]);
     }

   }
}
```

```
#define NRANSI
void spline(double x[], double y[], int n, double yp1, double ypn, double y2[])
[
        int i,k;
        double p,qn,sig,un,*u;

        u=vector(1,n-1);
        if (yp1 > 0.99e30)
                y2[1]=u[1]=0.0;
        else {
                y2[1] = -0.5;
                u[1]=(3.0/(x[2]-x[1]))*((y[2]-y[1])/(x[2]-x[1])-yp1);
        ]
        for (i=2;i<=n-1;i++) {
                sig=(x[i]-x[i-1])/(x[i+1]-x[i-1]);
                p=sig*y2[i-1]+2.0;
                y2[i]=(sig-1.0)/p;
                u[i]=(y[i+1]-y[i])/(x[i+1]-x[i]) - (y[i]-y[i-1])/(x[i]-x[i-1]);
                u[i]=(6.0*u[i]/(x[i+1]-x[i-1])-sig*u[i-1])/p;
        ]
        if (ypn > 0.99e30)
                qn=un=0.0;
        else {
                qn=0.5;
                un=(3.0/(x[n]-x[n-1]))*(ypn-(y[n]-y[n-1])/(x[n]-x[n-1]));
        ]
        y2[n]=(un-qn*u[n-1])/(qn*y2[n-1]+1.0);
        for (k=n-1;k>=1;k--)
                y2[k]=y2[k]*y2[k+1]+u[k];
        free_vector(u,1,n-1);
}
#undef NRANSI
/* (C) Copr. 1986-92 Numerical Recipes Software 1+5-5i. */
```

```
void splint(double xa[], double ya[], double y2a[], int n, double x, double *y)
{
        void nrerror(char error_text[]);
        int klo,khi,k;
        double h,b,a;

        klo=1;
        khi=n;
        while (khi-klo > 1) {
                k=(khi+klo) >> 1;
                if (xa[k] > x) khi=k;
                else klo=k;
        }
        h=xa[khi]-xa[klo];
        if (h == 0.0) nrerror("Bad xa input to routine splint");
        a=(xa[khi]-x)/h;
        b=(x-xa[klo])/h;
        *y=a*ya[klo]+b*ya[khi]+((a*a*a-a)*y2a[klo]+(b*b*b-b)*y2a[khi])*(h*h)/6.0;
}
/* (C) Copr. 1986-92 Numerical Recipes Software 1+5-5i. */
```

```
/* 階段状補間 */
void step(int N,double re[],phrase_def *phrase,int pr,FILE *file)
{
  int     k, count=0, num;
  double  t, Time[CMAX];
  double  length, height;

  for(t=0.0,num=0; t<Max_Time; t+=Max_Time/(double)N,num++){
  label1:
    if(phrase->time[count] == t){
      Time[num] = t;
      re[num] = phrase->amp[count];}
    else if(phrase->time[count+1] == t){
      Time[num] = t;
      re[num] = phrase->amp[count+1];}
    else if((phrase->time[count] < t) && (phrase->time[count+1] > t)){
      Time[num] = t;
      re[num]=phrase->amp[count];}
    else{count++; goto label1;}
  }
  if(pr==1){
    for(k=0; k<N; k++){
      fprintf(file,"%lf %lf\n",(Time[k]/60.0)*(N/2.0),re[k]);
    }
  }

}
```

```
/* 線形補間関数中心部 */
void liner(int N,double re[],phrase_def *phrase,int pr,FILE *file)

{
  int     k, count=0, num;
  double  t, Time[CMAX], tmp;
  double  length, height;

  for(t=0.0,num=0; t<Max_Time; t+=Max_Time/(double)N,num++){
  label1:
    if(phrase->time[count] == t){
      Time[num] = t;
      re[num] = phrase->amp[count];
    }
    else if(phrase->time[count+1] == t){
      Time[num] = t;
      re[num] = phrase->amp[count+1];
    }
    else if((phrase->time[count] < t) && (phrase->time[count+1] > t)){
      Time[num] = t;
      length = phrase->time[count+1] - phrase->time[count];
      height = phrase->amp[count+1] - phrase->amp[count];
      tmp    = t  - phrase->time[count];
      re[num]=height/length*tmp +  phrase->amp[count];
    }
    else{
      count++; goto label1;
    }

  }
  if(pr==1){
    for(k=0; k<N; k++){
      fprintf(file,"%lf %lf\n",(Time[k]/60.0)*(N/2.0),re[k]);
    }
  }

}
```

```
/*高速フーリエ変換 : FFT */

void birv(double a[],int length,int ex);
void cstb(int length,int inv,double sin_tbl[],double cos_tbl[]);
void fft_core(double re[],double im[],int length,int ex,double sin_tbl[],double cos_t
bl[]);
void smooth(int N,double a[]);


void FFT(double re[],double im[],int ex,int inv)
/* re : データ実数部（入出力兼用）*/
/* im : データ虚数部（入出力兼用）*/
/* ex : データ個数を２のべき乗で与える（データ個数＝２のex乗個）*/
/* inv: 1 ：DFT 、-1 :IDFT*/
/* 出力ファイルのポインタ */
[

  int     i, length =1;
  double  *sin_tbl;       /* sin計算用テーブル */
  double  *cos_tbl;       /* cos計算用テーブル */
  double  pw[CMAX];       /* パワースペクトラム */
  double  Freq1;          /* スペクトル（超低周波）の総量 */
  double  Freq2;          /* スペクトル（低周波）の総量 */
  double  Freq3;          /* スペクトル（中周波）の総量 */
  double  Freq4;          /* スペクトル（高周波）の総量 */
  double  Freq5;          /* スペクトル（超高周波）の総量 */

