(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4301125号
(P4301125)
(45)発行日　平成21年7月22日(2009.7.22)　(24)登録日　平成21年5月1日(2009.5.1)

(51)Int.Cl.　　F I
G10G 1/02 (2006.01)　　G10G 1/02
G10H 1/00 (2006.01)　　G10H 1/00　102Z

請求項の数 6　(全 36 頁)

(21)出願番号　特願2004-258616(P2004-258616)
(22)出願日　平成16年9月6日(2004.9.6)
(65)公開番号　特開2006-72230(P2006-72230A)
(43)公開日　平成18年3月16日(2006.3.16)
審査請求日　平成19年6月11日(2007.6.11)

(73)特許権者　000001443
カシオ計算機株式会社
東京都渋谷区本町１丁目６番２号
(74)代理人　100074099
弁理士　大菅　義之
(72)発明者　副島　淳一郎
東京都羽村市栄町３丁目２番１号　カシオ計算機株式会社羽村技術センター内
(72)発明者　南高　純一
東京都羽村市栄町３丁目２番１号　カシオ計算機株式会社羽村技術センター内

審査官　間宮　嘉誉

最終頁に続く

(54)【発明の名称】運指情報生成装置、及びプログラム

(57)【特許請求の範囲】
【請求項１】
楽器に備えられた演奏操作子群を操作して楽曲を演奏していく際の指の運びである運指を示す運指情報を生成する装置であって、
前記楽曲の演奏内容を示す演奏情報を取得する演奏情報取得手段と、
前記演奏情報取得手段が取得した演奏情報を参照して、該演奏情報が演奏内容を示す楽曲の運指情報を生成する運指情報生成手段と、
前記運指情報生成手段が生成した運指情報を参照して、該運指情報が示す運指を行ううえでの難易度を算出する難易度算出手段と、
前記難易度算出手段が算出した難易度を基に、前記運指情報生成手段が生成した運指情報の修正を行う運指情報修正手段と、
を具備することを特徴とする運指情報生成装置。
【請求項２】
前記運指情報修正手段は、前記演奏情報及び運指情報の夫々が有する複数種の特性の中からいずれかひとつの選択された特性に基づき、前記運指情報生成手段が生成した運指情報に対する修正を行ない、
前記運指情報生成装置はさらに、前記修正された運指情報に対して前記難易度算出手段により再度難易度算出を行なわせ、この難易度算出結果を基に前記選択された特性を変更するとともに、前記運指情報修正手段に対して、前記変更された特性に基づいて前記修正された運指情報の再度修正を行なわせるように制御する制御手段を有する請求項１に記載

の運指情報生成装置。
【請求項３】
前記運指情報修正手段は、前記演奏情報を参照して該演奏情報を１つ以上の部分演奏情報に仮想的に分割し、該部分演奏情報単位で前記選択された特性に基づいた修正を前記運指情報に対して行う、
ことを特徴とする請求項２記載の運指情報生成装置。
【請求項４】
前記選択された特性に基づいた前記部分演奏情報単位の修正は、該部分演奏情報が示す演奏内容のパターン、該演奏内容の種類、該部分演奏情報間の演奏内容の類似性、及び該部分演奏情報から生成された運指情報に従った運指の難易度に基づいた修正を一つ以上、含む、
ことを特徴とする請求項３記載の運指情報生成装置。
【請求項５】
前記部分演奏情報間の演奏内容の類似性に基づいた修正は、該部分演奏情報を参照して生成される運指情報を他の部分演奏情報に適用させる形での修正を含む、
ことを特徴とする請求項４記載の運指情報生成装置。
【請求項６】
楽器に備えられた演奏操作子群を操作して楽曲を演奏していく際の指の運びである運指を示す運指情報を生成する運指情報生成装置として用いられるコンピュータに、
前記楽曲の演奏内容を示す演奏情報を取得する機能と、
前記取得する機能により取得した演奏情報を参照して、該演奏情報が演奏内容を示す楽曲の運指情報を生成する機能と、
前記生成する機能により生成した運指情報を参照して、該運指情報が示す運指を行ううえでの難易度を算出する機能と、
前記算出する機能により算出した難易度を基に、前記生成する機能により生成した運指情報の修正を行う機能と、
を実現させるためのプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、楽器に備えられた演奏操作子群を操作して楽曲を演奏していく際の指の運びである運指を示す運指情報を生成するための技術に関する。
【背景技術】
【０００２】
現在では、多くの鍵盤楽器に、演奏の進行に応じて押鍵（操作）すべき鍵を順次、ユーザ（演奏者）に通知していくナビゲーション機能が搭載されている。しかし、ナビゲーション機能を利用するユーザの多くは技術的に未熟であり、そのナビゲーション機能によって次に押鍵すべき鍵を知ることができても、その鍵をどの指で押鍵すべきかが判らないのが実状である。このことから、鍵の押鍵に使うべき指を表示する演奏動作表示装置を搭載した鍵盤楽器も製品化されている。その演奏動作表示装置を利用することにより、指の運び方を意識した演奏を行えるため、ユーザにとってはより効率的な練習（学習）を行うことができる。
【０００３】
運指情報生成装置は、楽曲の演奏内容を示す演奏情報を参照して、楽器に備えられた鍵盤（演奏操作子群）を操作してその楽曲を演奏していく際の指の運びである運指を示す運指情報を自動的に生成するものである。通常、上記演奏動作表示装置では、運指情報を参照して、演奏の進行に応じて順次、押鍵すべき鍵の押鍵に使うべき指、その指を離鍵させるべきタイミング、等を教示するようになっている。従来の運指情報生成装置としては、例えば特許文献１に記載されたものがある。
【０００４】

特許文献１に記載された従来の運指情報生成装置では、押鍵すべき鍵の押鍵に使うべき指として、その鍵の押鍵を行う難易度を示すコストを指別に算出することにより、その押鍵が比較的に容易に行える指を割り当てている。その鍵以降に押鍵すべき鍵も考慮することにより、滑らかに演奏できるようにさせている。また、和音等の楽曲上の構造（演奏内容の種類）や、指潜り等の運指（演奏）方法に着目してフレーズを判定することにより、フレーズ別に適切な方法で指の割り当てを行うようにさせている。

【０００５】

上記コストや楽曲の構造（演奏内容の種類）、及び運指方法に着目することにより、ユーザにとってより容易な（適切な）運指情報を生成することができる。上記従来の運指情報生成装置では、それらを全て考慮しつつ、運指情報を生成していくようになっていた。このため、演奏情報が示す演奏内容に係わらず、運指情報の生成に常に長い時間がかかるという問題点があった。

【０００６】

楽曲のなかには、その構造等を考慮しなくとも最適、或いはそれに近い運指情報を生成できるものも存在する。ユーザのより快適な利用を可能とさせるために、そのような楽曲の運指情報はより迅速に生成できるようにすることが望ましいと考えられる。

【特許文献１】特開２００１－３３１１７３号公報

【発明の開示】

【発明が解決しようとする課題】

【０００７】

本発明の課題は、演奏情報が示す演奏内容に応じて、運指情報の生成をより迅速に行えるようにするための技術を提供することにある。

【課題を解決するための手段】

【０００８】

本発明の第１の態様の運指情報生成装置は、楽器に備えられた演奏操作子群を操作して楽曲を演奏していく際の指の運びである運指を示す運指情報を生成することを前提とし、それぞれ以下の手段を具備する。

【０００９】

第１の態様の運指情報生成装置は、前記楽曲の演奏内容を示す演奏情報を取得する演奏情報取得手段と、前記演奏情報取得手段が取得した演奏情報を参照して、該演奏情報が演奏内容を示す楽曲の運指情報を生成する運指情報生成手段と、前記運指情報生成手段が生成した運指情報を参照して、該運指情報が示す運指を行ううえでの難易度を算出する難易度算出手段と、前記難易度算出手段が算出した難易度を基に、前記運指情報生成手段が生成した運指情報の修正を行う運指情報修正手段と、を具備する。

【００１０】

なお、前記運指情報修正手段は、前記演奏情報及び運指情報の夫々が有する複数種の特性の中からいずれかひとつ選択された特性に基づき、前記運指情報生成手段が生成した運指情報に対する修正を行ない、前記運指情報生成装置はさらに、前記修正された運指情報に対して前記難易度算出手段により再度難易度算出を行なわせ、この難易度算出結果を基に前記選択された特性を変更するとともに、前記運指情報修正手段に対して、前記変更された特性に基づいて前記修正された運指情報を再度修正を行なわせるように制御する制御手段を有する、ことが望ましい。

【００１１】

上記運指情報修正手段は、演奏情報を参照して該演奏情報を１つ以上の部分演奏情報に仮想的に分割し、該部分演奏情報単位で前記選択された特性に基づいた修正を運指情報に対して行う、ことが望ましい。前記選択された特性に基づいた前記部分演奏情報単位の修正は、該部分演奏情報が示す演奏内容のパターン、該演奏内容の種類、該部分演奏情報間の演奏内容の類似性、及び該部分演奏情報から生成された運指情報に従った運指の難易度、に基づいた修正を一つ以上、含む、ことが望ましい。部分演奏情報間の演奏内容の類似性に基づいた修正については、該部分演奏情報を参照して生成される運指情報を他の部分

演奏情報に適用させる形での修正を含む、ことが望ましい。

【００１３】

本発明の第１の態様のプログラムは、上記第１の態様の運指情報生成装置が具備する手段を実現させるための機能を搭載している。

【発明の効果】

【００１４】

本発明は、演奏情報及び運指情報の夫々が有する複数種の特性の中からいずれかひとつ選択された特性を参照して、その演奏情報が演奏内容を示す楽曲の運指情報を生成し、必要に応じて、生成した運指情報に対する修正（補正）を、前記選択された特性を変更させながら段階的に行わせる。

【００１５】

生成した運指情報に対する修正を必要に応じて行うようにしたことから、運指情報の生成には比較的に簡単なアルゴリズムを採用することができる。そのため、簡単なアルゴリズムで最適、或いはそれに近い運指情報の生成が可能な楽曲（演奏情報）では、その生成を迅速に行えるようになる。そうでない楽曲でも、生成後の修正により、最適、或いはそれに近い運指情報を得ることができる。生成した運指情報、及び修正後の運指情報を対象にその評価を行うようにした場合には、最適、或いはそれに近い運指情報を、必要最小限の時間で確実に得られるようになる。

【発明を実施するための最良の形態】

【００１７】

以下、本発明の実施の形態について、図面を参照しながら詳細に説明する。

図１は、本実施の形態による運指情報生成装置を搭載した電子楽器の構成を説明する図である。

【００１８】

その電子楽器は、図１に示すように、楽器全体の制御を行うＣＰＵ１と、そのＣＰＵ１が実行するプログラムや各種制御用データを格納したＲＯＭ２と、ＣＰＵ１がワークに用いるＲＡＭ３と、各種スイッチ等の操作子を有する入力部４と、例えば液晶表示装置である表示部５と、各種データの格納に用いることができる外部記憶装置６と、演奏操作の対象となる演奏操作子群である鍵盤７と、ＣＰＵ１の指示に従って楽音を発音させるサウンドシステム８と、を備えている。

【００１９】

なお、上記外部記憶装置６は、例えば着脱自在な記録媒体にアクセスするものである。入力部４に備えられた運指情報の生成に係わる操作子としては、運指情報生成の対象となる曲（演奏情報）を指定するための曲指定スイッチ、運指情報の生成を指示するための運指生成スイッチ、運指情報生成の動作モードを設定するためのモード設定スイッチ、その動作モードとして、指定区間（範囲）のみ運指情報を生成対象とするモード（以降「指定区間モード」と呼ぶ）が設定されている場合に、その区間を設定するための範囲設定スイッチ、指定区間の前、或いは後を考慮する生成を行うか否か設定するためのオプション設定スイッチ、及び内容が同一のフレーズを考慮した補正を行うか否か設定するための同一フレーズ補正スイッチ、などが設けられている。

【００２０】

運指情報生成の対象となる演奏データ（情報）は、例えば曲指定スイッチを操作して、外部記憶装置６がアクセス可能なもののなかから選択するようになっている。ＣＰＵ１は、ユーザが選択した演奏データを外部記憶装置６に読み出させ、それをＲＡＭ３に格納する。

【００２１】

図２は運指情報生成のためにＲＡＭ３に格納される各種データの構成を説明する図である。図２（ａ）は演奏データの構成、図２（ｂ）は制御変数をそれぞれ示している。

図２（ａ）に括弧を付して表記の「ＭＥ」は、曲を構成する１音符に係わるデータ（音符データ）を表している。その音符データＭＥは、「ｔｉｍｅ」と表記の発音開始時刻、

「ｇａｔｅ」と表記の発音継続時間、「ｐｉｔｃｈ」と表記の発音ピッチ（ノート番号）、「ｐｏｓｘ」と表記の発音ピッチに対応する鍵の鍵盤座標、「ｔｅｎｄ」と表記のピッチ増減傾向、「ｆｉｇ」と表記の運指番号（その鍵の押鍵に使うべき指を示す番号）、「ｃｏｓｔ」と表記の運指コスト（その鍵を押鍵するうえでの難易度）、「ｐｉｄ」と表記のフレーズ番号（その発音ピッチの音符が属するフレーズに割り当てられる識別用番号）、及び「ｐｓｔａｔ」と表記のフレーズ状態（その音符が属するフレーズ内での位置）、の各データから構成されている。

【００２２】

図示していないデータとしては、鍵の押鍵に使うべき指が右手か否かを示すパートや、その押鍵時の強さを示すベロシティなども存在する。しかし、ここでは説明上、便宜的に省略している。

【００２３】

データｐｏｓｘは、ＭＩＤＩフォーマットで６０のノート番号が割り当てられた鍵を基準（０）として、白鍵毎に２つずつ変化させる値で鍵盤座標を表現したものである。そのように変化させることにより、白鍵は偶数、黒鍵は奇数とさせている。これは、白鍵と黒鍵とでは鍵盤７の長手方向の交差方向上の位置に違いがあるからである。

【００２４】

上記ピッチ増減傾向は、その増減の切り替わる点に当たる音符では０、前の音符と同じピッチ（音高）の音符では±１、前の音符のピッチと異なる音符では±２、で表現するようにしている。その符号は＋はピッチが増加中であることを示し、「－」はピッチが減少中であることを示している。

【００２５】

上記運指番号としては、親指は０、人指し指は１，中指は２、薬指は３、小指は４をそれぞれ割り当てている。上記フレーズ状態としては、フレーズ先頭位置の音符には０、フレーズ終了位置の音符には２、それ以外、つまりその間の音符には１、をそれぞれ割り当てている。

【００２６】

上記ｔｉｍｅ、ｇａｔｅ、及びｐｉｔｃｈは、外部記憶装置６から読み出された演奏データを構成するデータか、或いはそのデータから生成されるものである。それら以外のデータは、運指情報を構成するデータ、或いはそのデータ生成に用いるデータである。

【００２７】

一方、制御変数としては、図２（ｂ）に示すように、「ｅｖ＿ｍａｘ」と表記の全音符数－１の値（演奏データ最大インデックス）、「ｍｏｄｅ」と表記の動作モードを示す値、「ｅｖ＿ｓ」と表記の指定区間先頭インデックス、「ｅｖ＿ｅ」と表記の指定区間末端インデックス、「ｌｏｏｋｕｐｂ」と表記の指定区間前考慮フラグ、「ｌｏｏｋｕｐｆ」と表記の指定区間後考慮フラグ、「ｍｅ＿ｓ」と表記の指定区間前を考慮する場合の指定区間先頭インデックス、「ｍｅ＿ｅと表記の指定区間後を考慮する場合の指定区間末端インデックス、「ｍｅ＿ｔｏｐ」と表記の処理区間先頭インデックス、「ｍｅ＿ｔａｉｌ」と表記の処理区間末端インデックス、「ｌｏｏｐｃｎｔ」と表記の処理ループカウンタ、及び「ｐｈｒｃｏｒ」と表記の同一フレーズ補正実行フラグ、の代入用の変数が用意される。

【００２８】

上記ｍｏｄｅの値としては、上記指定区間モードの設定時には１、演奏データ全体を運指情報の生成の対象にするモード（以降「全体モード」と呼ぶ）の設定時には０、が代入される。上記ｌｏｏｋｕｐｂ、ｌｏｏｋｕｐｆの各値としては、指定区間を越える範囲を考慮する場合には１、そうでない場合には０がそれぞれ代入される。上記ｐｈｒｃｏｒとしても同様に、同一内容のフレーズを対象にした補正を行う場合には１、そうでない場合には０が代入される。上記ｅｖ＿ｓ、ｅｖ＿ｅ、ｌｏｏｋｕｐｂ、ｌｏｏｋｕｐｆ、及びｐｈｒｃｏｒとしては、ユーザの設定に応じた値が代入される。

【００２９】

以降は、図３～図８、図１０～図１４、図１６～図２１、及び図２３～図２９に示す各種処理のフローチャート、並びに図９、図１５、及び図２２に示す各種説明図を参照しつつ、電子楽器の動作について詳細に説明する。なお、それらの図に示す各処理は、ＣＰＵ１がＲＯＭ２に格納されたプログラムを実行することで実現される。

