(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6428854号
(P6428854)
(45)発行日　平成30年11月28日(2018.11.28)　(24)登録日　平成30年11月9日(2018.11.9)

(51)Int.Cl.　F I
G10H 1/00 (2006.01)　G10H 1/00 102Z
G10G 1/04 (2006.01)　G10G 1/04

請求項の数 6　(全 47 頁)

(21)出願番号　特願2017-114113(P2017-114113)
(22)出願日　平成29年6月9日(2017.6.9)
(62)分割の表示　特願2014-235236(P2014-235236)の分割
原出願日　平成26年11月20日(2014.11.20)
(65)公開番号　特開2017-151487(P2017-151487A)
(43)公開日　平成29年8月31日(2017.8.31)
審査請求日　平成29年11月16日(2017.11.16)

(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
(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　山下　剛史

最終頁に続く

(54)【発明の名称】自動作曲装置、方法、およびプログラム

(57)【特許請求の範囲】
【請求項１】
入力されたモチーフに含まれる各ノートのピッチを変更することにより変形メロディを生成する変形メロディ生成処理と、
連続するノートのノートタイプと前記連続するノートの音高の差を示す隣接音程との組み合わせを評価するノート接続ルールにおける前記評価に基づいて、前記変形メロディ生成処理により生成された変形メロディに含まれる各ノートのピッチを夫々シフトさせることにより得られる複数通りのピッチ列に対して、前記ピッチ列ごとに適合度を算出する適合度算出処理と、
前記適合度算出処理により算出された適合度に応じて前記複数通りのピッチ列のなかから選択されるピッチ列に基づいて、メロディを生成するメロディ生成処理と、
を実行する制御部を備えることを特徴とする自動作曲装置。
【請求項２】
前記制御部は、
前記入力されたモチーフに含まれる各ノートのピッチをどのように変更するかを示す変形タイプをユーザに指定させる変形タイプ指定処理、
を実行し、
前記変形メロディ生成処理は、
前記変形タイプ指定処理により指定された変形タイプに基づいて、前記モチーフに含まれる各ノートのピッチを変更することを特徴する請求項１に記載の自動作曲装置。

【請求項３】
前記変形タイプは、前記各ノートのピッチを夫々ピッチシフトさせるタイプ、或いは、前記各ノートのピッチを小節内で左右反転させるタイプを含むことを特徴とする請求項２に記載の自動作曲装置。
【請求項４】
前記メロディ生成処理は、
前記入力されたモチーフに含まれる各ノートのピッチ及び前記選択されるピッチ列に基づいて、前記メロディを生成することを特徴する請求項１乃至３のいずれかに記載の自動作曲装置。
【請求項５】
自動作曲装置が、
入力されたモチーフに含まれる各ノートのピッチを変更することにより変形メロディを生成し、
連続するノートのノートタイプと前記連続するノートの音高の差を示す隣接音程との組み合わせを評価するノート接続ルールにおける前記評価に基づいて、前記変形メロディ生成処理により生成された変形メロディに含まれる各ノートのピッチを夫々シフトさせることにより得られる複数通りのピッチ列に対して、前記ピッチ列ごとに適合度を算出し、
前記算出された適合度に応じて前記複数通りのピッチ列のなかから選択されるピッチ列に基づいて、メロディを生成する、自動作曲方法。
【請求項６】
自動作曲装置として用いられるコンピュータに、
入力されたモチーフに含まれる各ノートのピッチを変更することにより変形メロディを生成させるステップと、
連続するノートのノートタイプと前記連続するノートの音高の差を示す隣接音程との組み合わせを評価するノート接続ルールにおける前記評価に基づいて、前記変形メロディ生成処理により生成された変形メロディに含まれる各ノートのピッチを夫々シフトさせることにより得られる複数通りのピッチ列に対して、前記ピッチ列ごとに適合度を算出するステップと、
前記算出された適合度に応じて前記複数通りのピッチ列のなかから選択されるピッチ列に基づいて、メロディを生成するステップと、
を実行させるプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、自動作曲装置、方法、およびプログラムに関する。
【背景技術】
【０００２】
複数のノートデータから構成されるモチーフメロディに基づいて、自動作曲を行う技術が知られている。例えば、次のような従来技術が知られている（例えば特許文献１に記載の技術）。モチーフ区間メロディから所定の音に関する楽音情報を抽出する手段と、抽出した楽音情報を演算する手段と、該演算した楽音情報に基づいて、モチーフ区間に後続する区間のメロディを生成する手段を具える。
【０００３】
また、次のような従来技術も知られている（例えば特許文献２に記載の技術）。メロディ生成用データをスタイル等に応じて選択しロードする。音符数条件に応じて音符数パラメータをランダムに生成する。跳躍条件に応じて度数パラメータと跳躍回数パラメータをランダムに生成する。シンコペーション条件に応じてシンコペーション回数パラメータをランダムに生成する。変化幅大／変化幅小のスイッチの操作選択により音符数条件、跳躍条件、シンコペーション条件を一括コントロールし、各条件におけるランダム特性を操作選択されたスイッチに対応したものとし、そのランダム特性で各パラメータをランダム生