  FILE    *fp_f, *fp_c;

  if((fp_f = fopen("data/fft.dat","w")) == NULL){
    printf("can't open file scores count.dat\n"); exit(0);]
  if((fp_c = fopen("data/fft_c.dat","w")) == NULL){
    printf("can't open file scores mean.dat\n"); exit(0);]

  for(i=0; i<ex; i++){length *=2;] /*データ個数の計算*/
  sin_tbl=(double*)malloc((size_t)length*sizeof(double));
  cos_tbl=(double*)malloc((size_t)length*sizeof(double));
  if((sin_tbl==NULL)||(cos_tbl==NULL)){
    printf("Memory mot enough\n");
    exit(0);
  ]
  cstb(length,inv,sin_tbl,cos_tbl); /*sin,cosテーブル作成*/
  fft_core(re,im,length,ex,sin_tbl,cos_tbl);
  Freq1 = Freq2 = Freq3 = Freq4 = Freq5 = 0.0;
  for(i=1; i< length/2; i++){       /* パワースペクトルの出力 */
    pw[i]=power(re[i],im[i]);
      fprintf(fp_f," %lf",pw[i]);
      fprintf(fp_c," %d",i);
/*この処理は、パワースペクトルを求める際に、あらかじめ１００で
割ってやることにより、スペクトルの符合をあらかじめ、マイナスにし
ている。こうすることにより、各成分の和をとった場合、プラスと
マイナスが相殺されるのを防ぐ                                    */

    if( i < length/(32) ){
      Freq1 += log10(sqrt((re[i]*re[i])/100. + (im[i]*im[i])/100.));
    ]
    if( i < length/(16) ){
      Freq2 += log10(sqrt((re[i]*re[i])/100. + (im[i]*im[i])/100.));
    ]
    if( i < length/(8) ){
      Freq3 += log10(sqrt((re[i]*re[i])/100. + (im[i]*im[i])/100.));
    ]
    else if( i < length/4 ){
      Freq4 += log10(sqrt((re[i]*re[i])/100. + (im[i]*im[i])/100.));
    ]
    else{
```



```
      Freq5 += log10(sqrt((re[i]*re[i])/100. + (im[i]*im[i])/100.));
    ]
  ]
  printf("Freq1=%6.2lf Freq2=%6.2lf 3Freq=%6.2lf 4Freq=%6.2lf 5Freq=%6.2lf\n "
        ,Freq1, Freq2,Freq3,Freq4,Freq5);
  free(sin_tbl);
  free(cos_tbl);
  fclose(fp_f);
  fclose(fp_c);
]

void fft_core(double re[],double im[],int length,int ex,double sin_tbl[],double cos_t
bl[])
{
  int     i,j,k,w,j1,j2;
  int     numb,lenb,timb;
  double  xr,xi,yr,yi,nrml;

  numb=1;
  lenb=length;
  for(i=0;i<ex;i++){
    lenb/=2;
    timb=0;
    for(j=0; j<numb; j++){
      w=0;
      for(k=0; k<lenb; k++){
        j1=timb+k;
        j2=j1+lenb;
        xr=re[j1];
        xi=im[j1];
        yr=re[j2];
        yi=im[j2];
        re[j1] = xr+yr;
        im[j1] = xi+yi;
        xr = xr-yr;
        xi = xi-yi;
        re[j2]=xr*cos_tbl[w]-xi*sin_tbl[w];
        im[j2]=xr*sin_tbl[w]+xi*cos_tbl[w];
        w+=numb;
      ]
      timb+=(2*lenb);
    ]
    numb*=2;
  ]
  birv(re,length,ex); /*実数データの並び替え*/
  birv(im,length,ex); /*実数データの並び替え*/
  nrml=1.0/sqrt((double)length);
  for(i=0; i<length; i++){
    re[i]*=nrml;
    im[i]*=nrml;
  ]
]

void cstb(int length,int inv,double sin_tbl[],double cos_tbl[])
{
  int     i;
  double  xx,arg;

  xx=((-PI)*2.0)/(double)length;
  if(inv<0)[xx=-xx;]
  for(i=0; i<length; i++){
    arg=(double)i*xx;
    sin_tbl[i]=sin(arg);
    cos_tbl[i]=cos(arg);
  ]
]
```

```
void birv(double a[],int length,int ex)
     /*データの並べ替え*/
{
  int i,ii,k,bit;
  static double *b; /*作業用バッファ*/

  b=(double*)malloc((size_t)length*sizeof(double));
  if(b==NULL){
    printf("Memory mot enough\n");
    exit(0);
  }
  for(i=0; i<length; i++){
    for(k=0, ii=i,bit=0;;bit<<=1,ii>>=1){
      bit=(ii&1)|bit;
      if(++k==ex){break;}
    }
    b[i]=a[bit];
  }
  for(i=0; i<length; i++){
    a[i]=b[i];}
  free(b);
}
```