【００３０】

図３は、全体処理のフローチャートである。始めに図３を参照して、全体処理について詳細に説明する。

電源がオンされると、先ず、ステップＳＡ１で初期処理を実行し、電子楽器を所定の状態に設定する。続くステップＳＡ２では、曲指定スイッチがオンしたか否か判定する。そのスイッチをユーザが操作したことで入力部４からその旨を示す信号を受け取った場合、判定はＹＥＳとなり、次にステップＳＡ３において、ユーザの入力部４への操作により指定される曲（演奏データ）を外部記憶装置６から取り込むための処理を実行してからステップＳＡ４に移行する。演奏データの取り込みを行う際、音符データＭＥの個数をカウントして、そのカウント値－１の値を変数ｅｖ＿ｍａｘに代入する。一方、そうでない場合には、判定はＮＯとなり、次にそのステップＳＡ４の処理を実行する。

【００３１】

ステップＳＡ４では、ステップＳＡ３で取り込まれる演奏データのなかで運指情報の生成の対象となる範囲をユーザに設定させるための範囲設定処理を実行する。次のステップＳＡ５では、運指生成スイッチがオンしたか否か判定する。そのスイッチをユーザが操作した場合、判定はＹＥＳとなり、次にステップＳＡ６で運指情報の生成を行うための運指生成処理を実行した後、ステップＳＡ７に移行する。そうでない場合には、次にそのステップＳＡ７に移行する。

【００３２】

ステップＳＡ７では、入力部４に設けられたその他のスイッチへの操作に対応するためのその他スイッチ処理を実行する。続くステップＳＡ８では、鍵盤７へのユーザの操作に応じて楽音を発音させるための鍵盤処理を実行する。その次のステップＳＡ９では、ユーザに曲を指定させ、指定された曲の自動演奏を実現させるための自動演奏処理を実行する。それ以降は、ステップＳＡ１０で表示部５に表示させるべき情報を表示させる表示処理、ステップＳＡ１１でその他処理を実行してから、上記ステップＳＡ２に戻る。

【００３３】

図４以降に示すフローチャートは、上記全体処理内で直接的、或いは間接的に呼び出されるサブルーチン処理を示すものである。このことから以降、そのサブルーチン処理について詳細に説明する。

【００３４】

図４は、上記ステップＳＡ４として実行される範囲設定処理のフローチャートである。次に図４を参照して、その設定処理について詳細に説明する。

先ず、ステップＳＣ１では、モード設定スイッチがオンしたか否か判定する。そのスイッチをユーザが操作した場合、判定はＹＥＳとなり、次のステップＳＣ２で変数ｍｏｄｅに値の代入を行ってからステップＳＣ７に移行する。そうでない場合には、判定はＮＯとなってステップＳＣ３に移行する。

【００３５】

上記指定区間モード、全体モードの間の切り替えは、例えばモード設定スイッチが操作される度に行うようになっている。そのようにモード間の切り替えが行われる場合、ステップＳＣ２で変数ｍｏｄｅに代入される値は、それまでの値が０であれば１、それまでの値が１であれば０、である。

【００３６】

ステップＳＣ３では、範囲設定スイッチがオンしたか否か判定する。そのスイッチをユーザが操作した場合、判定はＹＥＳとなり、ステップＳＣ４で変数ｅｖ＿ｓ、或いはｅｖ＿ｅへの値の代入をユーザの入力部４への操作に応じて行ってからステップＳＣ７に移行する。そうでない場合には、判定はＮＯとなってステップＳＣ５に移行する。

【００３７】
運指情報の生成対象とする区間（範囲）は、特には図示していないが、例えば先頭からの音符数で指定するようになっている。その指定は、数値を直接、入力させることで行わせても良いが、演奏データから楽譜を生成して表示部５に表示させることにより、表示部５上で指定できるようにさせても良い。当然のことながら、それら以外の方法を採用しても良い。
【００３８】
ステップＳＣ５では、オプション設定スイッチがオンしたか否か判定する。そのスイッチをユーザが操作した場合、判定はＹＥＳとなり、ステップＳＣ６で変数ｌｏｏｋｕｐｂ、及びｌｏｏｋｕｐｆのうちの少なくとも一方の値を変更させてから、ステップＳＣ７に移行する。そうでない場合には、判定はＮＯとなってステップＳＣ１１に移行する。
【００３９】
ステップＳＣ６では、変数ｌｏｏｋｕｐｂ、及びｌｏｏｋｕｐｆの値は、例えばオプション設定スイッチが操作される度に、（０，０）（前者は変数ｌｏｏｋｕｐｂに代入される値、後者は変数ｌｏｏｋｕｐｆに代入される値。以下、同様）→（０，１）→（１，０）→（１，１）→（０，０）の順序でサイクリックに変化させるようになっている。それにより、オプション設定スイッチを操作することで所望のオプションを設定できるようにさせている。
【００４０】
ステップＳＣ７では、変数ｍｏｄｅの値が０か否か判定する。その値が０、つまり全体モードが設定されている場合、判定はＹＥＳとなり、ステップＳＣ８で変数ｍｅ＿ｓに０、変数ｍｅ＿ｅに変数ｅｖ＿ｍａｘの値を代入した後、一連の処理を終了する。そうでない場合には、判定はＮＯとなってステップＳＣ９に移行する。
【００４１】
ステップＳＣ９では、変数ｍｅ＿ｓに、変数ｌｏｏｋｕｐｂの値が０ならば変数ｅｖ＿ｓの値、その変数ｌｏｏｋｕｐｂの値が１ならば変数ｅｖ＿ｓの値から定数ＣＯＳＴＳＥＥＫＣＮＴを引いた値（その値が０より小さければ０）を代入する。次のステップＳＣ１０では、変数ｍｅ＿ｅに、変数ｌｏｏｋｕｐｆの値が０ならば変数ｅｖ＿ｅの値、その変数ｌｏｏｋｕｐｆの値が１ならば変数ｅｖ＿ｅの値に定数ＣＯＳＴＳＥＥＫＣＮＴを加算した値（その値が変数ｅｖ＿ｍａｘの値より大きければ変数ｅｖ＿ｍａｘの値）を代入する。一連の処理はその後に終了する。
【００４２】
このようにして結果的に、運指情報の生成の対象となる区間範囲を示す値は変数ｍｅ＿ｓ、ｍｅ＿ｅに代入される。それにより、それらの変数により指定される範囲（区間）を対象にして運指情報の生成を行うようにしている。上記定数ＣＯＳＴＳＥＥＫＣＮＴはＣＰＵ１が実行するプログラム内で定義された変数か、或いは制御用データとしてＲＯＭ２に格納されたデータである。これは他の定数も同様である。
【００４３】
一方、上記ステップＳＣ５の判定がＮＯとなって移行するステップＳＣ１１では、同一フレーズ補正スイッチがオンしたか否か判定する。そのスイッチをユーザが操作した場合、判定はＹＥＳとなり、ステップＳＣ１２で変数ｐｈｒｃｏｒに値の代入を行った後、一連の処理を終了する。そうでない場合には、判定はＮＯとなり、ここで一連の処理を終了する。
【００４４】
上記ステップＳＣ１２での変数ｐｈｒｃｏｒへの値の代入は、例えばそれまでの値が０であれば１、それまでの値が１であれば０、を代入することで行うようになっている。それにより、内容が同一のフレーズを考慮した運指情報の補正を行うか否かを同一フレーズ補正スイッチへの操作により選択できるようにさせている。
【００４５】
図５は、図３に示す全体処理内でステップＳＡ６として実行される運指生成処理のフロ

ーチャートである。次に図５を参照して、その生成処理について詳細に説明する。
先ず、ステップＳＢ１では、変数ｌｏｏｐｃｎｔへの０の代入を含め、各種変数の初期化を行う。次に移行するステップＳＢ２では、変数ｍｅ＿ｓ、ｍｅ＿ｅにそれぞれ代入された値で指定された範囲を対象にして運指情報を生成する指定区間運指生成処理を実行する。その実行後はステップＳＢ３に移行する。

【００４６】

ステップＳＢ３では、ステップＳＢ２で生成される運指情報に従った演奏の難易度を評価するための評価処理を実行する。次のステップＳＢ４では、変数ｌｏｏｐｃｎｔの値が０でないか否か判定する。評価処理では、後述するように、難易度として、１音符平均の難易度（コスト）を算出し、算出した難易度が判定用の定数ＴＨＲＣＯＳＴ２より小さければ、つまり生成された運指情報に従った演奏の難易度が許容範囲内と云えるのであれば、その運指情報は適切なものであるとして変数ｌｏｏｐｃｎｔに０を代入するようにしている。このことから、直前のステップＳＢ２の実行により生成された運指情報が適切なものであった場合、判定はＮＯとなり、ここで一連の処理は終了する。そうでない場合には、判定はＹＥＳとなって上記ステップＳＢ２に戻る。

【００４７】

上記指定区間運指生成処理では、変数ｌｏｏｐｃｎｔの値が０であれば運指情報の生成を行い、０以外の値であれば、その値に応じた運指情報に対する修正を行うようになっている。それにより、ステップＳＢ４の判定がＹＥＳとなってステップＳＢ２に戻った場合には、運指情報をより適切なものとするための修正を行うようにさせている。

【００４８】

図６、及び図７は、そのステップＳＢ２として実行される指定区間運指生成処理のフローチャートである。次に図６、及び図７を参照して、その生成処理について詳細に説明する。

【００４９】

先ず、ステップＳＪ１では、変数ｌｏｏｐｃｎｔの値が０か否か判定する。その値が０であった場合、判定はＹＥＳとなり、次にステップＳＪ２で１音符毎に運指情報を生成する逐次生成処理を実行し、その次のステップＳＪ３で変数ｍｅ＿ｔａｉｌに変数ｍｅ＿ｅの値、変数ｌｏｏｐｃｎｔに１をそれぞれ代入した後、一連の処理を終了する。そうでない場合には、判定はＮＯとなってステップＳＪ４に移行する。

【００５０】

ステップＳＪ４では、変数ｌｏｏｐｃｎｔの値が１か否か判定する。その値が１でなかった場合、判定はＮＯとなってステップＳＪ９に移行する。そうでない場合には、判定はＹＥＳとなってステップＳＪ５に移行する。

【００５１】

ステップＳＪ５では、区間内の演奏データが示す演奏をフレーズで仮想的に分割するためのフレーズ分割処理を実行する。続くステップＳＪ６では、ステップＳＪ５でのフレーズ分割処理の実行で抽出される１フレーズ分の演奏データを対象に運指のパターンを適用するためのパターン適用処理を実行する。ステップＳＪ７にはその後に移行する。

【００５２】

上記フレーズ分割処理の実行により、詳細は後述するように、分割・抽出したフレーズの最後に位置する音符（データ）のインデックス値が変数ｍｅ＿ｔａｉｌに代入される。ステップＳＪ７では、その変数ｍｅ＿ｔａｉｌの値が変数ｍｅ＿ｅの値と等しいか否か判定する。区間内に存在する最後のフレーズの分割・抽出が終了した場合、変数ｍｅ＿ｔａｉｌには変数ｍｅ＿ｅの値が代入されることから、判定はＹＥＳとなり、ステップＳＪ８で変数ｌｏｏｐｃｎｔに２を代入した後、一連の処理を終了する。そうでない場合には、判定はＮＯとなり、ここで一連の処理を終了する。

【００５３】

上記ステップＳＪ４の判定がＮＯとなって移行するステップＳＪ９では、変数ｌｏｏｐｃｎｔの値が２か否か判定する。その値が２であった場合、判定はＹＥＳとなってステッ

プＳＪ１０に移行し、そうでない場合には、判定はＮＯとなって図７のステップＳＪ１４に移行する。

【００５４】

ステップＳＪ１０では、区間内の演奏データが示す演奏で発音される和音を判定する和音判定処理を実行する。続くステップＳＪ１１では、その区間内における全ての運指の組み合わせを調査するための和音全数検査処理を実行する。ステップＳＪ１２にはその後に移行する。

【００５５】

上記和音全数検査処理の実行により、詳細は後述するように、変数ｍｅ＿ｔａｉｌには変数ｍｅ＿ｅの値、または判定された和音の最後に位置する音符（データ）のインデックス値が代入される。ステップＳＪ１２では、その変数ｍｅ＿ｔａｉｌの値が変数ｍｅ＿ｅの値と等しいか否か判定する。それらが一致する場合、判定はＹＥＳとなり、ステップＳＪ１３で変数ｌｏｏｐｃｎｔに３を代入した後、一連の処理を終了する。そうでない場合には、判定はＮＯとなり、ここで一連の処理を終了する。

【００５６】

上記ステップＳＪ９の判定がＮＯとなって移行する図７のステップＳＪ１４では、変数ｌｏｏｐｃｎｔの値が３か否か判定する。その値が３であった場合、判定はＹＥＳとなってステップＳＪ１５に移行し、そうでない場合には、判定はＮＯとなってステップＳＪ１９に移行する。

【００５７】

ステップＳＪ１５では、区間内の演奏データが示す演奏で発音される分散和音を判定する分散和音判定処理を実行する。続くステップＳＪ１６では、その区間内における全ての運指の組み合わせを調査するための分散和音全数検査処理を実行する。ステップＳＪ１７にはその後に移行する。

【００５８】

上記分散和音全数検査処理の実行により、上記和音全数検査処理の実行時と同様に、変数ｍｅ＿ｔａｉｌには変数ｍｅ＿ｅの値、または判定された分散和音の最後に位置する音符（データ）のインデックス値が代入される。ステップＳＪ１７では、その変数ｍｅ＿ｔａｉｌの値が変数ｍｅ＿ｅの値と等しいか否か判定する。それらが一致する場合、判定はＹＥＳとなり、ステップＳＪ１８で変数ｌｏｏｐｃｎｔに４を代入した後、一連の処理を終了する。そうでない場合には、判定はＮＯとなり、ここで一連の処理を終了する。

【００５９】

ステップＳＪ１９では、変数ｌｏｏｐｃｎｔの値が４か否か判定する。その値が４であった場合、判定はＹＥＳとなってステップＳＪ２０に移行し、そうでない場合には、判定はＮＯとなってステップＳＪ２４に移行する。

【００６０】

ステップＳＪ２０では、運指コストが閾値として定めた定数ＴＨＲＣＯＳＴ１よりも大きな音符を抽出する検索を行う修正個所検索処理を実行する。次のステップＳＪ２１では、その処理の実行により抽出される修正対象範囲内における全ての運指の組み合わせを調査するための全数検査処理を実行する。ステップＳＪ２２にはその後に移行する。

【００６１】

上記全数検査処理の実行により、他の全数検査処理の実行時と同様に、変数ｍｅ＿ｔａｉｌには変数ｍｅ＿ｅの値、または抽出された修正対象範囲の最後に位置する音符（データ）のインデックス値が代入される。ステップＳＪ２２では、その変数ｍｅ＿ｔａｉｌの値が変数ｍｅ＿ｅの値と等しいか否か判定する。それらが一致する場合、判定はＹＥＳとなり、ステップＳＪ２３で変数ｌｏｏｐｃｎｔに５を代入した後、一連の処理を終了する。そうでない場合には、判定はＮＯとなり、ここで一連の処理を終了する。

【００６２】

ステップＳＪ２４では、変数ｌｏｏｐｃｎｔの値が５、且つ変数ｐｈｒｃｏｒの値が１か否か判定する。変数ｌｏｏｐｃｎｔの値が５，且つ変数ｐｈｒｃｏｒの値が１であった

場合、判定はＹＥＳとなり、ステップＳＪ２５で同一フレーズ補正処理を実行し、更にステップＳＪ２６で変数ｌｏｏｐｃｎｔに０を代入した後、一連の処理を終了する。そうでない場合には、判定はＮＯとなり、次にステップＳＪ２６の処理を実行する。

【００６３】

図５に示す運指生成処理内でステップＳＢ２として実行される指定区間運指生成処理では、上述したように、実行する度に変数ｌｏｏｐｃｎｔの値がインクリメントされる。それにより、最初に生成した運指情報に対し、変数ｌｏｏｐｃｎｔの値に対応する内容に着目した補正を適切な運指情報が生成されるか、或いは着目する内容が無くなるまで順次、行うようにしている。そのようにして、適切（最適）な運指情報を生成するうえで必要な補正を必要に応じて行うようにすることにより、適切（最適）な運指情報を演奏データが示す演奏内容に応じて確実、且つより迅速に生成できるようになる。

【００６４】

次に、上記指定区間運指生成処理内で実行されるサブルーチン処理について詳細に説明する。

図８は、上記ステップＳＪ２として実行される逐次生成処理のフローチャートである。そのサブルーチン処理としては、始めに図８を参照して、逐次生成処理について詳細に説明する。その逐次生成処理は、区間内に発音される音符別に運指情報を生成していく処理である。

【００６５】

新たな押鍵に使うべき指として、その押鍵時に別の鍵の押鍵に使用中の指を割り当てることはできない。新たな押鍵に使うべき指はそのときの状況によって制限される。このことから、本実施の形態では、図９に示すように、手の状態管理用変数を指別に用意している。

【００６６】

その図９において、括弧を付して表記の「ｈｆｉｇ」が指別にその状態を管理するためのデータ（以降「指状態データ」と呼ぶ）を示している。括弧内の数字は対応する指の番号（図２（ａ）に示すｆｉｇと同様）を示している。その指状態データは、「ｓｔａｔｕｓ」と表記の押鍵状態フラグ、「ｅｖｅｎｔ」と表記の押鍵イベントのインデックス（値）（そのイベントを表す音符データＭＥのインデックス）、「ｐｏｓｘ」と表記の指の位置座標、から構成されている。