成する。生成されたパラメータ、メロディ生成用データおよび音楽ルールに基づいてメロディを生成する。

【先行技術文献】

【特許文献】

【０００４】

【特許文献１】特許公開２００２－３２０７９号公報

【特許文献２】特許公開２０００－１６３０５２号公報

【発明の概要】

【発明が解決しようとする課題】

【０００５】

しかし、上記一の従来技術では、参照元のメロディから音楽情報を抽出して、全体を生成する手段があるが、生成されるメロディが最適解であるかどうかは保障されない。このようなメロディは、ある規則にのっとって生成されたものであり、自然なメロディであるかもしれないが部分最適である可能性もあるという課題があった。また、上記他の一の従来技術では、メロディ生成のためのパラメータ生成をランダムに生成する方法が提案されているが、それでも部分最適であり、また、ユーザがコントロールしずらいシステムであるという課題があった。

【０００６】

そこで、本発明は、コードやスケールに対して最適なメロディ（音列）を保障することを目的とする。

【課題を解決するための手段】

【０００７】

態様の一例では、入力されたモチーフに含まれる各ノートのピッチを変更することにより変形メロディを生成する変形メロディ生成処理と、連続するノートのノートタイプと前記連続するノートの音高の差を示す隣接音程との組み合わせを評価するノート接続ルールにおける前記評価に基づいて、前記変形メロディ生成処理により生成された変形メロディに含まれる各ノートのピッチを夫々シフトさせることにより得られる複数通りのピッチ列に対して、前記ピッチ列ごとに適合度を算出する適合度算出処理と、前記適合度算出処理により算出された適合度に応じて前記複数通りのピッチ列のなかから選択されるピッチ列に基づいて、メロディを生成するメロディ生成処理と、を実行する制御部を備える。