```
void correlation( int N, double xr[], FILE *file ){

  int      i, k;
  int      Mine_Count;
  int      count=0;
  double   peek=0.0;
  double   relate=0.0;
  double   u, sigma;
  double   fore, mid, back;
  FILE    *fp_s, *fp_c;

  if((fp_s = fopen("data/correlation.dat","w")) == NULL){
    printf("can't open file scores count.dat\n"); exit(0);}
  if((fp_c = fopen("data/correlation_c.dat","w")) == NULL){
    printf("can't open file scores count.dat\n"); exit(0);}

  fore = mid = back = 0.0;
  Mine_Count = round_int(( Mine_Time*N)/(Max_Time ));

  /* データの正規化 */
  u     = mean( xr, 0, N );
  sigma = deviation( u, xr, 0, N );
  for(k=0; k < N; k++){
    xr[k] = (xr[k]-u)/sigma;
  }

  /* 相関関数の計算 */
  peek = 0.0;
  fore = mid = back = 0.0;
  for(i=0; i < N/2+2; i+=Mine_Count){
    for(k=0; k < N; k++){
      if( k+i >= N ){
        relate += xr[k]*xr[k+i-N];
      }
      else{
        relate+=xr[k]*xr[k+i];
      }
    }
    fprintf(fp_c," %d",i);
    fprintf(fp_s," %lf",relate/(N));
    fore = mid;
    mid  = back;
    back = relate+1.0*N;        /* 相関関数の底上げ */

    if( (i >= 3) && (fore < mid ) && (mid > back) ){ /* ピークの検出 */
      if(peek < mid){
        peek   = mid;
        count  = i-1;
      }
    }

   relate=0.0;
  }
  fprintf( file," %d", count );

  fclose(fp_s);
  fclose(fp_c);

}
```

```
#ifndef _NR_UTILS_H_
#define _NR_UTILS_H_

static double sqrarg;
#define SQR(a) ((sqrarg=(a)) == 0.0 ? 0.0 : sqrarg*sqrarg)

static double dsqrarg;
#define DSQR(a) ((dsqrarg=(a)) == 0.0 ? 0.0 : dsqrarg*dsqrarg)

static double dmaxarg1,dmaxarg2;
#define DMAX(a,b) (dmaxarg1=(a),dmaxarg2=(b),(dmaxarg1) > (dmaxarg2) ?\
        (dmaxarg1) : (dmaxarg2))

static double dminarg1,dminarg2;
#define DMIN(a,b) (dminarg1=(a),dminarg2=(b),(dminarg1) < (dminarg2) ?\
        (dminarg1) : (dminarg2))

static double maxarg1,maxarg2;
#define FMAX(a,b) (maxarg1=(a),maxarg2=(b),(maxarg1) > (maxarg2) ?\
        (maxarg1) : (maxarg2))

static double minarg1,minarg2;
#define FMIN(a,b) (minarg1=(a),minarg2=(b),(minarg1) < (minarg2) ?\
        (minarg1) : (minarg2))

static long lmaxarg1,lmaxarg2;
#define LMAX(a,b) (lmaxarg1=(a),lmaxarg2=(b),(lmaxarg1) > (lmaxarg2) ?\
        (lmaxarg1) : (lmaxarg2))

static long lminarg1,lminarg2;
#define LMIN(a,b) (lminarg1=(a),lminarg2=(b),(lminarg1) < (lminarg2) ?\
        (lminarg1) : (lminarg2))

static int imaxarg1,imaxarg2;
#define IMAX(a,b) (imaxarg1=(a),imaxarg2=(b),(imaxarg1) > (imaxarg2) ?\
        (imaxarg1) : (imaxarg2))

static int iminarg1,iminarg2;
#define IMIN(a,b) (iminarg1=(a),iminarg2=(b),(iminarg1) < (iminarg2) ?\
        (iminarg1) : (iminarg2))

#define SIGN(a,b) ((b) >= 0.0 ? fabs(a) : -fabs(a))

#if defined(__STDC__) || defined(ANSI) || defined(NRANSI) /* ANSI */

void nrerror(char error_text[]);
double *vector(long nl, long nh);
int *ivector(long nl, long nh);
unsigned char *cvector(long nl, long nh);
unsigned long *lvector(long nl, long nh);
double *dvector(long nl, long nh);
double **matrix(long nrl, long nrh, long ncl, long nch);
double **dmatrix(long nrl, long nrh, long ncl, long nch);
int **imatrix(long nrl, long nrh, long ncl, long nch);
double **submatrix(double **a, long oldrl, long oldrh, long oldcl, long oldch,
        long newrl, long newcl);
double **convert_matrix(double *a, long nrl, long nrh, long ncl, long nch);
double ***f3tensor(long nrl, long nrh, long ncl, long nch, long ndl, long ndh);
void free_vector(double *v, long nl, long nh);
void free_ivector(int *v, long nl, long nh);
void free_cvector(unsigned char *v, long nl, long nh);
void free_lvector(unsigned long *v, long nl, long nh);
void free_dvector(double *v, long nl, long nh);
void free_matrix(double **m, long nrl, long nrh, long ncl, long nch);
void free_dmatrix(double **m, long nrl, long nrh, long ncl, long nch);
void free_imatrix(int **m, long nrl, long nrh, long ncl, long nch);
```