【００６７】

上記押鍵状態フラグとしては、押鍵中でない指では０、押鍵中の指では１を割り当てるようにしている。上記位置座標は、図２（ａ）に示すデータｐｏｓｘと同じく鍵盤座標で表している。逐次生成処理では、指状態データ（手の状態管理用変数の代入値）を更新しつつ、運指情報を生成する。指状態データを構成するデータが代入される変数は、これまでと同様に、「変数ｓｔａｔｕｓ」といったように図中に表記のシンボルを付して表記する。

【００６８】

先ず、ステップＳＫ１では、変数ｍｅに変数ｍｅ＿ｓの値、変数ｐに、変数ｍｅの値で指定される音符データＭＥのデータｐｏｓｘ（図中「ＭＥ［ｍｅ］．ｐｏｓｘ」と表記。以降、その表記法も併せて用いることとする）をそれぞれ代入し、各指の指状態データの更新を行う。その更新は、具体的には各変数ｓｔａｔｕｓ（図中「ｈｆｉｇ［ｎ］．ｓｔａｔｕｓ」（ｎ＝０～４）と表記）に０をそれぞれ代入し、各変数ｐｏｓｘに親指から、変数ｐの値から４を減算した値、変数ｐの値から２を減算した値、変数ｐの値、変数ｐの値に２を加算した値、変数ｐの値に４を加算した値、をそれぞれ代入する。

【００６９】

ステップＳＫ１に続くステップＳＫ２では、変数ｍｅの値で指定される音符データＭＥを処理する時点での各指の状態を確認するための指状態確認処理を実行する。その次に移行するステップＳＫ３では、その音符データＭＥのデータｐｉｔｃｈで指定される鍵の押鍵に使うべき指を割り当てる指割り当て処理を実行する。その後はステップＳＫ４に移行

して、変数ｍｅの値が変数ｍｅ＿ｅの値と等しいか否か判定する。それらの値が等しい場合、判定はＹＥＳとなり、ここで一連の処理を終了する。そうでない場合には、判定はＮＯとなり、ステップＳＫ５で変数ｍｅの値をインクリメントしてから上記ステップＳＫ２に戻る。それにより、変数ｍｅの値を順次、インクリメントしながら、変数ｍｅの値で指定される音符データＭＥに従った押鍵に使うべき指の割り当てを行う。

【００７０】

次に、上記ステップＳＫ２として実行される指状態確認処理について、図１０に示すそのフローチャートを参照して詳細に説明する。

先ず、ステップＳＬ１では、着目する指を管理するための変数ｎに０、指の位置座標の最小値、最大値を管理するための変数ｐｏｓｍｉｎ、及びｐｏｓｍａｘにそれぞれ－１を代入する。続くステップＳＬ２では、変数ｎの値が５より小さいか否か判定する。その値が５より小さい場合、判定はＹＥＳとなってステップＳＬ３に移行する。そうでない場合には、判定はＮＯとなり、ここで一連の処理を終了する。

【００７１】

ステップＳＬ３では、変数ｎの値で指定される指状態データを構成する変数ｓｔａｔｕｓ（図中「ｈｆｉｇ［ｎ］．ｓｔａｔｕｓ」と表記）の値が１か否か判定する。変数ｎの値に対応する指が押鍵中である場合、その値は１であることから、判定はＹＥＳとなってステップＳＬ４に移行し、そうでない場合には、判定はＮＯとなってステップＳＬ７に移行する。

【００７２】

ステップＳＬ４では、変数ｅｖに、変数ｎの値で指定される指状態データを構成する変数ｅｖｅｎｔ（図中「ｈｆｉｇ［ｎ］．ｅｖｅｎｔ」と表記）の値を代入する。その代入後に移行するステップＳＬ５では、変数ｅｖの値で指定される音符データＭＥを構成する、データｔｉｍｅにデータｇａｔｅを加算して得られる値が、変数ｍｅの値で指定される音符データＭＥを構成するデータｔｉｍｅの値より小さいか否か判定する。変数ｍｅの値で指定される音符データＭＥに従った押鍵開始時に、変数ｅｖの値で指定される音符データＭＥに従った押鍵が終了している場合、前者は後者より小さくなる。このことから、そのような場合、判定はＹＥＳとなり、ステップＳＬ６において変数ｎの値で指定される指状態データを構成する変数ｓｔａｔｕｓ（図中「ｈｆｉｇ［ｎ］．ｓｔａｔｕｓ」と表記）の値として０を代入してからステップＳＬ７に移行する。そうでない場合には、判定はＮＯとなり、次にそのステップＳＬ７に移行する。

【００７３】

ステップＳＬ７では、変数ｐｏｓｍｉｎの値が－１、またはｈｆｉｇ［ｎ］．ｐｏｓｘの値より大きいならば、その変数ｐｏｓｍｉｎにｈｆｉｇ［ｎ］．ｐｏｓｘの値を代入する。続くステップＳＬ８では、変数ｐｏｓｍａｘの値が－１、またはｈｆｉｇ［ｎ］．ｐｏｓｘの値より小さいならば、その変数ｐｏｓｍａｘにｈｆｉｇ［ｎ］．ｐｏｓｘの値を代入する。その後は、ステップＳＬ９で変数ｎの値をインクリメントしてから上記ステップＳＬ２に戻る。

【００７４】

このようにして、指状態確認処理を実行することにより、変数ｍｅの値で指定される音符データＭＥに従った押鍵開始時に離鍵中とすべき指は離鍵中に設定され、その押鍵開始時に位置座標が最小値、及び最大値となる２つの指の座標値は変数ｐｏｓｍｉｎ、ｐｏｓｍａｘにそれぞれ代入される。

【００７５】

図１１は、図８に示す逐次生成処理内でステップＳＫ３として実行される指割り当て処理のフローチャートである。次に図１１を参照して、その割り当て処理について詳細に説明する。その割り当て処理では、押鍵に使われていない指を探しだし、着目する音符データＭＥに従って押鍵すべき鍵の座標値よりも小さい座標値に位置している指を優先的に押鍵に使うべき指として割り当てるようにしている。

【００７６】

先ず、ステップＳＭ１では、変数ｎに０、変数ｉｆｉｇに－１をそれぞれ代入する。続くステップＳＭ２では、変数ｎの値が５より小さいか否か判定する。その値が５より小さい場合、判定はＹＥＳとなってステップＳＭ３に移行し、そうでない場合には、判定はＮＯとなってステップＳＭ９に移行する。

【００７７】

ステップＳＭ３では、ｈｆｉｇ［ｎ］．ｓｔａｔｕｓの値が０か否か判定する。変数ｎの値で指定される指が押鍵に使われていない場合、その値は０であることから、判定はＹＥＳとなり、ステップＳＭ５で変数ｉｆｉｇに変数ｎの値を代入してからステップＳＭ６に移行する。そうでない場合には、判定はＮＯとなり、ステップＳＭ４で変数ｎの値をインクリメントしてから上記ステップＳＭ２に戻る。

【００７８】

ステップＳＭ６では、ｈｆｉｇ［ｎ］．ｐｏｓｘの値がＭＥ［ｍｅ］．ｐｏｓｘの値以下か否か判定する。変数ｎの値で指定される指の座標値が変数ｍｅの値で指定される音符データＭＥに従って押鍵すべき鍵の座標値以下であった場合、判定はＹＥＳとなってステップＳＭ７に移行する。そうでない場合には、判定はＮＯとなり、ステップＳＭ４で変数ｎの値のインクリメントを行う。そしてステップＳＭ２の判定がＮＯになった場合は、ステップＳＭ９に移行する。ステップＳＭ９では、変数ｉｆｉｇが－１か否か判定する。変数ｉｆｉｇが－１の場合は、判定がＹＥＳとなり、ステップＳＭ１０で変数ｉｆｉｇに０を代入してからステップＳＭ７に移行する。そうでない場合には、判定はＮＯとなってステップＳＭ７に移行する。

【００７９】

ステップＳＭ７では、ｈｆｉｇ［ｉｆｉｇ］．ｓｔａｔｕｓに１、ｈｆｉｇ［ｉｆｉｇ］．ｅｖｅｎｔに変数ｍｅの値、ｈｆｉｇ［ｉｆｉｇ］．ｐｏｓｘにＭＥ［ｍｅ］．ｐｏｓｘの値、ＭＥ［ｍｅ］．ｆｉｇに変数ｉｆｉｇの値、をそれぞれ代入する。次のステップＳＭ８では、押鍵に割り当てた指以外の指の座標値を指定する指位置指定処理を実行する。その実行後、一連の処理を終了する。

【００８０】

図１２は、その指位置指定処理のフローチャートである。次にその指定処理について、図１２を参照して詳細に説明する。その指定処理では、全ての指を対象に位置させるべき座標値の基準からの変位量を算出して配列変数ｆｉｇｔｍｐの対応する要素に代入し、押鍵を割り当てた指以外の指を位置させるべき座標値として、対応する要素に代入した座標値から算出される値を設定するようにしている。

【００８１】

先ず、ステップＳＮ１では、変数ｎに０、変数ｒａｎｇｅに、変数ｐｏｓｍａｘの値から変数ｐｏｓｍｉｎの値を減算した値を代入する。次のステップＳＮ２では、変数ｎの値が５より小さいか否か判定する。その値が５より小さい場合、判定はＹＥＳとなってステップＳＮ３に移行し、そうでない場合には、判定はＮＯとなってステップＳＮ５に移行する。

【００８２】

ステップＳＮ３では、ｈｆｉｇ［ｎ］．ｐｏｓｘの値からｈｆｉｇ［０］．ｐｏｓｘの値を減算した値を、ｈｆｉｇ［４］．ｐｏｓｘの値からｈｆｉｇ［０］．ｐｏｓｘの値を減算した値で除算し、その除算値を変数ｒａｎｇｅの値に乗算して得られる変位量を、配列変数ｆｉｇｔｍｐの変数ｎの値で指定される要素ｆｉｇｔｍｐ［ｎ］に代入する。その代入後は、ステップＳＮ４で変数ｎの値をインクリメントしてから上記ステップＳＮ２に戻る。

【００８３】

一方、ステップＳＮ５では、変数ｎに０を代入する。次のステップＳＮ６では、変数ｎの値が５より小さいか否か判定する。その値が５より小さい場合、判定はＹＥＳとなってステップＳＮ７に移行する。そうでない場合には、判定はＮＯとなり、ここで一連の処理を終了する。

【００８４】
ステップＳＮ７では、変数ｎの値が変数ｉｆｉｇの値と等しくないか否か判定する。それらの値が一致していた場合、判定はＮＯとなり、ステップＳＮ９で変数ｎの値をインクリメントしてからステップＳＮ６に戻る。そうでない場合には、判定はＹＥＳとなってステップＳＮ８に移行し、ｈｆｉｇ［ｎ］．ｐｏｓｘの値として、ｈｆｉｇ［ｉｆｉｇ］．ｐｏｓｘの値に、要素ｆｉｇｔｍｐ［ｎ］の値から要素ｆｉｇｔｍｐ［ｉｆｉｇ］の値を減算した値を代入する。その代入後は、ステップＳＮ９に移行して変数ｎの値をインクリメントする。
【００８５】
図６、及び図７に示す指定区間運指生成処理内で実行されるサブルーチン処理の説明に戻る。
図１３は、その生成処理内でステップＳＪ５として実行されるフレーズ分割処理のフローチャートである。次にその分割処理について、図１３を参照して詳細に説明する。その分割処理は、区間内の演奏をフレーズで仮想的に分割して１フレーズ分の演奏を抽出するための処理である。
【００８６】
先ず、ステップＳＦ１では、フレーズ分割の開始位置を管理するための変数ｍｅｆに、変数ｍｅ＿ｔａｉｌの値に１を加算した値（その加算値が変数ｍｅ＿ｅの値より大きければ変数ｍｅ＿ｓの値）を代入する。次のステップＳＦ２では、その終了位置として考えられる最大位置を管理するための変数ｍｅｄに、変数ｍｅｆの値に定数ＤＩＶＮＯＴＥＣＮＴを加算した値（その加算値が変数ｍｅ＿ｅの値より大きければ変数ｍｅ＿ｅの値）を代入する。その後は、ステップＳＦ３で変数ｍｅに変数ｍｅｆの値、変数ｍｅｔに変数ｍｅｄの値、変数ｎｏｔｅｏｎｉｎｔｖに定数ＴＨＲＩＮＴＶをそれぞれ代入してからステップＳＦ４に移行する。
【００８７】
本実施の形態では、フレーズ分割は隣接する２音間の発音開始時刻の時間差に着目して行っている。上記定数ＴＨＲＩＮＴＶはその時間差として考えられる最小の閾値として定めたものである。そのような定数ＴＨＲＩＮＴＶを用意したことにより、変数ｍｅｄの値で指定される音符までの演奏区間の間で、その定数ＴＨＲＩＮＴＶより長い時間差のなかで最大のものをフレーズの境界としてフレーズ分割を行うようにしている。そのフレーズ分割は、ステップＳＦ４以降の処理を実行することで実現される。
【００８８】
先ず、ステップＳＦ４では、変数ｍｅの値が変数ｍｅｄの値と等しいか否か判定する。それらの値が等しい場合、判定はＹＥＳとなってステップＳＦ８に移行し、そうでない場合には、判定はＮＯとなってステップＳＦ５に移行する。
【００８９】
ステップＳＦ５では、変数ｎｏｔｅｏｎｉｎｔｖの値が、ＭＥ［ｍｅ＋１］．ｔｉｍｅの値からＭＥ［ｍｅ］．ｔｉｍｅの値を減算した値より小さいか否か判定する。変数ｍｅの値で指定される音符とその次に発音される音符との間の発音開始時刻の時間差が変数ｎｏｔｅｏｎｉｎｔｖの値で表される時間差より大きい場合、判定はＹＥＳとなり、ステップＳＦ６において、変数ｍｅｔに変数ｍｅの値、変数ｎｏｔｅｏｎｉｎｔｖにＭＥ［ｍｅ＋１］．ｔｉｍｅの値からＭＥ［ｍｅ］．ｔｉｍｅの値を減算した値をそれぞれ代入し、更にステップＳＦ７で変数ｍｅの値をインクリメントしてから上記ステップＳＦ４に戻る。そうでない場合には、判定はＮＯとなって次にステップＳＦ７の処理を実行する。
【００９０】
そのステップＳＦ４の判定がＹＥＳとなって移行するステップＳＦ８では、変数ｍｅ＿ｔｏｐに変数ｍｅｆの値、変数ｍｅ＿ｔａｉｌに変数ｍｅｔの値をそれぞれ代入する。続くステップＳＦ９では、変数ｍｅｆの値と変数ｍｅｔの値で指定される区間内に存在する音符データＭＥ中のデータｐｉｄ、ｐｓｔａｔの更新を行う。その更新は、各データｐｉｄとして０をそれぞれ設定し、データｐｓｔａｔとしては、変数ｍｅｆの値で指定される

音符データＭＥのものには０、変数ｍｅｔの値で指定される音符データＭＥのものには２、それ以外の音符データＭＥのものには１をそれぞれ設定することで行う。一連の処理はその更新後に終了する。

【００９１】

上記ステップＳＦ８で値が代入される変数ｍｅ＿ｔｏｐ、ｍｅ＿ｔａｉｌは、分割により抽出されたフレーズの先頭位置、終了位置を音符データＭＥのインデックス値で表すものとして扱われる。フレーズ分割処理に続けて実行されるパターン適用処理では、それらのインデックス値で指定される区間として抽出された１フレーズに運指のパターンを適用させる。

【００９２】

図１４は、そのパターン適用処理のフローチャートである。次にその適用処理について、図１４を参照して詳細に説明する。

１フレーズ分の演奏に適用可能な運指のパターンを示すパターンデータは、ＲＯＭ２に制御用データとして格納されている。そのパターンデータは、例えば図１５に示すような構成でＲＯＭ２に格納されている。それにより、適用処理は、ＲＯＭ２に格納されたパターンデータを参照して行うようになっている。

【００９３】

図１５中に括弧を付して表記の「ＦＰ」はパターンデータを示している。それにより、ＲＯＭ２には、計ＦＰ＿Ｎ個のパターンデータが格納されている。括弧内の数値は、パターンデータＦＰの特定用に割り当てられた番号（パターン番号）を表している。

【００９４】

各パターンデータＦＰは、音符単位のデータ（以下「音符データ」と呼ぶ）が複数、まとめられて構成されている。図１５中に括弧を付して表記の「ｎｏｔｅ」はその音符データを表している。括弧内の数値は、パターンデータＦＰの先頭を基準にした音符データｎｏｔｅの位置を表している。音符データｎｏｔｅの指定はその数値により行われる。「ＦＰＮ＿Ｎ」は各パターンデータＦＰを構成する音符データ数に１を加算した値である。

【００９５】

音符データｎｏｔｅは、「ｂｗ」と表記の鍵盤種類、「ｐｉｎｔ」と表記の前音符との発音開始時刻の時間間隔、「ｎｉｎｔ」と表記の次音符との発音開始時刻の時間間隔、及び「ｆｉｇ」と表記の運指番号（押鍵に用いる指の番号）、を備えた構成となっている。鍵盤種類は、白健では０、黒鍵では１、最後の音符データでは－１で表している。