【発明の効果】

【０００８】

本発明によれば、コードやスケールに対して最適なメロディ（音列）を保障することが可能となる。

【図面の簡単な説明】

【０００９】

【図１】自動作曲装置の実施形態のブロック図である。

【図２】本実施形態において自動作曲される楽曲の構造例を示す図である。

【図３】入力モチーフ１０８とコード進行データの適合動作例

【図４】入力モチーフのデータ構成例を示す図である。

【図５】伴奏・コード進行ＤＢのデータ構成例を示す図である。

【図６】１レコード中の曲構造データのデータ構成例を示す図である。

【図７】標準ピッチクラスセットテーブルのデータ構成例を示す図である。

【図８】ノートタイプ、隣接音程、およびノートタイプと隣接音程の配列変数データについての説明図である。

【図９】ノート接続ルールのデータ構成例を示す図である。

【図１０】コード進行選択部１０２の動作説明図である。

【図１１】フレーズセットＤＢのデータ構成例を示す図である。

【図１２】メロディ変形処理およびメロディ最適化処理の動作説明図である。

【図１３】メロディ最適化処理の詳細動作説明図である。
【図１４】自動作曲装置のハードウェア構成例を示す図である。
【図１５Ａ】各種変数データ、配列変数データ、および定数データのリストを示す図（その１）である。
【図１５Ｂ】各種変数データ、配列変数データ、および定数データのリストを示す図（その２）である。
【図１６】自動作曲処理の例を示すフローチャートである。
【図１７】コード進行選択処理の詳細例を示すフローチャートである。
【図１８】コードデザインデータ作成処理の詳細例を示すフローチャートである。
【図１９】入力モチーフとコード進行の適合度チェック処理の詳細例を示すフローチャートである。
【図２０】チェック処理の詳細例を示すフローチャートである。
【図２１】入力モチーフの現在のノートのタイミングに対応するコード情報の取得処理の詳細例を示す図である。
【図２２】ノートタイプ取得処理の詳細例を示す図である。
【図２３】ノート接続性チェック処理の詳細例を示す図である。
【図２４】メロディ生成処理の詳細例を示す図である。
【図２５】メロディ生成１処理の詳細例を示す図である。
【図２６】フレーズセットＤＢ検索処理の詳細例を示す図ある。
【図２７】メロディ変形処理の詳細例を示す図である。
【図２８】メロディ最適化処理の詳細例を示す図である。
【図２９】メロディ生成２処理の詳細例を示す図である。
【発明を実施するための形態】
【００１０】
以下、本発明を実施するため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、自動作曲装置１００の実施形態のブロック図である。この自動作曲装置１００は、モチーフ入力部１０１、コード進行選択部１０２、伴奏・コード進行データベース（以下、「データベース」を「ＤＢ」と称する）１０３、ルールＤＢ１０４、メロディ生成部１０５、フレーズセットＤＢ１０６、および出力部１０７を備える。
【００１１】
モチーフ入力部１０１は、いわゆるＡメロ、Ｂメロ、Ｃメロ（サビメロ）などの、曲調を決定付ける特徴的なメロディ部分のいずれかを、入力モチーフ１０８としてユーザに入力させる。入力モチーフ１０８は、Ａメロ部分のモチーフであるモチーフＡ、Ｂメロ部分のモチーフであるモチーフＢ、あるいはＣメロ（サビメロ）部分のモチーフであるモチーフＣのいずれかであり、例えば各メロディ部分の先頭の２小節の長さを有する。モチーフ入力部１０１は例えば、ユーザが鍵盤によりメロディを入力する鍵盤入力部１０１－１、ユーザがマイクから歌声によりメロディを入力する音声入力部１０１－２、ユーザがメロディを構成する音符のデータをキーボード等から入力する音符入力部１０１－３のいずれか一つ以上の手段を備える。また入力部１０１は、Ａメロ、Ｂメロ、Ｃメロ（サビメロ）というモチーフの種別を入力する、独立した操作子等を有する。
【００１２】
コード進行選択部１０２は、伴奏・コード進行ＤＢ１０３に記憶されている複数のコード進行データごとに、ルールＤＢ１０４を参照しながら、そのコード進行データがモチーフ入力部１０１から入力された入力モチーフ１０８にどの程度適合しているかを示す適合度を算出し、適合度が高かった例えば上位３個のコード進行データをそれぞれ指し示す＃０、＃１、＃２のコード進行候補指示データ（図１中では「コード進行候補」と表示）１０９を出力する。
【００１３】
メロディ生成部１０５は、例えばユーザに、コード進行選択部１０２が出力した＃０、＃１、＃２のコード進行候補指示データ１０９に対応する３つのコード進行候補のうちの