```
void free_submatrix(double **b, long nrl, long nrh, long ncl, long nch);
void free_convert_matrix(double **b, long nrl, long nrh, long ncl, long nch);
void free_f3tensor(double ***t, long nrl, long nrh, long ncl, long nch,
        long ndl, long ndh);

#else /* ANSI */
/* traditional - K&R */

void nrerror();
double *vector();
double **matrix();
double **submatrix();
double **convert_matrix();
double ***f3tensor();
double *dvector();
double **dmatrix();
int *ivector();
int **imatrix();
unsigned char *cvector();
unsigned long *lvector();
void free_vector();
void free_dvector();
void free_ivector();
void free_cvector();
void free_lvector();
void free_matrix();
void free_submatrix();
void free_convert_matrix();
void free_dmatrix();
void free_imatrix();
void free_f3tensor();

#endif /* ANSI */

#endif /* _NR_UTILS_H_ */
```

```
#if defined(__STDC__) || defined(ANSI) || defined(NRANSI) /* ANSI */

#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#define NR_END 1
#define FREE_ARG char*

void nrerror(char error_text[])
/* Numerical Recipes standard error handler */
{
        fprintf(stderr,"Numerical Recipes run-time error...\n");
        fprintf(stderr,"%s\n",error_text);
        fprintf(stderr,"...now exiting to system...\n");
        exit(1);
]

double *vector(long nl, long nh)
/* allocate a double vector with subscript range v[nl..nh] */
{
        double *v;

        v=(double *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(double)));
        if (!v) nrerror("allocation failure in vector()");
        return v-nl+NR_END;
]

int *ivector(long nl, long nh)
/* allocate an int vector with subscript range v[nl..nh] */
{
        int *v;

        v=(int *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(int)));
        if (!v) nrerror("allocation failure in ivector()");
        return v-nl+NR_END;
}

unsigned char *cvector(long nl, long nh)
/* allocate an unsigned char vector with subscript range v[nl..nh] */
{
        unsigned char *v;

        v=(unsigned char *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(unsigned char)));
        if (!v) nrerror("allocation failure in cvector()");
        return v-nl+NR_END;
]

unsigned long *lvector(long nl, long nh)
/* allocate an unsigned long vector with subscript range v[nl..nh] */
{
        unsigned long *v;

        v=(unsigned long *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(long)));
        if (!v) nrerror("allocation failure in lvector()");
        return v-nl+NR_END;
]

double *dvector(long nl, long nh)
/* allocate a double vector with subscript range v[nl..nh] */
{
        double *v;

        v=(double *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(double)));
        if (!v) nrerror("allocation failure in dvector()");
        return v-nl+NR_END;
]
```



```
double **matrix(long nrl, long nrh, long ncl, long nch)
/* allocate a double matrix with subscript range m[nrl..nrh][ncl..nch] */
{
        long i, nrow=nrh-nrl+1,ncol=nch-ncl+1;
        double **m;

        /* allocate pointers to rows */
        m=(double **) malloc((size_t)((nrow+NR_END)*sizeof(double*)));
        if (!m) nrerror("allocation failure 1 in matrix()");
        m += NR_END;
        m -= nrl;

        /* allocate rows and set pointers to them */
        m[nrl]=(double *) malloc((size_t)((nrow*ncol+NR_END)*sizeof(double)));
        if (!m[nrl]) nrerror("allocation failure 2 in matrix()");
        m[nrl] += NR_END;
        m[nrl] -= ncl;

        for(i=nrl+1;i<=nrh;i++) m[i]=m[i-1]+ncol;

        /* return pointer to array of pointers to rows */
        return m;
]

double **dmatrix(long nrl, long nrh, long ncl, long nch)
/* allocate a double matrix with subscript range m[nrl..nrh][ncl..nch] */
{
        long i, nrow=nrh-nrl+1,ncol=nch-ncl+1;
        double **m;

        /* allocate pointers to rows */
        m=(double **) malloc((size_t)((nrow+NR_END)*sizeof(double*)));
        if (!m) nrerror("allocation failure 1 in matrix()");
        m += NR_END;
        m -= nrl;

        /* allocate rows and set pointers to them */
        m[nrl]=(double *) malloc((size_t)((nrow*ncol+NR_END)*sizeof(double)));
        if (!m[nrl]) nrerror("allocation failure 2 in matrix()");
        m[nrl] += NR_END;
        m[nrl] -= ncl;

        for(i=nrl+1;i<=nrh;i++) m[i]=m[i-1]+ncol;

        /* return pointer to array of pointers to rows */
        return m;
]

int **imatrix(long nrl, long nrh, long ncl, long nch)
/* allocate a int matrix with subscript range m[nrl..nrh][ncl..nch] */
{
        long i, nrow=nrh-nrl+1,ncol=nch-ncl+1;
        int **m;

        /* allocate pointers to rows */
        m=(int **) malloc((size_t)((nrow+NR_END)*sizeof(int*)));
        if (!m) nrerror("allocation failure 1 in matrix()");
        m += NR_END;
        m -= nrl;


        /* allocate rows and set pointers to them */
        m[nrl]=(int *) malloc((size_t)((nrow*ncol+NR_END)*sizeof(int)));
        if (!m[nrl]) nrerror("allocation failure 2 in matrix()");
        m[nrl] += NR_END;
```