【００９６】

図１４に示す適用処理では先ず、ステップＳＱ１で変数ｐａｔに０を代入する。その変数ｐａｔは、注目するパターンデータＦＰを管理するためのものである。その変数ｐａｔに０を代入した後はステップＳＱ２に移行して、変数ｐａｔの値がパターンデータＦＰの総数ＦＰ＿Ｎより小さいか否か判定する。パターンの適用のために全てのパターンデータＦＰに注目した場合、変数ｐａｔの値は総数ＦＰ＿Ｎ以上となる。このことから、その場合、判定はＮＯとなり、ここで一連の処理を終了する。そうでない場合には、判定はＹＥＳとなってステップＳＱ３に移行する。

【００９７】

ステップＳＱ３では、変数ｍａｔｃｈに１を代入する。続くステップＳＱ４では、変数ｐａｔの値で指定されるパターンデータＦＰに着目したパターンマッチング処理を実行する。その後に移行するステップＳＱ５では、変数ｍａｔｃｈの値が０か否か判定する。その値が０であった場合、判定はＹＥＳとなり、ステップＳＱ６で変数ｐａｔの値をインクリメントしてから上記ステップＳＱ２に戻る。そうでない場合には、判定はＮＯとなり、ここで一連の処理を終了する。

【００９８】

図１６は、上記ステップＳＱ４として実行されるパターンマッチング処理のフローチャートである。ここで図１６を参照して、そのマッチング処理について詳細に説明する。そのマッチング処理は、変数ｐａｔの値で指定されるパターンデータＦＰ［ｐａｔ］と抽出

フレーズの間のマッチングを行い、それらの間の一致している部分を確認するための処理である。

【００９９】

パターンデータＦＰを構成する音符データｎｏｔｅは、鍵盤種類（押鍵すべき鍵の種類：データｂｗ）、前音符との発音開始時刻の時間間隔（データｐｉｎｔ）、次音符との発音開始時刻の時間間隔（データｎｉｎｔ）、及び運指番号（データｆｉｇ）、を備えた構成である。それにより、パターンマッチングは、パターンデータＦＰ［ｐａｔ］と抽出フレーズのそれぞれ先頭に位置している音符データ（ｎｏｔｅおよびＭＥ）から、押鍵していくべき鍵の種類の流れ、及びその押鍵間隔の許容範囲内での一致を確認していく形で行うようにしている。許容範囲内で一致するか否か判定するために、定数ＴＨＲＩＮＴＶ２を用意している。

【０１００】

先ず、ステップＳＲ１では、変数ｐｐに変数ｍｅ＿ｔｏｐの値、変数ｐｃｎｔに０をそれぞれ代入する。変数ｐｐは演奏データのなかで注目する音符データＭＥを管理するために用いられ、変数ｐｃｎｔはパターンデータＦＰ［ｐａｔ］のなかで注目する音符データｎｏｔｅの管理（及び一致している音符データの個数のカウント）に用いられる。それらの代入後に移行するステップＳＲ２では、変数ｐｐの値が変数ｍｅ＿ｔａｉｌの値より大きいか否か判定する。変数ｐｐの値が変数ｍｅ＿ｔａｉｌの値より大きい場合、判定はＹＥＳとなってステップＳＲ１０に移行し、そうでない場合には、判定はＮＯとなってステップＳＲ３に移行する。

【０１０１】

ステップＳＲ３では、各種変数への値の代入を行う。具体的には、変数ｆｐに、パターンデータＦＰ［ｐａｔ］のなかで変数ｐｃｎｔの値で指定される音符データ（図中「ＦＰ［ｐａｔ］．ｎｏｔｅ［ｐｃｎｔ］」と表記。以降、その表記法も用いる）を代入し、変数ｐｐの値が０であれば０、そうでなければ変数ｐｐで指定される音符とその前音符間の発音開始時刻の時間間隔（＝ＭＥ［ｐｐ］．ｔｉｍｅ－ＭＥ［ｐｐ－１］．ｔｉｍｅ）を変数ｐｉｎｔに代入し、変数ｐｐの値が変数ｅｖ＿ｍａｘの値と等しければ０、そうでなければ変数ｐｐで指定される音符とその次音符間の発音開始時刻の時間間隔（＝ＭＥ［ｐｐ＋１］．ｔｉｍｅ－ＭＥ［ｐｐ］．ｔｉｍｅ）を変数ｎｉｎｔに代入する。そのような代入を行った後、ステップＳＲ４に移行する。

【０１０２】

ステップＳＲ４では、変数ｆｐに代入されたＦＰ［ｐａｔ］．ｎｏｔｅ［ｐｃｎｔ］のなかのデータｂｗ（図中「ｆｐ．ｂｗ」と表記）の値が－１か否か判定する。そのデータｂｗの値が－１であった場合、判定はＹＥＳとなってステップＳＲ１０に移行し、そうでない場合には、判定はＮＯとなってステップＳＲ５に移行する。ここでのＹＥＳの判定は、着目するパターンデータＦＰ［ｐａｔ］を構成する音符データｎｏｔｅが全て抽出フレーム中の対応する音符データＭＥと一致判定条件（鍵盤種類が一致し、且つ前音符、及び次音符との発音開始時刻の時間間隔が許容範囲内でそれぞれ一致するという条件）を満たしていることを意味する。

【０１０３】

ステップＳＲ５では、ｆｐ．ｂｗの値がＭＥ［ｐｐ］．ｐｏｓｘを２で割った余り（図中「ＭＥ［ｐｐ］．ｐｏｓｘ　％２」と表記。以降、その表記法も用いる）と等しくないか否か判定する。音符データｎｏｔｅ［ｐｃｎｔ］中のデータｂｗが表す鍵の種類が音符データＭＥ［ｐｐ］中のデータｐｏｓｘで指定される鍵の種類と同じであった場合、それらは一致することから、判定はＮＯとなってステップＳＲ６に移行する。そうでない場合には、判定はＹＥＳとなり、ステップＳＲ９で変数ｍａｔｃｈに０を代入した後、一連の処理を終了する。

【０１０４】

ステップＳＲ６では、変数ｐｉｎｔの値からｆｐ．ｐｉｎｔの値を引いた値の絶対値が上記定数ＴＨＲＩＮＴＶ２より大きいか否か判定する。パターンデータＦＰと抽出フレー

ズ間で前音符との時間間隔が許容範囲内で一致しなかった場合、その関係が満たされることから、判定はＹＥＳとなってステップＳＲ９に移行し、そうでない場合には、判定はＮＯとなってステップＳＲ７に移行する。

【０１０５】

ステップＳＲ７では、変数ｎｉｎｔの値からｆｐ．ｎｉｎｔの値を引いた値の絶対値が上記定数ＴＨＲＩＮＴＶ２より大きいか否か判定する。パターンデータＦＰと抽出フレーズ間で次音符との時間間隔が許容範囲内で一致しなかった場合、その関係が満たされることから、判定はＹＥＳとなってステップＳＲ９に移行する。そうでない場合には、つまり一致判定条件が全て満たされている場合には、判定はＮＯとなり、ステップＳＲ８で変数ｐｐ、及びｐｃｎｔの値をそれぞれインクリメントしてから上記ステップＳＲ２に戻る。それにより、ステップＳＲ２でのＹＥＳの判定は、抽出フレーズを構成する音符データＭＥは全て、パターンデータＦＰ［ｐａｔ］を構成する音符データｎｏｔｅのなかで対応する音符データｎｏｔｅと一致判定条件を満たしていることを意味している。

【０１０６】

ステップＳＲ２、或いはＳＲ４の判定がＹＥＳとなって移行するステップＳＲ１０では、変数ｐｃｎｔの値が０か否か判定する。その値が０であった場合、判定はＹＥＳとなり、ここで一連の処理を終了する。そうでない場合には、判定はＮＯとなってステップＳＲ１１に移行する。

【０１０７】

ステップＳＲ１１以降では、パターンデータＦＰ［ｐａｔ］で定義された運指番号を抽出フレーズを構成する各音符データＭＥ中のデータｆｉｇとして設定するための処理が行われる。

【０１０８】

先ず、ステップＳＲ１１では、変数ｐｐに変数ｍｅ＿ｔｏｐの値、変数ｎに０をそれぞれ代入する。続くステップＳＲ１２では、ＭＥ［ｐｐ］．ｆｉｇとして、パターンデータＦＰ［ｐａｔ］の変数ｎの値で指定される音符データｎｏｔｅ［ｎ］中のデータｆｉｇ（図中「ＦＰ［ｐａｔ］．ｎｏｔｅ［ｎ］．ｆｉｇ」と表記）を設定し、変数ｎの値をインクリメントする。

【０１０９】

ステップＳＲ１２に続くステップＳＲ１３では、変数ｎの値が変数ｐｃｎｔの値より小さいか否か判定する。パターンデータＦＰ［ｐａｔ］で定義された運指番号の抽出フレーズを構成する各音符データＭＥ中のデータｆｉｇとしての設定が完了した場合、その関係は満たされなくなることから、判定はＮＯとなり、ここで一連の処理を終了する。そうでない場合には、判定はＹＥＳとなり、ステップＳＲ１４で変数ｐｐの値をインクリメントしてから上記ステップＳＲ１２に戻る。それにより、データｆｉｇの設定を継続させる。

【０１１０】

図１７は、図６、及び図７に示す指定区間運指生成処理内でステップＳＪ１０として実行される和音判定処理のフローチャートである。次にその判定処理について、図１７を参照して詳細に説明する。本実施の形態では、発音開始時刻が同じ（許容範囲内で一致する）３音以上の音符群を和音の構成音として検出するようにしている。

【０１１１】

先ず、ステップＳＥ１では、変数ｍｅ＿ｔａｉｌの値に１を加算した値が変数ｍｅ＿ｅを越えなければその加算値、それを越えれば変数ｍｅ＿ｓの値を変数ｍｅに代入する。次のステップＳＥ２では、変数ｍｅの値が、変数ｍｅ＿ｅの値から１を引いた値以上か否か判定する。変数ｍｅの値がその減算値以上であった場合、判定はＹＥＳとなり、ステップＳＥ３で変数ｍｅ＿ｔｏｐ、及びｍｅ＿ｔａｉｌに変数ｍｅ＿ｅの値を代入した後、一連の処理を終了する。そうでない場合には、判定はＮＯとなってステップＳＥ４に移行する。ここでのＹＥＳの判定は、和音の検出が全て終了したことを意味している。

【０１１２】

ステップＳＥ４では、ＭＥ［ｍｅ］．ｔｉｍｅの値がＭＥ［ｍｅ＋１］．ｔｉｍｅの値

と一致するか否か判定する。変数ｍｅの値で指定される音符とその次音符の発音開始時刻が一致している場合、判定はＹＥＳとなり、ステップＳＥ６で変数ｃｎｔに２、変数ｎｅに変数ｍｅの値に２を加算した値をそれぞれ代入してからステップＳＥ７に移行する。そうでない場合には、判定はＮＯとなり、ステップＳＥ５で変数ｍｅの値をインクリメントしてから上記ステップＳＥ２に戻る。

【０１１３】

ステップＳＥ７では、ＭＥ［ｎｅ］．ｔｉｍｅの値がＭＥ［ｍｅ］．ｔｉｍｅの値と一致するか否か判定する。変数ｍｅの値で指定される音符と変数ｎｅの値で指定される音符の発音開始時刻が一致している場合、判定はＹＥＳとなり、ステップＳＥ８で変数ｃｎｔ、及びｎｅの各値のインクリメントを行った後、ステップＳＥ９に移行する。そうでない場合には、判定はＮＯとなり、ステップＳＥ１０に移行する。

【０１１４】

ステップＳＥ９では、変数ｎｅの値が変数ｍｅ＿ｅの値より大きいか否か判定する。その大小関係が成立している場合、判定はＹＥＳとなってステップＳＥ１０に移行する。そうでない場合には、判定はＮＯとなり、上記ステップＳＥ７に戻る。それにより、変数ｍｅの値で指定される音符の発音開始時刻と発音開始時刻が一致しない音符が見つかるか（ステップＳＥ７のＮＯの判定）、或いは発音開始時刻を比較する対象となる音符が無くなるまで（ステップＳＥ９のＹＥＳの判定）、ステップＳＥ７～ＳＥ９で形成される処理ループを繰り返し実行する。

【０１１５】

ステップＳＥ１０では、変数ｃｎｔの値が３以上か否か判定する。その値が３、つまり発音開始時刻が一致すると見なす音符が３つ以上、見つかった場合、判定はＹＥＳとなり、ステップＳＥ１１で変数ｍｅの値を変数ｍｅ＿ｔｏｐ、変数ｃｎｔの値から１を引いた値に対し、変数ｍｅの値を加算した値を変数ｍｅ＿ｔａｉｌにそれぞれ代入した後、一連の処理を終了する。そうでない場合には、判定はＮＯとなり、ステップＳＥ１２で変数ｍｅにそれまでの値に変数ｃｎｔの値を加算した値を新たに代入してから上記ステップＳＥ２に戻る。それにより、和音の構成音の検出を継続させる。

【０１１６】

図６、及び図７に示す指定区間運指生成処理では、上記和音検出処理の実行後に移行するステップＳＪ１１で和音全数検索処理を実行する。次にその検索処理について、図１８に示すそのフローチャートを参照して詳細に説明する。

【０１１７】

先ず、ステップＳＰ１では、変数ｍｅ＿ｔｏｐの値が変数ｍｅ＿ｔａｉｌの値と等しいか否か判定する。上述したように、それらの変数には、ステップＳＥ２の判定がＹＥＳ、つまり和音の検出が全て終了した場合に同じ値が代入される。このことから、そのような場合、判定はＹＥＳとなり、ここで一連の処理を終了する。そうでない場合には、判定はＮＯとなってステップＳＰ２に移行し、全数検査処理を実行する。その後、一連の処理を終了する。

【０１１８】

次に上記全数検査処理について、図１９に示すそのフローチャートを参照して詳細に説明する。その全数検査処理は、図６、及び図７に示す指定区間運指生成処理内ではステップＳＪ２１として実行される。

【０１１９】

先ず、ステップＳＴ１では、指定の区間内の運指上のコストを算出する区間コスト算出処理を実行する。次のステップＳＴ２では、その算出処理の実行により変数ｃｏｓｔｔｍｐに代入されたコストを変数ｃｏｓｔｂｅｓｔ、変数ｍｅ＿ｔｏｐの値を変数ｍｅ、１を変数ｌｏｏｐｍａｘ、０を変数ｌｏｏｐにそれぞれ代入する。そして、変数ｍｅ＿ｔｏｐと変数ｍｅ＿ｔａｉｌで指定される区間の運指を、配列変数ｆｉｇｔｍｐにコピーする。そのコピーは、変数ｍｅに変数ｍｅ＿ｔｏｐの値から変数ｍｅ＿ｔａｉｌの値まで順次、代入しながら、その変数ｍｅの値で指定される音符データＭＥ［ｍｅ］のデータｆｉｇを

、初期値が０でその値から変数ｍｅの値と同じタイミングで順次インクリメントする変数ｃｎｔの値で指定される配列変数ｆｉｇｔｍｐ［ｃｎｔ］に設定（つまりｆｉｇｔｍｐ［ｃｎｔ］←ＭＥ［ｍｅ］．ｆｉｇ）することで行われる。そして全ての運指のコピーが終了した後で、変数ｍｅ＿ｔｏｐの値を改めて変数ｍｅに代入する。ステップＳＴ３にはその後に移行する。

【０１２０】

ステップＳＴ３～ＳＴ５では、変数ｌｏｏｐｍａｘに、区間内における運指の組み合わせ総数、つまりその区間内における全ての運指の組み合わせの数を求めて代入するための処理が行われる。

【０１２１】

先ず、ステップＳＴ３では、変数ｌｏｏｐｍａｘに、それまでの値に５を掛けた値を新たに代入する。続くステップＳＴ４では、変数ｍｅの値が変数ｍｅ＿ｔａｉｌの値と等しいか否か判定する。それらの値が一致した場合、判定はＹＥＳとなってステップＳＴ６に移行する。そうでない場合には、判定はＮＯとなり、ステップＳＴ５で変数ｍｅの値をインクリメントしてから上記ステップＳＴ３に戻る。それにより、ステップＳＴ４の判定がＹＥＳとなった場合、変数ｌｏｏｐｍａｘには組み合わせ総数が代入されていることになる。

【０１２２】

ステップＳＴ６では、変数ｌｏｏｐの値が変数ｌｏｏｐｍａｘの値より小さいか否か判定する。その大小関係が成立している場合、判定はＹＥＳとなり、ステップＳＴ７に移行する。そうでない場合には、判定はＮＯとなり、ステップＳＴ１８に移行する。ステップＳＴ１８では、変数ｍｅ＿ｔｏｐと変数ｍｅ＿ｔａｉｌで指定される区間の運指に、配列変数ｆｉｇｔｍｐの運指をコピーする。そのコピーは、変数ｍｅに変数ｍｅ＿ｔｏｐの値から変数ｍｅ＿ｔａｉｌの値まで順次、代入しながら、その変数ｍｅの値で指定される音符データＭＥ［ｍｅ］のデータｆｉｇに、初期値が０でその値から変数ｍｅの値と同じタイミングで順次インクリメントする変数ｃｎｔの値で指定される配列変数ｆｉｇｔｍｐ［ｃｎｔ］の運指を設定（つまりＭＥ［ｍｅ］．ｆｉｇ←ｆｉｇｔｍｐ［ｃｎｔ］）することで行われる。その後、一連の処理を終了する。