１つを選択させる。あるいは、メロディ生成部１０５は、＃０、＃１、＃２のコード進行候補指示データ１０９のいずれかに対応するコード進行候補を自動的に順番に選択するようにしてもよい。この結果、メロディ生成部１０５は、選択されたコード進行候補に対応する曲構造データを、伴奏・コード進行ＤＢ１０３から読み込む。メロディ生成部１０５は、この曲構造データによって示される小節のフレーズごとに、入力モチーフ１０８とフレーズセットＤＢ１０６に登録されているフレーズセット、およびルールＤＢ１０４を参照しながら、そのフレーズのメロディを自動生成する。メロディ生成部１０５は、楽曲全体の小節にわたってメロディの自動生成処理を実行し、自動生成されたメロディ１１０を出力する。

【００１４】

出力部１０７は、メロディ生成部１０５が自動生成したメロディデータ１１０に基づいてメロディの楽譜を表示する楽譜表示部１０７－１と、メロディデータ１１０および伴奏・コード進行ＤＢ１０３から取得した伴奏用ＭＩＤＩ（Ｍｕｓｉｃａｌ　Ｉｎｓｔｒｕｍｅｎｔ　Ｄｉｇｉｔａｌ　Ｉｎｔｅｒｆａｃｅ）データとに基づいて、メロディおよび伴奏の再生を実行する楽音再生部１０７－２とを備える。

【００１５】

次に、図１の機能構成を有する自動作曲装置１００の動作の概略について説明する。図２は、本実施形態において自動作曲される楽曲の構造例を示す図である。楽曲は通常、イントロ、Ａメロ、Ｂメロ、間奏、Ｃメロ（サビメロ）、エンディングなどのフレーズから構成される。イントロは、メロディが開始する前の伴奏のみからなる前奏部分である。Ａメロは、通常、イントロの次に出てくるフレーズをいい、曲の中で一般には落ち着いたメロディが奏でられる。Ｂメロは、Ａメロの次にでてくるフレーズをいい、Ａメロより少し盛り上がった曲調になることが多い。Ｃメロは、Ｂメロの次に出てくるフレーズの場合が多く、日本の曲だとＣメロが曲で一番盛り上がるサビメロになる場合が多い。エンディングは、イントロの逆で、曲の終わりのフレーズをいう。間奏は、例えば１曲目と２曲目の間のメロディの存在しない楽器演奏のみのフレーズである。図２に示される楽曲の構造例では、イントロ、Ａメロ、Ｂメロ、Ａメロ、間奏、Ａメロ、Ｂメロ、Ｃメロ、エンディングの順に楽曲が構成されている。

【００１６】

本実施形態では、ユーザは例えば、楽曲中で最初に現れるＡメロの例えば先頭２小節のメロディを、モチーフ入力部１０１（図１参照）から、図２（ａ）のモチーフＡとして（図１の入力モチーフ１０８の一例）として入力することができる。または、ユーザは例えば、楽曲中で最初に現れるＢメロの例えば先頭２小節のメロディを、モチーフ入力部１０１（図１参照）から、図２（ｂ）のモチーフＢ（図１の入力モチーフ１０８の他の一例）として入力モチーフ１０８として入力することができる。あるいは、ユーザは例えば、楽曲中で最初に現れるＣメロ（サビメロ）の例えば先頭２小節のメロディを、モチーフ入力部１０１（図１参照）から、図２（ｃ）のモチーフＣ（図１の入力モチーフ１０８のさらに他の一例）として入力することができる。

【００１７】

図３（ａ）は、上述のように入力される入力モチーフ１０８の音符例を示す図である。このように、入力モチーフ１０８としては、例えば２小節分のメロディが指定される。

【００１８】

このような入力に対して、コード進行選択部１０２（図１参照）が、伴奏・コード進行ＤＢ１０３に登録されているコード進行データの中から、例えば上位３位まで適合するコードとキー、スケールとからなるコード進行データを抽出する。コード進行データを構成するコードおよびキー、スケールは、図２（ｆ）および（ｇ）に示されるように、楽曲全体にわたって設定されている。