```
        m[nrl] -= ncl;

        for(i=nrl+1;i<=nrh;i++) m[i]=m[i-1]+ncol;

        /* return pointer to array of pointers to rows */
        return m;
}

double **submatrix(double **a, long oldrl, long oldrh, long oldcl, long oldch,
        long newrl, long newcl)
/* point a submatrix [newrl..][newcl..] to a[oldrl..oldrh][oldcl..oldch] */
{
        long i,j,nrow=oldrh-oldrl+1,ncol=oldcl-newcl;
        double **m;

        /* allocate array of pointers to rows */
        m=(double **) malloc((size_t) ((nrow+NR_END)*sizeof(double*)));
        if (!m) nrerror("allocation failure in submatrix()");
        m += NR_END;
        m -= newrl;

        /* set pointers to rows */
        for(i=oldrl,j=newrl;i<=oldrh;i++,j++) m[j]=a[i]+ncol;

        /* return pointer to array of pointers to rows */
        return m;
}

double **convert_matrix(double *a, long nrl, long nrh, long ncl, long nch)
/* allocate a double matrix m[nrl..nrh][ncl..nch] that points to the matrix
declared in the standard C manner as a[nrow][ncol], where nrow=nrh-nrl+1
and ncol=nch-ncl+1. The routine should be called with the address
&a[0][0] as the first argument. */
{
        long i,j,nrow=nrh-nrl+1,ncol=nch-ncl+1;
        double **m;

        /* allocate pointers to rows */
        m=(double **) malloc((size_t) ((nrow+NR_END)*sizeof(double*)));
        if (!m) nrerror("allocation failure in convert_matrix()");
        m += NR_END;
        m -= nrl;

        /* set pointers to rows */
        m[nrl]=a-ncl;
        for(i=1,j=nrl+1;i<nrow;i++,j++) m[j]=m[j-1]+ncol;
        /* return pointer to array of pointers to rows */
        return m;
}

double ***f3tensor(long nrl, long nrh, long ncl, long nch, long ndl, long ndh)
/* allocate a double 3tensor with range t[nrl..nrh][ncl..nch][ndl..ndh] */
{
        long i,j,nrow=nrh-nrl+1,ncol=nch-ncl+1,ndep=ndh-ndl+1;
        double ***t;

        /* allocate pointers to pointers to rows */
        t=(double ***) malloc((size_t)((nrow+NR_END)*sizeof(double**)));
        if (!t) nrerror("allocation failure 1 in f3tensor()");
        t += NR_END;
        t -= nrl;

        /* allocate pointers to rows and set pointers to them */
        t[nrl]=(double **) malloc((size_t)((nrow*ncol+NR_END)*sizeof(double*)));
        if (!t[nrl]) nrerror("allocation failure 2 in f3tensor()");
        t[nrl] += NR_END;
```



```
        t[nrl] -= ncl;

        /* allocate rows and set pointers to them */
        t[nrl][ncl]=(double *) malloc((size_t)((nrow*ncol*ndep+NR_END)*sizeof(double)
));
        if (!t[nrl][ncl]) nrerror("allocation failure 3 in f3tensor()");
        t[nrl][ncl] += NR_END;
        t[nrl][ncl] -= ndl;

        for(j=ncl+1;j<=nch;j++) t[nrl][j]=t[nrl][j-1]+ndep;
        for(i=nrl+1;i<=nrh;i++) {
                t[i]=t[i-1]+ncol;
                t[i][ncl]=t[i-1][ncl]+ncol*ndep;
                for(j=ncl+1;j<=nch;j++) t[i][j]=t[i][j-1]+ndep;
        }

        /* return pointer to array of pointers to rows */
        return t;
}

void free_vector(double *v, long nl, long nh)
/* free a double vector allocated with vector() */
{
        free((FREE_ARG) (v+nl-NR_END));
}

void free_ivector(int *v, long nl, long nh)
/* free an int vector allocated with ivector() */
{
        free((FREE_ARG) (v+nl-NR_END));
}

void free_cvector(unsigned char *v, long nl, long nh)
/* free an unsigned char vector allocated with cvector() */
{
        free((FREE_ARG) (v+nl-NR_END));
}

void free_lvector(unsigned long *v, long nl, long nh)
/* free an unsigned long vector allocated with lvector() */
{
        free((FREE_ARG) (v+nl-NR_END));
}

void free_dvector(double *v, long nl, long nh)
/* free a double vector allocated with dvector() */
{
        free((FREE_ARG) (v+nl-NR_END));
}

void free_matrix(double **m, long nrl, long nrh, long ncl, long nch)
/* free a double matrix allocated by matrix() */
{
        free((FREE_ARG) (m[nrl]+ncl-NR_END));
        free((FREE_ARG) (m+nrl-NR_END));
}

void free_dmatrix(double **m, long nrl, long nrh, long ncl, long nch)
/* free a double matrix allocated by dmatrix() */
{
        free((FREE_ARG) (m[nrl]+ncl-NR_END));
        free((FREE_ARG) (m+nrl-NR_END));
}

void free_imatrix(int **m, long nrl, long nrh, long ncl, long nch)
/* free an int matrix allocated by imatrix() */
```