【０１２３】

ステップＳＴ７では、変数ｍｅに変数ｍｅ＿ｔｏｐの値、変数ｄｉｇｉｔに１、変数ｃｎｔに０をそれぞれ代入する。次のステップＳＴ８では、音符データＭＥの変数ｍｅの値で指定される要素ＭＥ［ｍｅ］のデータｆｉｇに、変数ｌｏｏｐの値を変数ｄｉｇｉｔの値で割った除算値を５で割って得られる余り（図中「（ｌｏｏｐ／ｄｉｇｉｔ）％５」と表記）を代入し、その代入後には変数ｄｉｇｉｔにそれまでの値に５を掛けた値を代入し、変数ｃｎｔの値をインクリメントする。その後に移行するステップＳＴ９では、変数ｍｅの値が変数ｍｅ＿ｔａｉｌの値と等しいか否か判定する。それらの値が等しい場合、判定はＹＥＳとなってステップＳＴ１１に移行する。そうでない場合には、判定はＮＯとなり、ステップＳＴ１０で変数ｍｅの値をインクリメントしてから上記ステップＳＴ８に戻る。

【０１２４】

上記ステップＳＴ８～ＳＴ１０で形成される処理ループをステップＳＴ９の判定がＹＥＳとなるまで繰り返し実行することにより、要素ＭＥ［ｍｅ］．ｆｉｇには順次、運指番号が代入される。その運指番号は、変数ｌｏｏｐの値を固定にして変数ｄｉｇｉｔの値のみを随時それまでの５倍とすることにより、変数ｌｏｏｐの値に応じて固有に変化していく数値となる。

【０１２５】

ステップＳＴ１１では、音符データＭＥに代入した運指番号に従った運指上のコストを算出するために、区間コスト算出処理を実行する。その実行後はステップＳＴ１２に移行する。

【０１２６】

ステップＳＴ１２では、変数ｃｏｓｔｂｅｓｔの値が変数ｃｏｓｔｔｍｐの値より大きいか否か判定する。変数ｃｏｓｔｔｍｐに代入されるコストは、その値が大きくなるほど難易度が高いことを示している。それにより、今回、音符データＭＥの各要素に代入していった運指番号の組み合わせから求めたコストがそれまでのものより小さい場合、判定はＹＥＳとなってステップＳＴ１３に移行し、そうでない場合には、判定はＮＯとなってステップＳＴ１７に移行する。

【０１２７】

ステップＳＴ１３では、変数ｃｏｓｔｂｅｓｔに変数ｃｏｓｔｔｍｐの値、変数ｍｅに変数ｍｅ＿ｔｏｐの値、変数ｃｎｔに０をそれぞれ代入する。次のステップＳＴ１４では、ＭＥ［ｍｅ］．ｆｉｇの値を要素ｆｉｇｔｍｐ［ｃｎｔ］のデータとして設定し、その設定後に変数ｃｎｔの値をインクリメントする。その後に移行するステップＳＴ１５では、変数ｍｅの値が変数ｍｅ＿ｔａｉｌの値と等しいか否か判定する。それらの値が一致する場合、判定はＹＥＳとなり、ステップＳＴ１７で変数ｌｏｏｐの値をインクリメントしてから上記ステップＳＴ６に戻る。そうでない場合には、判定はＮＯとなり、ステップＳＴ１６で変数ｍｅの値をインクリメントしてから上記ステップＳＴ１４に戻る。それにより、ステップＳＴ６の判定がＮＯとなった時点では、コストが最も低い運指（指の割り当て）の組み合わせが配列変数ｆｉｇｔｍｐに設定されていることになる。

【０１２８】

図２０は、上記ステップＳＴ１、或いはＳＴ１１で実行される区間コスト算出処理のフローチャートである。次にその算出処理について、図２０に示すそのフローチャートを参照して詳細に説明する。

【０１２９】

先ず、ステップＳＵ１では、変数ｃｏｓｔｔｍｐに０、変数ｍｅに変数ｍｅ＿ｔｏｐの値をそれぞれ代入する。続くステップＳＵ２では、変数ｍｅの値が変数ｍｅ＿ｔａｉｌの値より大きいか否か判定する。前者が後者より大きい場合、判定はＹＥＳとなり、ステップＳＵ６で変数ｃｏｓｔｔｍｐに、それまでの値を音符の総数（＝ｍｅ＿ｔａｉｌ－ｍｅ＿ｔｏｐ＋１）で割った値、つまり１音符当たりの平均コストを代入した後、一連の処理を終了する。そうでない場合には、判定はＮＯとなってステップＳＵ３に移行する。

【０１３０】

ステップＳＵ３では、０から定数ＣＯＳＴＳＥＥＫＣＮＴ未満までの変数ｎの値で指定される配列変数ｃｏｓｔｓの要素ｃｏｓｔｓ［ｎ］に０をそれぞれ代入し、変数ｃｎｔにも０を代入する。その次に移行するステップＳＵ４では、変数ｍｅの値で指定される音符に着目してコストを算出する指定音コスト算出処理を実行する。その実行後は、ステップＳＵ５に移行して、０から変数ｃｎｔの値未満までの変数ｎの値で指定される要素ｃｏｓｔｓ［ｎ］に代入されたコストの平均値を算出し、その算出結果をＭＥ［ｍｅ］．ｃｏｓｔとして設定し、変数ｃｏｓｔｔｍｐに、それまでの値にＭＥ［ｍｅ］．ｃｏｓｔの値を加算した値を代入し、変数ｍｅの値をインクリメントする。その後は上記ステップＳＵ２に戻る。

【０１３１】

図２１は、上記ステップＳＵ４として実行される指定音コスト算出処理のフローチャートである。次に図２１を参照して、そのコスト算出処理について詳細に説明する。

先ず、ステップＳＶ１では、変数ｆにＭＥ［ｍｅ］．ｆｉｇ、変数ｐにＭＥ［ｍｅ］．ｐｏｓｘ、変数ｂｗに変数ｐを２で割った余り、変数ｍｅｎに変数ｍｅの値に１を加算した値、をそれぞれ代入する。その次に移行するステップＳＶ２では、変数ｍｅの値が変数ｍｅ＿ｔａｉｌの値以上か、または変数ｃｎｔの値が定数ＣＯＳＴＳＥＥＫＣＮＴ以上か否か判定する。変数ｍｅの値が変数ｍｅ＿ｔａｉｌの値以上か、または変数ｃｎｔの値が定数ＣＯＳＴＳＥＥＫＣＮＴ以上であった場合、判定はＹＥＳとなり、ここで一連の処理を終了する。そうでない場合には、つまり変数ｍｅの値が変数ｍｅ＿ｔａｉｌの値未満、且つ変数ｃｎｔの値が定数ＣＯＳＴＳＥＥＫＣＮＴ未満であった場合には、判定はＮＯとなってステップＳＶ３に移行する。

【０１３２】
ステップＳＶ３では、変数ｎｆにＭＥ［ｍｅｎ］．ｆｉｇ、変数ｎｐにＭＥ［ｍｅｎ］．ｐｏｓｘをそれぞれ代入する。続くステップＳＶ４では、変数ｐの値が変数ｎｐの値より小さいか否か判定する。変数ｍｅの値で指定される音符が変数ｍｅｎの値で指定される音符より低音であった場合、変数ｐの値は変数ｎｐの値より小さくなることから、判定はＹＥＳとなり、ステップＳＶ５で変数ｐｍに０、変数ｉｎｔｖに変数ｎｐの値から変数ｐの値を引いた値（音符間のピッチ差）をそれぞれ代入してからステップＳＶ７に移行する。そうでない場合には、判定はＮＯとなり、ステップＳＶ６で変数ｐｍに１、変数ｉｎｔｖに変数ｐの値から変数ｎｐの値を引いた値をそれぞれ代入してからそのステップＳＶ７に移行する。変数ｐの値が変数ｎｐの値より小さいということは、発音させるべき音符のピッチ変化方向は増加の方向にあることを意味する。

【０１３３】
ステップＳＶ７では、変数ｉｎｔｖの値が定数ＭＡＸＩＮＴＶ未満か否か判定する。その定数ＭＡＸＩＮＴＶは、運指により押鍵を行う難易度が極めて高いと考えられる鍵間（ピッチ差）を判定するために用意したものである。このことから、着目する２つの音符の間にそのようなピッチ差が存在している場合、判定はＮＯとなり、ステップＳＶ８で要素ｃｏｓｔｓ［ｃｎｔ］にコストの最大値として設定の定数ＭＡＸＣＯＳＴを代入し、更にステップＳＶ９で変数ｃｎｔ、ｍｅｎの各値をインクリメントしてから上記ステップＳＶ２に戻る。一方、そうでない場合には、判定はＹＥＳとなり、ステップＳＶ１０で要素ｃｏｓｔｓ［ｃｎｔ］に、運指コストテーブルＣＴから抽出される値をコストとして代入してからステップＳＶ１１に移行する。

【０１３４】
図２２は、その運指コストテーブルＣＴのデータ構成を説明する図である。そのテーブルＣＴは、ＲＯＭ２に制御用データとして格納されている。
図２２中、「ｂｗ」は鍵盤種類（０は白鍵、１は黒鍵）、「ｐｍ」はピッチ変化方向（０は増加方向、１は減少方向）、「ｆｉｇ１」は変数ｍｅの値で指定される音符の指番号、「ｆｉｇ２」は変数ｍｅｎの値で指定される音符の指番号、「ｉｎｔｖ」はそれら２つの音符間のピッチ差、をそれぞれ表している。それにより、テーブルＣＴは、それらのデータ、つまり変数ｂｗ、ｐｍ、ｆ、ｎｆ、及びｉｎｔｖの各値の組み合わせで示される状況別に、その状況下での運指のコストを格納したものとなっている。このことから、要素ｃｏｓｔｓ［ｃｎｔ］には、変数ｂｗ、ｐｍ、ｆ、ｎｆ、及びｉｎｔｖの各値で指定される欄に格納されたコストが抽出して代入される。図２２では、５次元の配列変数ＣＴの形でテーブルＣＴを表記している。

【０１３５】
図２１の説明に戻る。
ステップＳＶ１１では、変数ｆの値が変数ｎｆの値と等しく、かつ変数ｐの値が変数ｎｐの値と等しくないか否か判定する。変数ｆ、ｎｆの値（指番号）が等しく、かつ変数ｐの値と変数ｎｐの値が等しくない（ピッチ（押鍵すべき鍵の音高）が等しくない）場合、判定はＹＥＳとなり、ステップＳＶ１２で変数ｓｆに１を代入した後、ステップＳＶ１４に移行する。そうでない場合には、つまり変数ｆの値が変数ｎｆの値と等しくないか、或いは変数ｐの値が変数ｎｐの値と等しいような場合には、判定はＮＯとなり、ステップＳＶ１３で変数ｓｆに０を代入した後、そのステップＳＶ１４に移行する。

【０１３６】
ステップＳＶ１４では、ＭＥ［ｍｅ］．ｔｉｍｅがＭＥ［ｍｅｎ］．ｔｉｍｅと等しいか否か判定する。変数ｍｅ、ｍｅｎの各値で指定される音符の発音開始時刻が同じ場合、判定はＹＥＳとなってステップＳＶ１５に移行し、そうでない場合には、判定はＮＯとなってステップＳＶ１７に移行する。

【０１３７】
ステップＳＶ１５では、変数ｓｆの値が１か、または変数ｆの値が変数ｎｆの値（指番号）より大きいか否か判定する。演奏データでは、発音開始時刻が同じ音符群はピッチの

小さな順に配列されている。このことから、発音開始時刻が同じでピッチが異なる２つの音符を同じ指で押鍵するか、またはピッチが大きいほうの音符を親指、或いは親指に近いほうの指で押鍵しなければならない運指であった場合、判定はＹＥＳとなり、ステップＳＶ１６で要素ｃｏｓｔｓ［ｃｎｔ］に定数ＭＡＸＣＯＳＴを代入してから上記ステップＳＶ９に移行する。そうでない場合には、つまり発音開始時刻が同じでピッチが異なる２つの音符を同じ指で押鍵せず、かつピッチが大きいほうの音符を親指、或いは親指に近いほうの指で押鍵しない運指であった場合には、判定はＮＯとなってそのステップＳＶ９に移行する。

【０１３８】

ステップＳＶ１７では、変数ｃｎｔの値が０、かつ変数ｓｆの値が１か否か判定する。変数ｃｎｔの値の０は、変数ｍｅｎの値で指定される音符は先頭の音符の次に位置していることを意味する。このことから、先頭、及びその次に位置する２つのピッチが異なる音符を同じ指で押鍵する運指であった場合、判定はＹＥＳとなって上記ステップＳＶ１６に移行し、そうでない場合には、判定はＮＯとなって上記ステップＳＶ９に移行する。

【０１３９】

このようにして、本実施の形態では、難易度の高い困難な運指の検出を行い、それによって検出された運指のコストとして定数ＭＡＸＣＯＳＴを設定するようにしている。それにより、設定された困難な運指はコストに大きく反映させている。

【０１４０】

図６、及び図７に示す指定区間運指生成処理内で実行されるサブルーチン処理の説明に戻る。

図２３、及び図２４は、その生成処理内でステップＳＪ１５として実行される分散和音判定処理のフローチャートである。次に図２３、及び図２４を参照して、その判定処理について詳細に説明する。

【０１４１】

先ず、ステップＳＤ１では、変数ｍｅ＿ｔａｉｌの値に１を加算した値が変数ｍｅ＿ｅの値を越えるならば変数ｍｅ＿ｓの値、そうでなければその加算値を変数ｍｅに代入する。次のステップＳＤ２では、変数ｍｅの値が変数ｍｅ＿ｅの値と等しいか否か判定する。それらの値が一致している場合、判定はＹＥＳとなり、ステップＳＤ３で変数ｍｅ＿ｔｏｐ、及びｍｅ＿ｔａｉｌにそれぞれ変数ｍｅ＿ｅの値を代入した後、一連の処理を終了する。そうでない場合には、判定はＮＯとなり、ステップＳＤ４に移行する。

【０１４２】

ステップＳＤ４では、変数ｔｃに０、変数ｎｐに変数ｍｅの値、変数ｓｔｅｐｔｉｍｅにＭＥ［ｎｐ＋１］．ｔｉｍｅからＭＥ［ｎｐ］．ｔｉｍｅを減算した値（変数ｎｐの値で指定される音符とその次音符間の発音開始時刻の時間間隔）をそれぞれ代入する。その後に移行するステップＳＤ５では、変数ｔｃの値が定数ＭＡＸＡＲＰの２倍以上、または変数ｎｐの値が変数ｍｅ＿ｅの値以上か否か判定する。変数ｔｃの値が定数ＭＡＸＡＲＰの２倍以上、または変数ｎｐの値が変数ｍｅ＿ｅの値以上であった場合、判定はＹＥＳとなってステップＳＤ８に移行する。そうでない場合には、つまり変数ｔｃの値が定数ＭＡＸＡＲＰの２倍未満、かつ変数ｎｐの値が変数ｍｅ＿ｅの値未満であった場合には、判定はＮＯとなってステップＳＤ６に移行する。上記定数ＭＡＸＡＲＰは、分散和音の構成音数として考えられる最大値として設定したものである。その最小値は、定数ＭＩＮＡＲＰとして設定している。

【０１４３】

ステップＳＤ６では、変数ｔｃの値が０でなく、かつ変数ｓｔｅｐｔｉｍｅの値がＭＥ［ｎｐ＋１］．ｔｉｍｅからＭＥ［ｎｐ］．ｔｉｍｅを減算した値と等しくないか否か判定する。変数ｔｃの値が０でないことは、減算値が示す発音開始時刻の時間間隔は、変数ｍｅの値で指定されない音符とその次音符間のものであることを意味する。このことから、その時間間隔が変数ｍｅの値で指定される音符とその次音符間のそれと一致しない場合、判定はＹＥＳとなってステップＳＤ８に移行する。そうでない場合には、判定はＮＯと

なり、ステップＳＤ７において、配列変数ｐｉｔｖｅｃの変数ｔｃの値で指定される要素ｐｉｔｖｅｃ［ｔｃ］にＭＥ［ｎｐ＋１］．ｐｉｔｃｈからＭＥ［ｎｐ］．ｐｉｔｃｈを減算した値（隣り合う音符のピッチ差）を代入し、その代入後に変数ｔｃ、及びｎｐの各値をインクリメントしてから上記ステップＳＤ５に戻る。それにより、変数ｔｃ、及びｎｐの各値を順次インクリメントしながら、分散和音が存在する可能性のある区間の抽出を行う。