【００１９】

図３（ｂ）は、上位３位までのコード進行データによって表されるコード進行（コードおよびキー、スケール）＃０、＃１、＃２の例を示す図である。

【００２０】
図１のメロディ生成部１０５は、これらの情報に基づいて、入力モチーフ１０８が入力された図２（ａ）、（ｂ）、または（ｃ）のいずれかのフレーズ部分以外の図２（ｄ）に示されるフレーズ部分に対応するメロディを自動生成し、入力モチーフ１０８のメロディとともにメロディ１１０として出力する。そして、図１の出力部１０７が、自動生成されたメロディ１１０に対応する楽譜表示または放音を行う。なお、伴奏については、伴奏・コード進行ＤＢ１０３において最終的に選択されたコード進行に対応して登録されている伴奏用ＭＩＤＩデータが順次読み出されて、そのデータに基づいて図２（ｅ）に示されるように楽曲全体にわたり伴奏が行われる。
【００２１】
図４は、図１のモチーフ入力部１０１において、ユーザ入力に基づいて生成される入力モチーフ１０８のデータ構成例を示す図である。図４（ａ）に示されるように、入力モチーフ１０８は、＃０、＃１、・・・という複数のノートデータによって構成され、最後に終端コードが記憶される。各ノートデータは、例えば図３（ａ）に例示される入力モチーフ１０８を構成する例えば２小節分の音符のそれぞれに対応し、モチーフとなるメロディ音の発音を指示するデータである。図４（ｂ）に示されるように、１つのノートデータは、そのノートデータに対応する音符の発音タイミングを例えば入力モチーフ１０８の先頭からの経過時間として示す「時間」データと、音符の長さを示す「長さ」データと、音符の強さを示す「強さ」データと、音符の音高を示す「ピッチ」データとから構成される。これらのデータによって、図３（ａ）に例示されるような２小節分の入力モチーフ１０８中の１つの音符が表現される。
【００２２】
図５は、図１の伴奏・コード進行ＤＢ１０３のデータ構成例を示す図である。図５（ａ）に示されるように、コード進行ＤＢには、１つのレコード（図５（ａ）の１行）がコード進行データ、伴奏用ＭＩＤＩデータ、および曲構造データとからなる、＃０、＃１、・・・という複数レコードが記憶され、最後に終端コードが記憶される。
【００２３】
１レコード中のコード進行データは、楽曲の１曲分のコード進行を示している。図５（ａ）に示されるコード進行ＤＢには例えば、５０レコード＝５０曲分のコード進行データが記憶されている。１レコード中（＝１曲分）のコード進行データは、図５（ｂ）に示されるように、＃０、＃１、・・・という複数のコードデータから構成され、最後に終端コードが記憶される。コードデータには、あるタイミングにおけるキーおよびスケールを指定するデータ（図５（ｃ））と、あるタイミングにおけるコードを指定するデータ（図５（ｄ））とがある（図３（ｂ）参照）。キーおよびスケールを指定するデータは、図５（ｃ）に示されるように、そのキーおよびスケールが始まるタイミングを示す「時間」データと、「キー」データと、「スケール」データとから構成される。コードを指定するデータは、図５（ｄ）に示されるように、そのコードが始まるタイミングを示す「時間」データと、コードの根音（ルート）を示す「ルート」データ、およびコードのタイプ（種類）を示す「タイプ」データとから構成される。コード進行データは例えば、ＭＩＤＩ規格のメタデータとして記憶される。
【００２４】
図５（ａ）に示される伴奏・コード進行ＤＢ１０３の１レコード中（＝１曲分）の曲構造データは、図６に示されるデータ構成例を有する。この曲構造データは、１曲中の小節ごとに１レコード（図６の１行）を形成する。曲構造データ中の１レコードには、その小節に対応するフレーズの種別およびそのフレーズにメロディが存在するか否かを示す情報が記憶される。
【００２５】
図６に示される曲構造データにおいて、「Ｍｅａｓｕｒｅ」項目には、各レコードのデータが楽曲中の何小節目であるかを示す値が登録される。以降、「Ｍｅａｓｕｒｅ」項目の値がＭであるレコードを第Ｍレコード、そのレコードが示す小節を第Ｍ＋１小節とする