```
{
        free((FREE_ARG) (m[nrl]+ncl-NR_END));
        free((FREE_ARG) (m+nrl-NR_END));
}

void free_submatrix(double **b, long nrl, long nrh, long ncl, long nch)
/* free a submatrix allocated by submatrix() */
{
        free((FREE_ARG) (b+nrl-NR_END));
}

void free_convert_matrix(double **b, long nrl, long nrh, long ncl, long nch)
/* free a matrix allocated by convert_matrix() */
{
        free((FREE_ARG) (b+nrl-NR_END));
}

void free_f3tensor(double ***t, long nrl, long nrh, long ncl, long nch,
        long ndl, long ndh)
/* free a double f3tensor allocated by f3tensor() */
{
        free((FREE_ARG) (t[nrl][ncl]+ndl-NR_END));
        free((FREE_ARG) (t[nrl]+ncl-NR_END));
        free((FREE_ARG) (t+nrl-NR_END));
}

#else /* ANSI */
/* traditional - K&R */

#include <stdio.h>
#define NR_END 1
#define FREE_ARG char*

void nrerror(error_text)
char error_text[];
/* Numerical Recipes standard error handler */
{
        void exit();

        fprintf(stderr,"Numerical Recipes run-time error...\n");
        fprintf(stderr,"%s\n",error_text);
        fprintf(stderr,"...now exiting to system...\n");
        exit(1);
}

double *vector(nl,nh)
long nh,nl;
/* allocate a double vector with subscript range v[nl..nh] */
{
        double *v;

        v=(double *)malloc((unsigned int) ((nh-nl+1+NR_END)*sizeof(double)));
        if (!v) nrerror("allocation failure in vector()");
        return v-nl+NR_END;
}

int *ivector(nl,nh)
long nh,nl;
/* allocate an int vector with subscript range v[nl..nh] */
{
        int *v;

        v=(int *)malloc((unsigned int) ((nh-nl+1+NR_END)*sizeof(int)));
        if (!v) nrerror("allocation failure in ivector()");
        return v-nl+NR_END;
}
```



```
unsigned char *cvector(nl,nh)
long nh,nl;
/* allocate an unsigned char vector with subscript range v[nl..nh] */
{
        unsigned char *v;

        v=(unsigned char *)malloc((unsigned int) ((nh-nl+1+NR_END)*sizeof(unsigned ch
ar)));
        if (!v) nrerror("allocation failure in cvector()");
        return v-nl+NR_END;
}

unsigned long *lvector(nl,nh)
long nh,nl;
/* allocate an unsigned long vector with subscript range v[nl..nh] */
{
        unsigned long *v;

        v=(unsigned long *)malloc((unsigned int) ((nh-nl+1+NR_END)*sizeof(long)));
        if (!v) nrerror("allocation failure in lvector()");
        return v-nl+NR_END;
}

double *dvector(nl,nh)
long nh,nl;
/* allocate a double vector with subscript range v[nl..nh] */
{
        double *v;

        v=(double *)malloc((unsigned int) ((nh-nl+1+NR_END)*sizeof(double)));
        if (!v) nrerror("allocation failure in dvector()");
        return v-nl+NR_END;
}

double **matrix(nrl,nrh,ncl,nch)
long nch,ncl,nrh,nrl;
/* allocate a double matrix with subscript range m[nrl..nrh][ncl..nch] */
{
        long i, nrow=nrh-nrl+1,ncol=nch-ncl+1;
        double **m;

        /* allocate pointers to rows */
        m=(double **) malloc((unsigned int)((nrow+NR_END)*sizeof(double*)));
        if (!m) nrerror("allocation failure 1 in matrix()");
        m += NR_END;
        m -= nrl;

        /* allocate rows and set pointers to them */
        m[nrl]=(double *) malloc((unsigned int)((nrow*ncol+NR_END)*sizeof(double)));
        if (!m[nrl]) nrerror("allocation failure 2 in matrix()");
        m[nrl] += NR_END;
        m[nrl] -= ncl;

        for(i=nrl+1;i<=nrh;i++) m[i]=m[i-1]+ncol;

        /* return pointer to array of pointers to rows */
        return m;
}

double **dmatrix(nrl,nrh,ncl,nch)
long nch,ncl,nrh,nrl;
/* allocate a double matrix with subscript range m[nrl..nrh][ncl..nch] */
{
        long i, nrow=nrh-nrl+1,ncol=nch-ncl+1;
        double **m;
```

```
	/* allocate pointers to rows */
	m=(double **) malloc((unsigned int)((nrow+NR_END)*sizeof(double*)));
	if (!m) nrerror("allocation failure 1 in matrix()");
	m += NR_END;
	m -= nrl;

	/* allocate rows and set pointers to them */
	m[nrl]=(double *) malloc((unsigned int)((nrow*ncol+NR_END)*sizeof(double)));
	if (!m[nrl]) nrerror("allocation failure 2 in matrix()");
	m[nrl] += NR_END;
	m[nrl] -= ncl;

	for(i=nrl+1;i<=nrh;i++) m[i]=m[i-1]+ncol;

	/* return pointer to array of pointers to rows */
	return m;
}

int **imatrix(nrl,nrh,ncl,nch)
long nch,ncl,nrh,nrl;
/* allocate a int matrix with subscript range m[nrl..nrh][ncl..nch] */
{
	long i, nrow=nrh-nrl+1,ncol=nch-ncl+1;
	int **m;