【０１４４】

上記ステップＳＤ５、或いはＳＤ６の判定がＹＥＳとなって移行するステップＳＤ８では、変数ｍａｘｆｒに、変数ｔｃの値を２で割った値を代入する。その代入後はステップＳＤ９に移行して、変数ｍａｘｆｒの値が定数ＭＩＮＡＲＰ未満か否か判定する。その値が定数ＭＩＮＡＲＰ未満であった場合、判定はＹＥＳとなり、ステップＳＤ１０で変数ｍｅにそれまでの値に変数ｔｃの値を加算した値を代入してから上記ステップＳＤ２に戻る。そうでない場合には、判定はＮＯとなり、図２４のステップＳＤ１１に移行する。

【０１４５】

ステップＳＤ９でのＮＯの判定は、変数ｍｅ、ｎｐの各値で指定される区間内に分散和音が存在する可能性があることを意味する。このことから、そのステップＳＤ１１以降では、その区間内に存在する分散和音を特定するための処理が行われる。

【０１４６】

先ず、ステップＳＤ１１では、変数ｆｒに定数ＭＩＮＡＲＰを代入する。続くステップＳＤ１２では、変数ｆｒの値が変数ｍａｘｆｒの値より大きいか否か判定する。変数ｆｒの値が変数ｍａｘｆｒの値より大きい場合、判定はＹＥＳとなって図２３のステップＳＤ１０に移行し、そうでない場合には、判定はＮＯとなってステップＳＤ１３に移行する。

【０１４７】

ステップＳＤ１３では、変数ｎに０を代入する。次のステップＳＤ１４では、変数ｎの値が変数ｆｒの値未満か否か判定する。変数ｎの値が変数ｆｒの値以上であった場合、判定はＮＯとなってステップＳＤ１８に移行する。そうでない場合には、判定はＹＥＳとなり、ステップＳＤ１５に移行して、要素ｐｉｔｖｅｃ［ｎ］の値に要素ｐｉｔｖｅｃ［ｎ＋ｆｒ］の値を掛けた値が０未満か否か判定する。各要素ｐｉｔｖｅｃ［］には、隣り合う音符間のピッチ差が代入されている。このため、変数ｆｒの値分、離れた２つの音符でその次音符とのピッチ差の符号（ピッチ変化方向）が異なる場合、判定はＹＥＳとなり、ステップＳＤ１７で変数ｆｒの値をインクリメントしてから上記ステップＳＤ１２に戻る。そうでない場合には、判定はＮＯとなり、ステップＳＤ１６で変数ｎの値をインクリメントしてから上記ステップＳＤ１４に戻る。

【０１４８】

ステップＳＤ１８では、要素ｐｉｔｖｅｃ［ｆｒ］の値に要素ｐｉｔｖｅｃ［ｆｒ－１］の値を掛けた値が０未満か否か判定する。変数ｆｒの値で指定される音符とその前後に位置する各音符との間のピッチ差の符号（ピッチ変化方向）が異なる場合、判定はＹＥＳとなり、ステップＳＤ１９に移行する。すなわち、一般的な分散和音はピッチの低い音から始まるため、ステップＳＤ１８の判定がＹＥＳとなる。そしてステップＳＤ１９では変数ｍｅの値を変数ｍｅ＿ｔｏｐに、変数ｍｅの値に変数ｆｒの値を加算し、その加算結果から１を引いて得られる値を変数ｍｅ＿ｔａｉｌにそれぞれ代入した後、一連の処理を終了する。つまり分散和音の構成音が存在していると見なす区間を、変数ｍｅ＿ｔｏｐ、及びｍｅ＿ｔａｉｌの各値で指定する。一方ステップＳＤ１８の判定がＮＯである場合は、次にステップＳＤ１７に移行する。

【０１４９】

図６、及び図７に示す指定区間運指生成処理内では、上記分散和音判定処理に続いてステップＳＪ１６で分散和音全数検査処理を実行する。次にその検査処理について、図２５に示すそのフローチャートを参照して詳細に説明する。

【０１５０】

先ず、ステップＳＧ１では、変数ｍｅ＿ｔｏｐの値が変数ｍｅ＿ｔａｉｌの値と等しい

か否か判定する。それらの値が等しい場合、判定はＹＥＳとなり、ここで一連の処理を終了する。そうでない場合には、判定はＮＯとなり、次にステップＳＧ２で図１９に示す全数検査処理を実行してからステップＳＧ３に移行する。

【０１５１】

ステップＳＧ３では、変数ｍｅ＿ｔｏｐの値を変数ｍｅｔｍｐに、変数ｍｅ＿ｔａｉｌの値から変数ｍｅ＿ｔｏｐの値を減算し、その減算結果に１を加算して得られる値（分散和音の構成音数）を変数ｄにそれぞれ代入する。続くステップＳＧ４では、ＭＥ［ｍｅｔｍｐ＋ｄ］．ｆｉｇとしてＭＥ［ｍｅｔｍｐ］．ｆｉｇを設定し、その設定後に変数ｍｅｔｍｐの値をインクリメントする。その後に移行するステップＳＧ５では、変数ｍｅｔｍｐの値が変数ｍｅ＿ｔａｉｌの値より大きいか否か判定する。変数ｍｅｔｍｐの値が変数ｍｅ＿ｔａｉｌの値より大きい場合、判定はＹＥＳとなり、ステップＳＧ６において、変数ｍｅ＿ｔｏｐにそれまでの値に変数ｄの値を加算した値、変数ｍｅ＿ｔａｉｌにそれまでの値に変数ｄの値を加算した値をそれぞれ代入した後、一連の処理を終了する。そうでない場合には、判定はＮＯとなって上記ステップＳＧ４に戻る。

【０１５２】

分散和音の構成音が存在していると見なす区間は変数ｍｅ＿ｔｏｐ、及びｍｅ＿ｔａｉｌの各値で指定される。上述の分散和音判定処理では、２つの分散和音が連続する区間を対象に抽出するようにしている。それにより、本実施の形態では、最初の分散和音の構成音を押鍵していく運指をその次の分散和音に対しても適用させるようにしている。

【０１５３】

図２６は、図６、及び図７に示す指定区間運指生成処理内でステップＳＪ２０として実行される修正箇所検索処理のフローチャートである。次にその検索処理について、図２６を参照して詳細に説明する。

【０１５４】

先ず、ステップＳＨ１では、変数ｍｅ＿ｔａｉｌの値に１を加算した値が変数ｍｅ＿ｅの値以下であればその加算値、そうでなければ変数ｍｅ＿ｓの値を変数ｍｅｐに代入する。次のステップＳＨ２では、変数ｍｅに変数ｍｅｐの値に１を加算した値を代入する。その次に移行するステップＳＨ３では、変数ｍｅの値が変数ｍｅ＿ｅの値より大きいか否か判定する。変数ｍｅの値が変数ｍｅ＿ｅ以下であった場合、判定はＮＯとなってステップＳＨ４に移行する。そうでない場合には、判定はＹＥＳとなり、ステップＳＨ１３で変数ｍｅ＿ｔｏｐ、及びｍｅ＿ｔａｉｌにそれぞれ変数ｍｅ＿ｅの値を代入した後、一連の処理を終了する。

【０１５５】

ステップＳＨ４では、ＭＥ［ｍｅ］．ｃｏｓｔが定数ＴＨＲＣＯＳＴ１以上か否か判定する。その定数ＴＨＲＣＯＳＴ１は、運指を修正すべき箇所のコストの閾値として設定したものである。このことから、変数ｍｅの値で指定される音符を発音させるための運指（その音符のピッチが割り当てられた鍵の押鍵に使うべき指）の修正が望ましいような場合、判定はＹＥＳとなってステップＳＨ６に移行し、そうでない場合には、判定はＮＯとなり、ステップＳＨ５で変数ｍｅの値をインクリメントしてから上記ステップＳＨ３に戻る。それにより、修正すべき箇所の特定を継続させる。

【０１５６】

ステップＳＨ６では、変数ｍｅｔに変数ｍｅの値を代入し、変数ｍｅの値が変数ｍｅ＿ｅの値未満ならば変数ｍｅの値に１を加算した値、そうでなければ変数ｍｅの値を変数ｍｅｌに代入し、変数ｃｎｔに１を代入する。次のステップＳＨ７では、変数ｍｅｔの値が０、またはＭＥ［ｍｅｔ］．ｔｅｎｄが０か否か判定する。変数ｍｅｔの値で指定される音符が曲の先頭、または極点（データｔｅｎｄが０）に位置する音符であった場合、判定はＹＥＳとなってステップＳＨ９に移行する。そうでない場合には、判定はＮＯとなり、ステップＳＨ８で変数ｍｅｔの値をデクリメントすると共に、変数ｃｎｔの値をインクリメントしてから再度ステップＳＨ７での判定を行う。なお、各音符データＭＥ中のデータｔｅｎｄは、例えば図３に示すメインルーチン内のステップＳＡ３において、音符間のピ

ッチ差を算出し参照して設定するようになっている。

【０１５７】

ステップＳＨ９では、変数ｃｎｔの値が定数ＲＥＴＲＹＮ未満か否か判定する。その値が定数ＲＥＴＲＹＮ未満であった場合、判定はＹＥＳとなってステップＳＨ１０に移行する。そうでない場合には、判定はＮＯとなり、ステップＳＨ１２で変数ｍｅ＿ｔｏｐに変数ｍｅｔの値、変数ｍｅ＿ｔａｉｌに変数ｍｅｌの値をそれぞれ代入した後、一連の処理を終了する。

【０１５８】

ステップＳＨ１０では、変数ｍｅｌの値が変数ｍｅ＿ｅ、またはＭＥ［ｍｅｌ］．ｔｅｎｄが０か否か判定する。変数ｍｅｌの値で指定される音符が区間の最後、または極点に位置する音符であった場合、判定はＹＥＳとなってステップＳＨ１２に移行する。そうでない場合には、判定はＮＯとなり、ステップＳＨ１１で変数ｍｅｌ、及びｃｎｔの各値をインクリメントしてから再度ステップＳＨ１０での判定を行う。

【０１５９】

このようにして本実施の形態では、修正すべきと見なす箇所は、その前後の部分を含めて修正するようにしている。これは、１箇所の修正はその前後の演奏に影響を及ぼすのが普通であり、また、その箇所に至る部分の演奏を考慮しなければよりコストの低い運指を設定するのは困難、といった理由からである。定数ＲＥＴＲＹＮは、そのように修正対象範囲を設定することから、その範囲が必要以上に広がらないように用意している。

【０１６０】

次に、図６、及び図７に示す指定区間運指生成処理内でステップＳＪ２５として実行される同一フレーズ補正処理について、図２７、及び図２８に示すそのフローチャートを参照して詳細に説明する。その補正処理は、同一の（同一と見なせる）フレーズを検出し、そのなかでコストが最小のフレーズを抽出し、抽出したフレーズで設定されている運指を他のフレーズに適用させて補正する処理である。

【０１６１】

演奏内容が同一のフレーズであっても、その前、或いは後に位置するフレーズの影響によって運指（の組み合わせ）が異なることがありうる。その運指の違いは特に初心者に対しては違和感を与える可能性がある。本実施の形態では、そのような違和感を与えない運指情報を生成できるようにするために、同一フレーズ補正スイッチを設け、そのスイッチへの操作によりこの同一フレーズ補正処理を実行させるか否かユーザが選択できるようにさせている。

【０１６２】

先ず、ステップＳＸ１では、変数ｐｉｄに１を代入する。続くステップＳＸ２では、変数ｍｅに変数ｍｅ＿ｓの値、変数ｃｏｓｂｅｓｔには－１をそれぞれ代入する。その次に移行するステップＳＸ３では、変数ｍｅの値を順次、インクリメントしながら、その値で指定される音符データＭＥ［ｍｅ］中のデータｐｉｄ、及びｐｓｔａｔを確認していくことにより、データｐｉｄ、及びｐｓｔａｔが共に０である最初の音符データＭＥを指定するインデックス値を変数ｍｅ＿ｔｏｐに、その後に確認される、データｐｉｄが０でデータｐｓｔａｔが２の音符データＭＥを指定するインデックス値を変数ｍｅ＿ｔａｉｌにそれぞれ代入するための処理を行う。ステップＳＸ４にはその後に移行する。

【０１６３】

データｐｉｄ、及びｐｓｔａｔが共に０である音符データＭＥは分割したフレーズの先頭に位置し、データｐｉｄが０でデータｐｓｔａｔが２の音符データＭＥはそのフレーズの最後に位置する。ステップＳＸ４では、変数ｍｅの値が変数ｍｅ＿ｅと一致するまでに、そのような音符データＭＥのインデックス値を代入することによるフレーズの検出が行えたか否か判定する。ステップＳＸ３で２つのインデックス値の代入が行えた場合、判定はＹＥＳとなってステップＳＸ５に移行し、そうでない場合には、判定はＮＯとなって図２８のステップＳＸ１２に移行する。

【０１６４】

ステップＳＸ５では、変数ｃｏｓｔｂｅｓｔの値が－１か否か判定する。その値が－１であった場合、判定はＹＥＳとなり、ステップＳＸ６で変数ｍｅｂｅｓｔに変数ｍｅ＿ｔｏｐの値を代入してからステップＳＸ８に移行する。そうでない場合には、判定はＮＯとなり、ステップＳＸ７に移行して、今回、検出したフレーズが、変数ｍｅｂｅｓｔに先頭位置の音符データＭＥのインデックス値が代入されたフレーズと同じか否か判定する。それらのフレーズが同一のものであった場合、判定はＹＥＳとなってステップＳＸ８に移行する。そうでない場合には、判定はＮＯとなって上記ステップＳＸ３に戻る。

【０１６５】

フレーズが同一か否かの判定は、特に詳細な説明は省略するが、例えば隣り合う音符間のピッチ差、押鍵すべき鍵の種類に着目して行っている。当然のことながら、別の内容に着目して判定を行うようにしても良い。

【０１６６】

ステップＳＸ８では、ＭＥ［ｍｅ＿ｔｏｐ］．ｐｉｄとして変数ｐｉｄの値をセットする。次のステップＳＸ９では、図２０に示す区間コスト算出処理を実行する。その次に実行するステップＳＸ１０では、変数ｃｏｓｔｂｅｓｔの値が－１、または変数ｃｏｓｔｔｍｐの値が変数ｃｏｓｔｂｅｓｔの値未満か否か判定する。変数ｃｏｓｔｂｅｓｔの値が－１、または変数ｃｏｓｔｔｍｐの値が変数ｃｏｓｔｂｅｓｔの値未満であった場合、判定はＹＥＳとなり、ステップＳＸ１１で変数ｃｏｓｔｂｅｓｔに変数ｃｏｓｔｔｍｐの値、変数ｍｅｂｅｓｔに変数ｍｅ＿ｔｏｐの値をそれぞれ代入した後、上記ステップＳＸ３に戻る。そうでない場合には、変数ｃｏｓｔｂｅｓｔの値が－１でなく、かつ変数ｃｏｓｔｔｍｐの値が変数ｃｏｓｔｂｅｓｔの値より大きい場合には、判定はＮＯとなって上記ステップＳＸ３に戻る。

【０１６７】

このようにして、本実施の形態では、分割したフレーズのなかで同一の（同一と見なす）フレーズには同一のフレーズ番号を割り当て、同一のフレーズのなかでコストが最小のフレーズを特定し、特定したフレーズの先頭に位置する音符データＭＥのインデックス値を変数ｍｅｂｅｓｔに代入している。図２８のステップＳＸ１２には、そのようなフレーズ番号の割り当て、コストが最小のフレーズの特定を試みた後に移行する。

【０１６８】

そのステップＳＸ１２では、変数ｃｏｓｔｂｅｓｔの値が－１、または変数ｃｏｓｔｂｅｓｔの値が定数ＭＡＸＣＯＳＴと等しいか否か判定する。変数ｃｏｓｔｂｅｓｔの値としての－１は、対象となるフレーズ自体が検出されなかったことを意味する。定数ＭＡＸＣＯＳＴは、他のフレーズに運指を適用させるフレーズのコストがその適用を行うべきレベルか否か判定するために用意したものである。このようなことから、結果として、運指を適用するフレーズが存在しない場合には、判定はＹＥＳとなってステップＳＸ１８に移行し、そうでない場合には、判定はＮＯとなってステップＳＸ１３に移行する。

【０１６９】

ステップＳＸ１３では、変数ｍｅに変数ｍｅ＿ｓの値を代入する。続くステップＳＸ１４では、上記ステップＳＸ３と同様にして、データｐｉｄが変数ｐｉｄの値でデータｐｓｔａｔが０である最初の音符データＭＥを指定するインデックス値を変数ｍｅ＿ｔｏｐに、その後に確認される、データｐｉｄが変数ｐｉｄの値でデータｐｓｔａｔが２の音符データＭＥを指定するインデックス値を変数ｍｅ＿ｔａｉｌにそれぞれ代入するための処理を行う。ステップＳＸ１５にはその後に移行する。

【０１７０】

ステップＳＸ１５では、変数ｍｅの値が変数ｍｅ＿ｅと一致するまでに、そのような音符データＭＥのインデックス値を代入することによるフレーズの検出が行えたか否か判定する。ステップＳＸ１４で２つのインデックス値の代入が行えた場合、判定はＹＥＳとなってステップＳＸ１６に移行し、そうでない場合には、判定はＮＯとなってステップＳＸ１８に移行する。

【０１７１】

ステップＳＸ１６では、変数ｍｅ＿ｔｏｐの値が変数ｍｅｂｅｓｔの値と等しいか否か判定する。それらの値が等しい場合、つまり変数ｍｅ＿ｔｏｐの値として代入されたインデックス値は運指を適用させる元のフレーズの先頭に位置する音符データＭＥを指定するものであった場合、判定はＹＥＳとなって上記ステップＳＸ１４に戻る。そうでない場合には、判定はＮＯとなってステップＳＸ１７に移行し、コストが最小のフレーズで設定された運指を適用させるためのコピーを行う。そのコピーは、変数ｐ１に変数ｍｅ＿ｔｏｐの値から変数ｍｅ＿ｔａｉｌの値まで順次、代入しながら、その変数ｐ１の値で指定される音符データＭＥ［ｐ１］のデータｆｉｇとして、初期値が変数ｍｅｂｅｓｔの値でその値から変数ｐ１の値と同じタイミングで順次インクリメントする変数ｐ２の値で指定される音符データＭＥ［ｐ２］のデータｆｉｇを設定（つまりＭＥ［ｐ１］．ｆｉｇ←ＭＥ［ｐ２］．ｆｉｇ）することで行われる。そのようにして適用を行った後は上記ステップＳＸ１４に戻る。

【０１７２】

上記ステップＳＸ１２の判定がＹＥＳ、或いはＳＸ１５の判定がＮＯとなって移行するステップＳＸ１８では、変数ｐｉｄの値をインクリメントする。続くステップＳＸ１９では、変数ｐｉｄの値が定数ＭＡＸＰＩＤと等しいか否か判定する。その値が定数ＭＡＸＰＩＤと等しい場合、判定はＹＥＳとなり、ここで一連の処理を終了する。そうでない場合には、判定はＮＯとなって図２７のステップＳＸ２に戻る。

【０１７３】

図６、及び図７に示す指定区間運指生成処理では、上述したようなサブルーチン処理が実行される。そのようなサブルーチン処理を変数ｌｏｏｐｃｎｔの値に応じて実行することから、その値をインクリメントさせながら指定区間運指生成処理を実行させる度に、運指情報はより最適化されていくこととなる。

【０１７４】

図２９は、図５に示す運指生成処理内でステップＳＢ３として実行される評価処理のフローチャートである。最後に図２９を参照して、その評価処理について詳細に説明する。その評価処理では、上述したように、運指コストとして、１音符平均のコストを算出し、算出したコストが定数ＴＨＲＣＯＳＴ２より小さければ、生成、或いは修正した運指情報は適切なものであるとして変数ｌｏｏｐｃｎｔに０を代入するようになっている。

【０１７５】

先ず、ステップＳＷ１では、変数ｍｅ＿ｔｏｐに変数ｍｅ＿ｓの値、変数ｍｅ＿ｔａｉｌに変数ｍｅ＿ｓの値をそれぞれ代入し、それに続くステップＳＷ２で図２０に示す区間コスト算出処理を実行する。その次に移行するステップＳＷ３では、変数ｃｏｓｔｔｍｐの値が定数ＴＨＲＣＯＳＴ２未満か否か判定する。その定数ＴＨＲＣＯＳＴ２は、生成された運指情報が最適（適切）なものか否か判定するために用意した閾値である。このことから、その運指情報が最適なものであった場合、判定はＹＥＳとなり、ステップＳＷ４で変数ｌｏｏｐｃｎｔに０を代入した後、一連の処理を終了する。そうでない場合には、判定はＮＯとなり、ここで一連の処理を終了する。

【０１７６】

図５の運指生成処理では、ステップＳＢ４の判定がＮＯ、つまり変数ｌｏｏｐｃｎｔの値が０となるまでステップＳＢ２～ＳＢ４で形成される処理ループを繰り返し実行する。その繰り返しを必要に応じて行わせることにより、ステップＳＢ２の指定区間運指生成処理によって最初に生成された運指情報は、それ以降の生成処理の実行により、より最適（適切）な運指情報となるまで段階的に順次、補正されていくことになる。

【０１７７】

なお、本実施の形態では、変数ｌｏｏｐｃｎｔの値に応じた運指情報の生成は一つの内容にのみ着目して行うようにしているが（図６、及び図７）、複数の内容に着目して行うようにしても良い。着目する内容、その順序は演奏データ、或いは運指情報を参照して決定するようにしても良い。

【０１７８】

上述したような運指情報生成装置を実現させるようなプログラムは、ＣＤ－ＲＯＭ、ＤＶＤ、或いは着脱自在なフラッシュメモリ等の記録媒体に記録させて配布しても良い。公衆網等の通信ネットワークを介して、そのプログラムの一部、若しくは全部を配信するようにしても良い。そのようにした場合には、ユーザーはプログラムを取得してデータ処理装置にロードすることにより、その装置に本発明を適用させることができる。このことから、記録媒体は、プログラムを配信する装置がアクセスできるものであっても良い。

【図面の簡単な説明】

【０１７９】

【図１】本実施の形態による運指情報生成装置を搭載した電子楽器の構成を説明する図である。

【図２】電子楽器のＲＡＭに格納されるデータを説明する図である。

【図３】全体処理のフローチャートである。

【図４】範囲設定処理のフローチャートである。

【図５】運指生成処理のフローチャートである。

【図６】指定区間運指生成処理のフローチャートである。

【図７】指定区間運指生成処理のフローチャートである（続き）。

【図８】逐次生成処理のフローチャートである。

【図９】電子楽器のＲＯＭに格納されたデータを説明する図である（その１）。

【図１０】指状態確認処理のフローチャートである。

【図１１】指割り当て処理のフローチャートである。

【図１２】指位置指定処理のフローチャートである。

【図１３】フレーズ分割処理のフローチャートである。

【図１４】パターン適用処理のフローチャートである。

【図１５】電子楽器のＲＯＭに格納されたデータを説明する図である（その２）。

【図１６】パターンマッチング処理のフローチャートである。

【図１７】和音判定処理のフローチャートである。

【図１８】和音全数検査処理のフローチャートである。

【図１９】全数検査処理のフローチャートである。

【図２０】区間コスト算出処理のフローチャートである。

【図２１】指定音コスト算出処理のフローチャートである。

【図２２】電子楽器のＲＯＭに格納されたデータを説明する図である（その３）。

【図２３】分散和音判定処理のフローチャートである。

【図２４】分散和音判定処理のフローチャートである（続き）。

【図２５】分散和音全数検査処理のフローチャートである。

【図２６】修正箇所検索処理のフローチャートである。

【図２７】同一フレーズ補正処理のフローチャートである。

【図２８】同一フレーズ補正処理のフローチャートである（続き）。

【図２９】評価処理のフローチャートである。

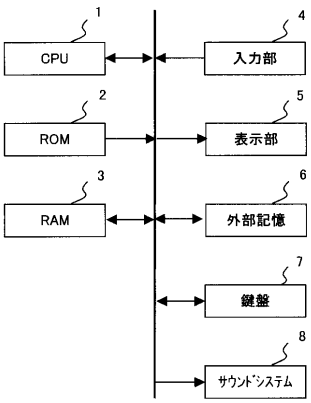
【符号の説明】

【０１８０】

１　ＣＰＵ
２　ＲＯＭ
３　ＲＡＭ
４　入力部
５　表示部
６　外部記憶装置

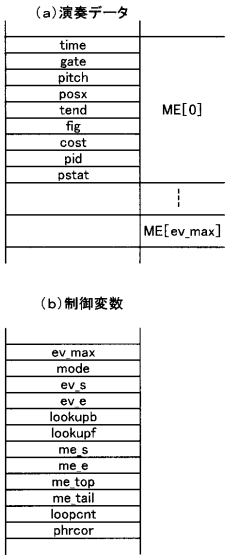
【図１】

本実施の形態による運指情報生成装置を搭載した電子楽器の構成を説明する図



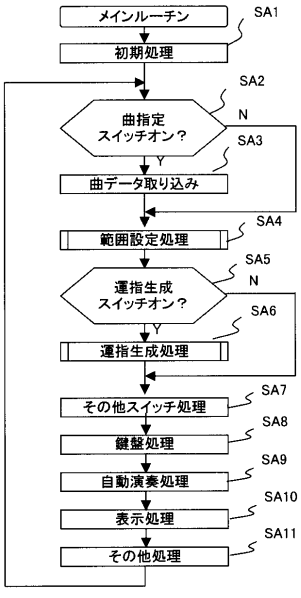
【図２】

電子楽器のRAMに格納されるデータを説明する図



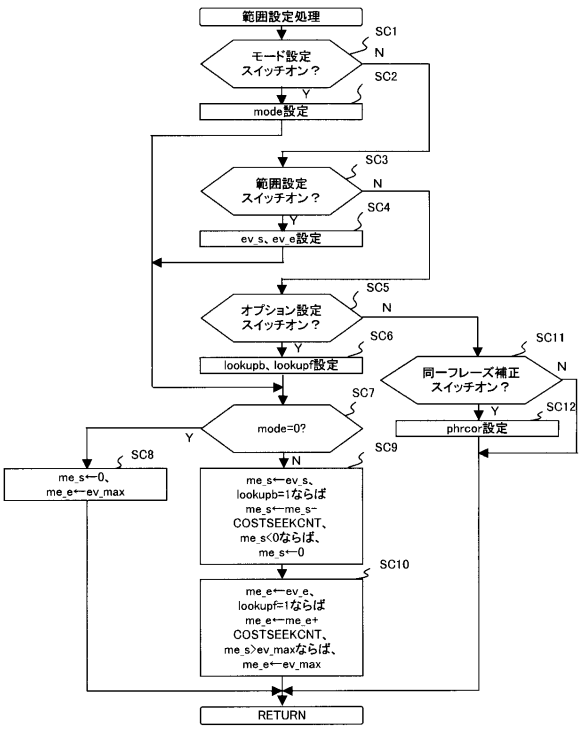
【図３】

全体処理のフローチャート



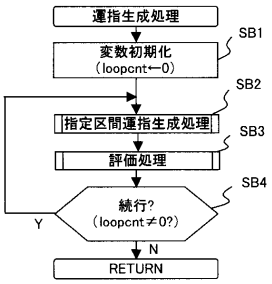
【図４】

範囲設定処理のフローチャート



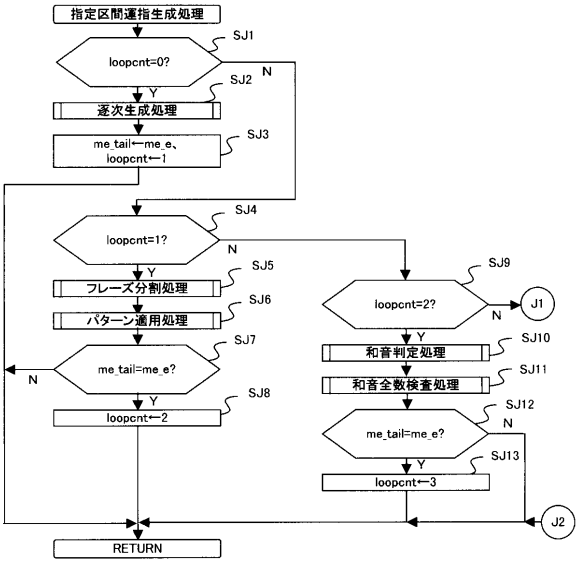
【図５】

運指生成処理のフローチャート



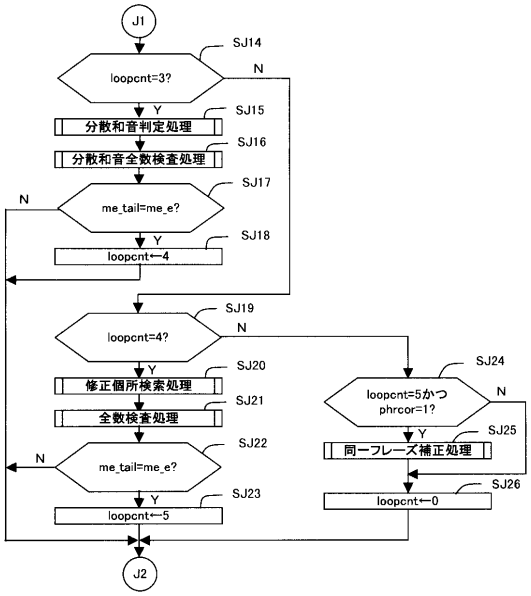
【図６】

指定区間運指生成処理のフローチャート



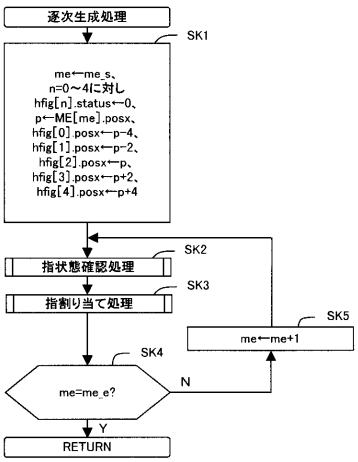
【図７】

指定区間運指生成処理のフローチャート(続き)