	/* allocate pointers to rows */
	m=(int **) malloc((unsigned int)((nrow+NR_END)*sizeof(int*)));
	if (!m) nrerror("allocation failure 1 in matrix()");
	m += NR_END;
	m -= nrl;


	/* allocate rows and set pointers to them */
	m[nrl]=(int *) malloc((unsigned int)((nrow*ncol+NR_END)*sizeof(int)));
	if (!m[nrl]) nrerror("allocation failure 2 in matrix()");
	m[nrl] += NR_END;
	m[nrl] -= ncl;

	for(i=nrl+1;i<=nrh;i++) m[i]=m[i-1]+ncol;

	/* return pointer to array of pointers to rows */
	return m;
}

double **submatrix(a,oldrl,oldrh,oldcl,oldch,newrl,newcl)
double **a;
long newcl,newrl,oldch,oldcl,oldrh,oldrl;
/* point a submatrix [newrl..][newcl..] to a[oldrl..oldrh][oldcl..oldch] */
{
	long i,j,nrow=oldrh-oldrl+1,ncol=oldcl-newcl;
	double **m;

	/* allocate array of pointers to rows */
	m=(double **) malloc((unsigned int) ((nrow+NR_END)*sizeof(double*)));
	if (!m) nrerror("allocation failure in submatrix()");
	m += NR_END;
	m -= newrl;

	/* set pointers to rows */
	for(i=oldrl,j=newrl;i<=oldrh;i++,j++) m[j]=a[i]+ncol;

	/* return pointer to array of pointers to rows */
	return m;
}
```



```
double **convert_matrix(a,nrl,nrh,ncl,nch)
double *a;
long nch,ncl,nrh,nrl;
/* allocate a double matrix m[nrl..nrh][ncl..nch] that points to the matrix
declared in the standard C manner as a[nrow][ncol], where nrow=nrh-nrl+1
and ncol=nch-ncl+1. The routine should be called with the address
&a[0][0] as the first argument. */
{
	long i,j,nrow=nrh-nrl+1,ncol=nch-ncl+1;
	double **m;

	/* allocate pointers to rows */
	m=(double **) malloc((unsigned int) ((nrow+NR_END)*sizeof(double*)));
	if (!m) nrerror("allocation failure in convert_matrix()");
	m += NR_END;
	m -= nrl;

	/* set pointers to rows */
	m[nrl]=a-ncl;
	for(i=1,j=nrl+1;i<nrow;i++,j++) m[j]=m[j-1]+ncol;
	/* return pointer to array of pointers to rows */
	return m;
}

double ***f3tensor(nrl,nrh,ncl,nch,ndl,ndh)
long nch,ncl,ndh,ndl,nrh,nrl;
/* allocate a double 3tensor with range t[nrl..nrh][ncl..nch][ndl..ndh] */
{
	long i,j,nrow=nrh-nrl+1,ncol=nch-ncl+1,ndep=ndh-ndl+1;
	double ***t;

	/* allocate pointers to pointers to rows */
	t=(double ***) malloc((unsigned int)((nrow+NR_END)*sizeof(double**)));
	if (!t) nrerror("allocation failure 1 in f3tensor()");
	t += NR_END;
	t -= nrl;

	/* allocate pointers to rows and set pointers to them */
	t[nrl]=(double **) malloc((unsigned int)((nrow*ncol+NR_END)*sizeof(double*)))
;
	if (!t[nrl]) nrerror("allocation failure 2 in f3tensor()");
	t[nrl] += NR_END;
	t[nrl] -= ncl;

	/* allocate rows and set pointers to them */
	t[nrl][ncl]=(double *) malloc((unsigned int)((nrow*ncol*ndep+NR_END)*sizeof(d
ouble)));
	if (!t[nrl][ncl]) nrerror("allocation failure 3 in f3tensor()");
	t[nrl][ncl] += NR_END;
	t[nrl][ncl] -= ndl;

	for(j=ncl+1;j<=nch;j++) t[nrl][j]=t[nrl][j-1]+ndep;
	for(i=nrl+1;i<=nrh;i++) {
		t[i]=t[i-1]+ncol;
		t[i][ncl]=t[i-1][ncl]+ncol*ndep;
		for(j=ncl+1;j<=nch;j++) t[i][j]=t[i][j-1]+ndep;
	}

	/* return pointer to array of pointers to rows */
	return t;
}

void free_vector(v,nl,nh)
double *v;
long nh,nl;
/* free a double vector allocated with vector() */
```

```
{
        free((FREE_ARG) (v+nl-NR_END));
}

void free_ivector(v,nl,nh)
int *v;
long nh,nl;
/* free an int vector allocated with ivector() */
{
        free((FREE_ARG) (v+nl-NR_END));
}

void free_cvector(v,nl,nh)
long nh,nl;
unsigned char *v;
/* free an unsigned char vector allocated with cvector() */
{
        free((FREE_ARG) (v+nl-NR_END));
}

void free_lvector(v,nl,nh)
long nh,nl;
unsigned long *v;
/* free an unsigned long vector allocated with lvector() */
{
        free((FREE_ARG) (v+nl-NR_END));
}

void free_dvector(v,nl,nh)
double *v;
long nh,nl;
/* free a double vector allocated with dvector() */
{
        free((FREE_ARG) (v+nl-NR_END));
}