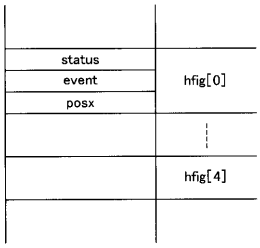
【図８】

逐次生成処理のフローチャート



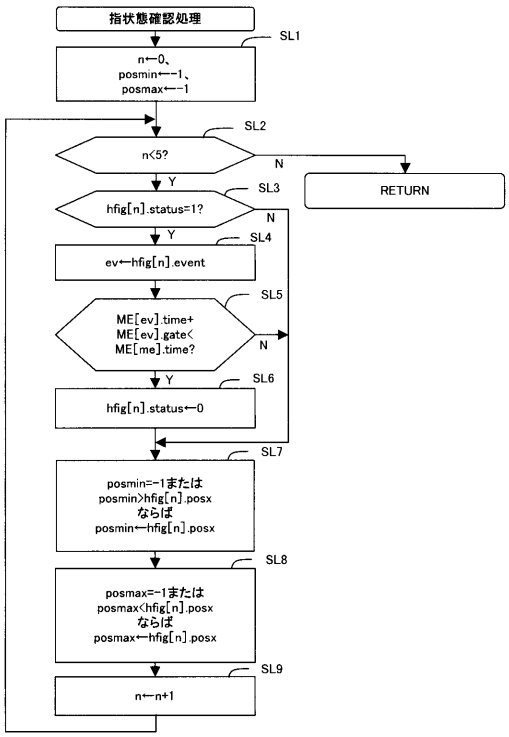
【図９】

電子楽器のROMに格納されたデータを説明する図（その1）



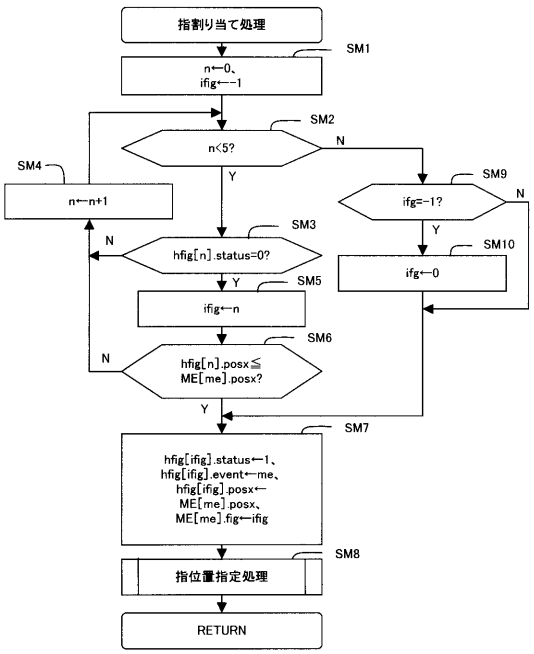
【図１０】

指状態確認処理のフローチャート



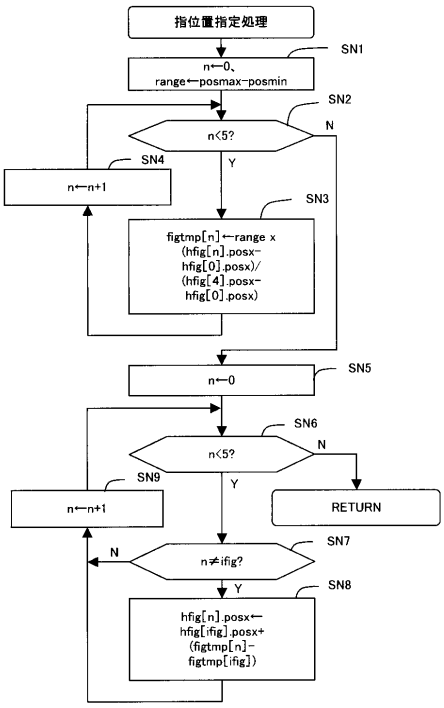
【図１１】

指割り当て処理のフローチャート



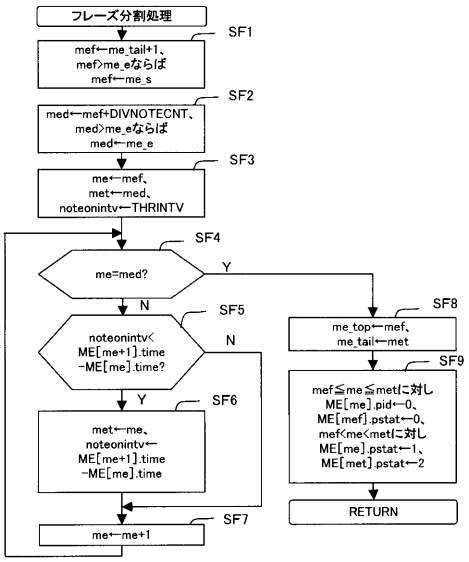
【図１２】

指位置指定処理のフローチャート



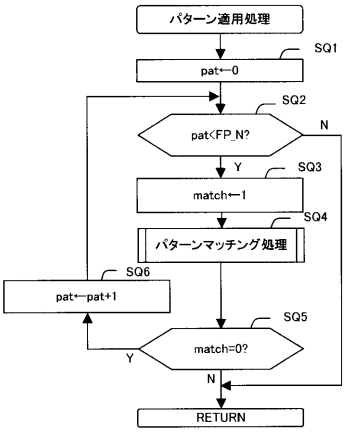
【図１３】

フレーズ分割処理のフローチャート



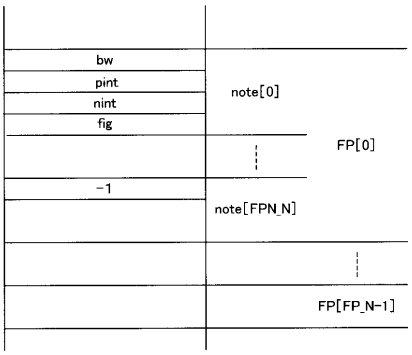
【図１４】

パターン適用処理のフローチャート



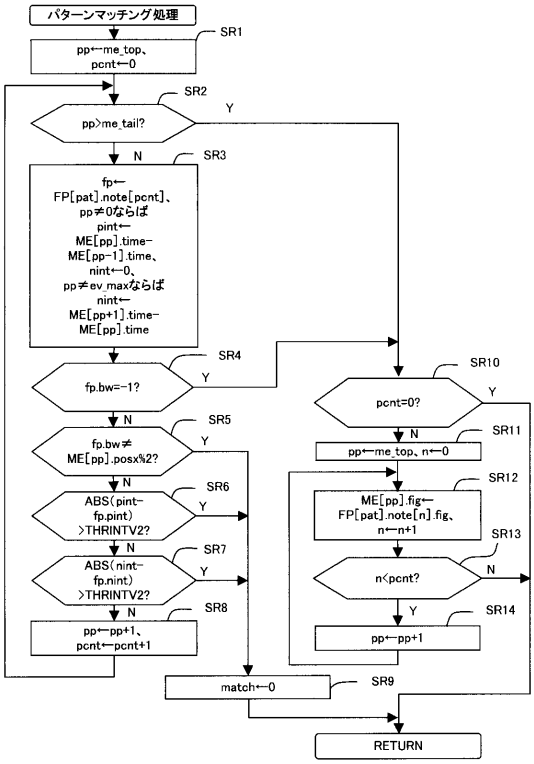
【図１５】

電子楽器のROMに格納されたデータを説明する図（その2）



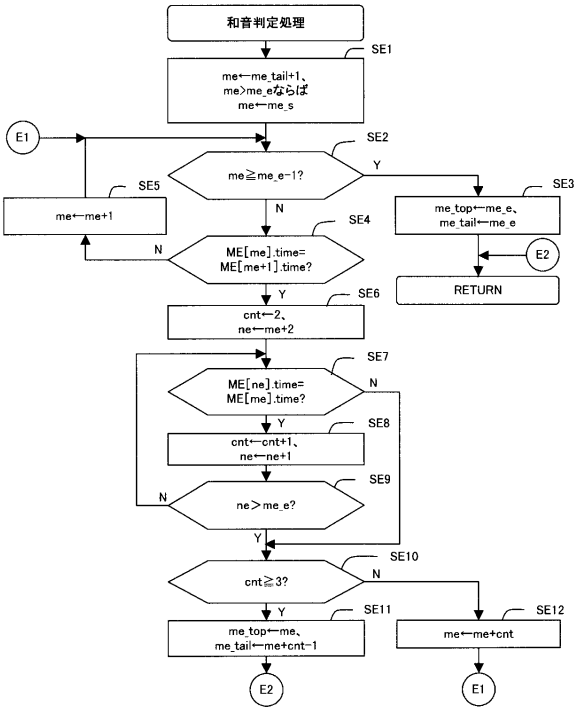
【図１６】

パターンマッチング処理のフローチャート



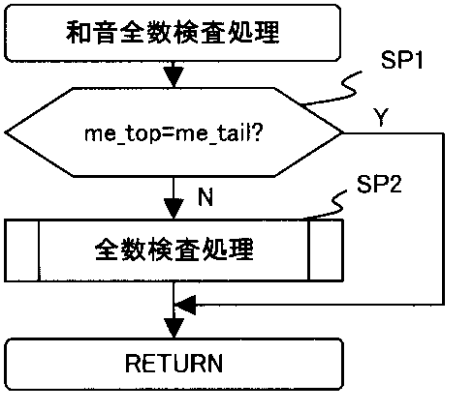
【図１７】

和音判定処理のフローチャート



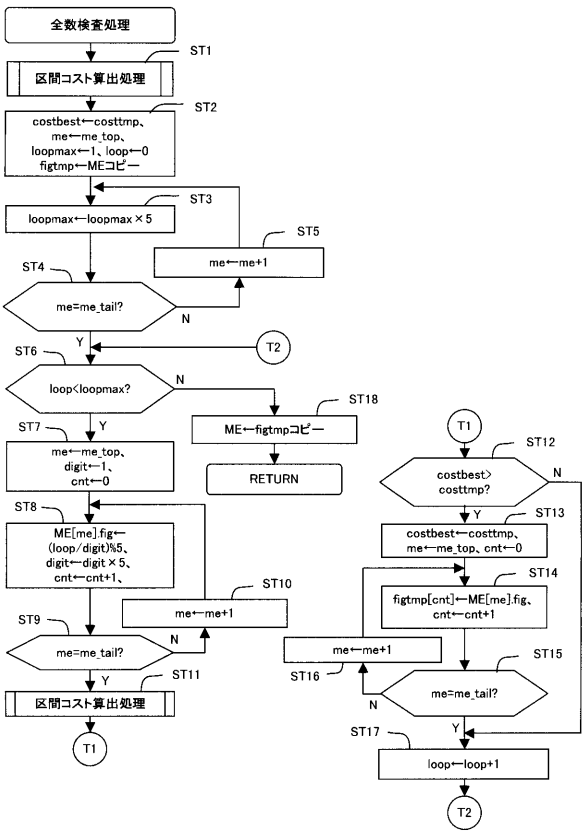
【図１８】

和音全数検査処理のフローチャート



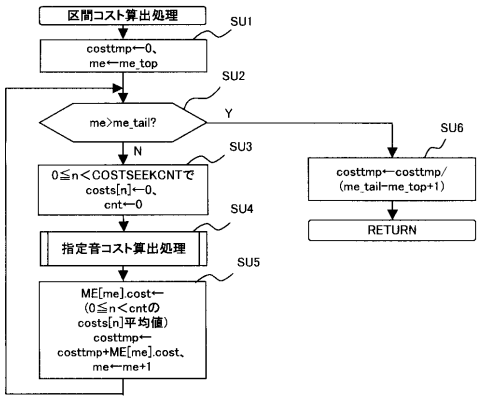
【図１９】

全数検査処理のフローチャート



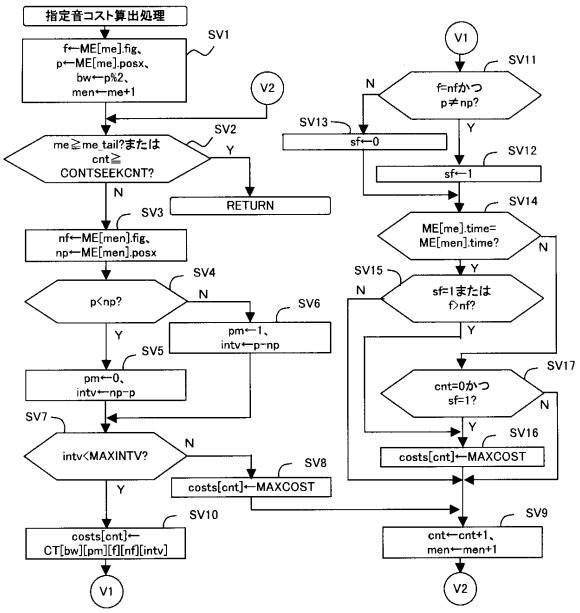
【図２０】

区間コスト算出処理のフローチャート



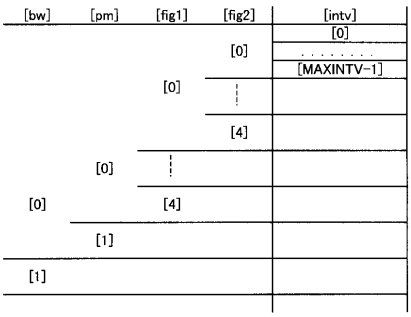
【図２１】

指定音コスト算出処理のフローチャート



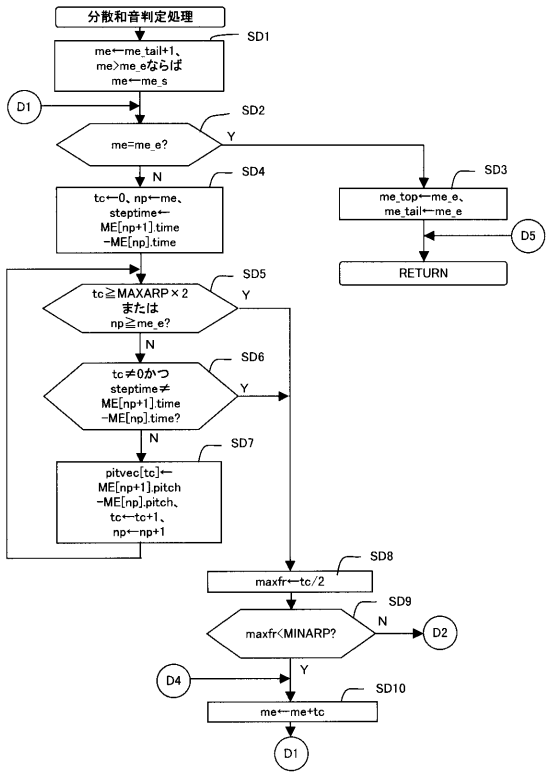
【図２２】

電子楽器のROMに格納されたデータを説明する図（その3）



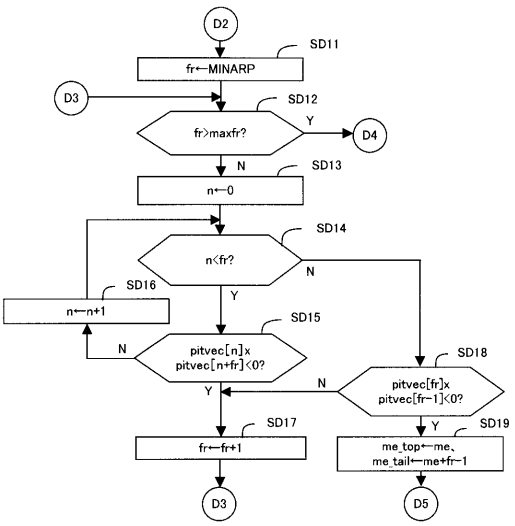
【図２３】

分散和音判定処理のフローチャート



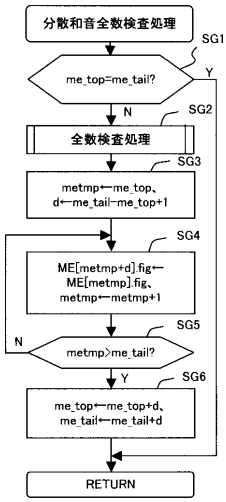
【図２４】

分散和音判定処理のフローチャート（続き）



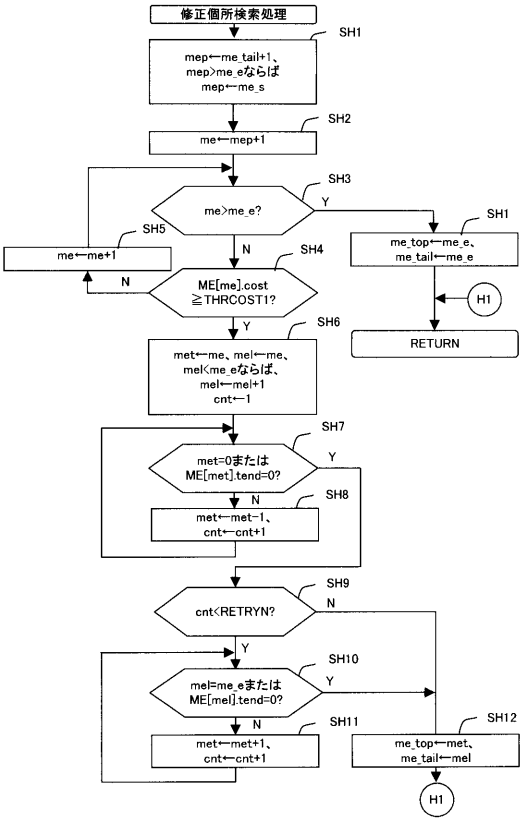
【図２５】

分散和音全数検査処理のフローチャート



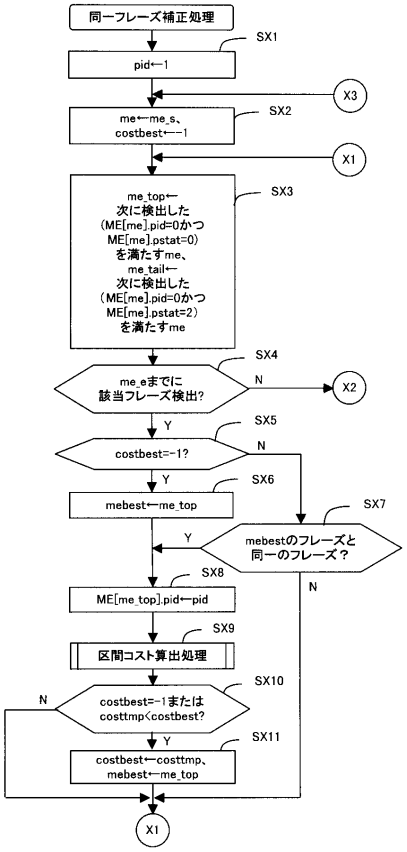
【図２６】

修正箇所検索処理のフローチャート



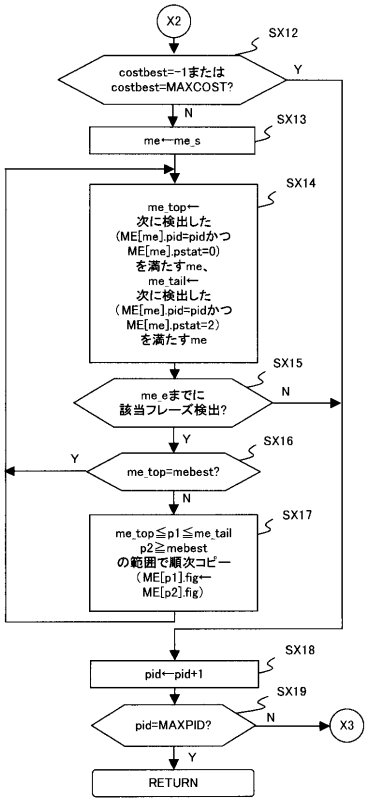
【図２７】

同一フレーズ補正処理のフローチャート



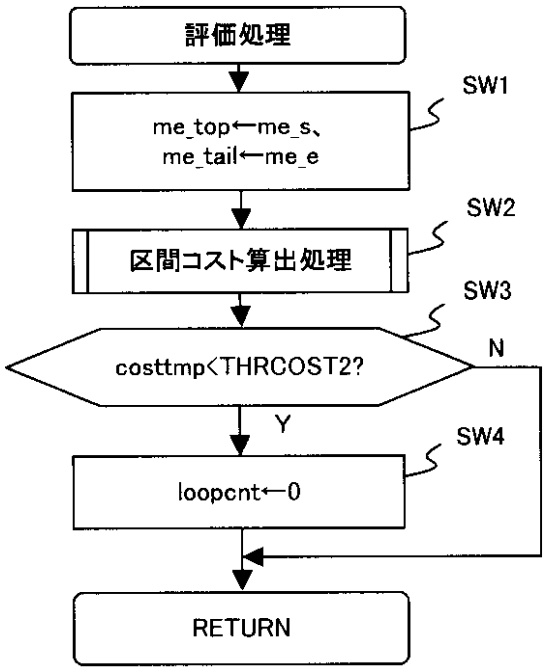
【図２８】

同一フレーズ補正処理のフローチャート（続き）



【図２９】

評価処理のフローチャート



フロントページの続き

(56)参考文献 特開２００１－３３１１７３（ＪＰ，Ａ）
特開２０００－３２２０５９（ＪＰ，Ａ）
特開２００４－２０５７９１（ＪＰ，Ａ）
特開２０００－１６３０５５（ＪＰ，Ａ）
特開２００６－７１６７５（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０９Ｂ　１５／００－１５／０８
Ｇ１０Ｇ　１／００－　７／０２
Ｇ１０Ｈ　１／００－　７／１２