void free_matrix(m,nrl,nrh,ncl,nch)
double **m;
long nch,ncl,nrh,nrl;
/* free a double matrix allocated by matrix() */
{
        free((FREE_ARG) (m[nrl]+ncl-NR_END));
        free((FREE_ARG) (m+nrl-NR_END));
}

void free_dmatrix(m,nrl,nrh,ncl,nch)
double **m;
long nch,ncl,nrh,nrl;
/* free a double matrix allocated by dmatrix() */
{
        free((FREE_ARG) (m[nrl]+ncl-NR_END));
        free((FREE_ARG) (m+nrl-NR_END));
}

void free_imatrix(m,nrl,nrh,ncl,nch)
int **m;
long nch,ncl,nrh,nrl;
/* free an int matrix allocated by imatrix() */
{
        free((FREE_ARG) (m[nrl]+ncl-NR_END));
        free((FREE_ARG) (m+nrl-NR_END));
}

void free_submatrix(b,nrl,nrh,ncl,nch)
double **b;
long nch,ncl,nrh,nrl;
```



```
/* free a submatrix allocated by submatrix() */
{
        free((FREE_ARG) (b+nrl-NR_END));
}

void free_convert_matrix(b,nrl,nrh,ncl,nch)
double **b;
long nch,ncl,nrh,nrl;
/* free a matrix allocated by convert_matrix() */
{
        free((FREE_ARG) (b+nrl-NR_END));
}

void free_f3tensor(t,nrl,nrh,ncl,nch,ndl,ndh)
double ***t;
long nch,ncl,ndh,ndl,nrh,nrl;
/* free a double f3tensor allocated by f3tensor() */
{
        free((FREE_ARG) (t[nrl][ncl]+ndl-NR_END));
        free((FREE_ARG) (t[nrl]+ncl-NR_END));
        free((FREE_ARG) (t+nrl-NR_END));
}

#endif /* ANSI */
```

```
double log2(double data);
double log1_5(double data);
double log20(double data);
double power(double a,double b);
double arg(double a,double b);
void Hanning_Window(int M,double w[]);
void Hamming_Window(int M,double w[]);
double mean(double x[],int fast,int last);
double deviation(double u,double x[],int fast,int last);
int round_int(double data);
void low_pass(int N,double x[]);
void FFT_cut(int N,double re[]);

/* パワースペクトラム */
double power(double a,double b)
{
  return(20.0*log10(sqrt(a*a+b*b)));
/*return(20.0*log2(a*a+b*b));*/
/*return(20.0*log1_5(a*a+b*b));*/
]

/* 位相特性 */
double arg(double a,double b)
{
  return(atan2(b,a));
]

/* log2 */
double log2(double data){
  return((log10(data))/log10(2.0));
]

/* log1.5*/
double log1_5(double data){
  return((log10(data))/log10(1.5));
]

/* log20 */
double log20(double data){
  return((log10(data))/log10(20));
]

/* ハニング窓 */
void Hanning_Window(int M,double w[]){

  int n;

  for(n=0; n<2*M+1; n++){
    w[n] = 0.5+0.5*cos((PI*(n-M))/M);
  ]
]

/* ハミング窓 */
void Hamming_Window(int M,double w[]){

  int n;

  for(n=0; n<2*M+1; n++){
    w[n] = 0.54+0.46*cos((PI*(n-M))/M);
  ]
]

/* クイックソート比較用関数(double型)*/
double doublecmp(double *a,double*b)
{
  return(*a-*b);
```



```
]

/* クイックソート比較用関数(int型)*/
int intcmp(int *a,int*b)
{
  return(*a-*b);
]

/* 平均 */
double mean(double x[],int fast,int last)
{
  int     i,N;
  double  tmp=0.0;

  N=last-fast;
  for(i=fast; i<last; i++){
    tmp+=x[i];}
  return(tmp/(double)N);

]

/* 標準偏差 */
double deviation(double u,double x[],int fast,int last)
{
  int i,N;
  double tmp;

  N=last-fast;
  for(i=fast; i<last; i++){
    tmp+= pow((x[i]-u),2.0);}
  return(sqrt(tmp/(double)N));
]

/* 四捨五入 (double型をint型にする) */
int round_int(double data)
{
  if(data-(int)data<0.5){
    return((int)data);
  ]
  else{
    return(ceil(data));
  ]
]

/* FIR ローパスフィルタ (ハニング窓) */
#define M   FILTER_ODER
#define Wc  1/3.0*PI
void low_pass(int N,double x[])
{
  int n, k;
  double h1[CMAX], w[CMAX], tmp[CMAX];

  for(n=2*M; n<N; n++){
    tmp[n]=0.0;
  ]

  /* フィルタ係数の導出 */
  Hamming_Window(M,w);
  for(n=0; n<2*M+1; n++){
    if(n-M==0){
      h1[n]=1.0/3.0;
    ]
    else{
      h1[n]=w[n]*sin((n-M)*Wc)/((n-M)*PI);
    ]
  ]
```

```
  /* フィルタの通過 */
  for(n=2*M; n<N; n++){
    for(k=0; k<2*M+1; k++){
      tmp[n]+=h1[k]*x[n-k];
    }
  }
  for(n=2*M; n<N; n++){
    x[n]=tmp[n];
  }
}
```