。例えば「Ｍｅａｓｕｒｅ」項目の値が０であるときそのレコードは第０レコード／第１小節、その値が１であるときそのレコードは第１レコード／第２小節である。

【００２６】

図６に示される曲構造データにおいて、「ＰａｒｔＮａｍｅ［Ｍ］」項目および「ｉＰａｒｔＩＤ［Ｍ］」項目（「Ｍ」は「Ｍｅａｓｕｒｅ」項目の値）にはそれぞれ、第Ｍレコード／第Ｍ＋１小節のフレーズの種別およびその種別に対応する識別値を示すデータが登録される。例えば、第０レコード（第１小節）の「ＰａｒｔＮａｍｅ［Ｍ］」項目および「ｉＰａｒｔＩＤ［Ｍ］」項目の値「Ｎｕｌｌ」および「０」は、その小節が無音であることを示している。第１、２レコード（第２，３小節）の「ＰａｒｔＮａｍｅ［Ｍ］」項目および「ｉＰａｒｔＩＤ［Ｍ］」項目の値「Ｉｎｔｒｏ」および「１」は、その小節がイントロフレーズであることを示している。第３～１０、２８～３４レコード（第４～１１、２９～３５小節）の「ＰａｒｔＮａｍｅ［Ｍ］」項目および「ｉＰａｒｔＩＤ［Ｍ］」項目の値「Ａ」および「１１」は、その小節がＡメロのフレーズであることを示している。第１１～１８レコード（第１２～１９小節）の「ＰａｒｔＮａｍｅ［Ｍ］」項目および「ｉＰａｒｔＩＤ［Ｍ］」項目の値「Ｂ」および「１２」は、その小節がＢメロのフレーズであることを示している。第１９～２７レコード（第２０～２８小節）の「ＰａｒｔＮａｍｅ［Ｍ］」項目および「ｉＰａｒｔＩＤ［Ｍ］」項目の値「Ｃ」および「１３」は、その小節がＣメロ（またはサビメロディ）のフレーズであることを示している。第３５レコード（第３６小節）の「ＰａｒｔＮａｍｅ［Ｍ］」項目および「ｉＰａｒｔＩＤ［Ｍ］」項目の値「Ｅｎｄｉｎｇ」および「３」は、その小節がエンディングのフレーズであることを示している。

【００２７】

また、図６に示される曲構造データにおいて、「ＥｘｉｓｔＭｅｌｏｄｙ［Ｍ］」項目（「Ｍ」は「Ｍｅａｓｕｒｅ」項目の値）には、第Ｍレコード（第Ｍ＋１小節）のフレーズにメロディが存在するか否かを示す値が登録される。メロディが存在するならば値「１」が、存在しないならば値「０」が登録される。例えば、Ｍ＝０、１、２、または３５（第０、１、２、３５レコード（第１、２、３、３６小節））である「ＰａｒｔＮａｍｅ［Ｍ］」項目が「Ｎｕｌｌ」、「Ｉｎｔｒｏ」、または「Ｅｎｄｉｎｇ」の各フレーズの「ＥｘｉｓｔＭｅｌｏｄｙ［Ｍ］」項目には値「０」が登録されて、メロディが存在しないことが示される。ＰａｒｔＮａｍｅ［Ｍ］＝「Ｎｕｌｌ」の場合は無音で、ＰａｒｔＮａｍｅ［Ｍ］＝「Ｉｎｔｒｏ」、または「Ｅｎｄｉｎｇ」の場合は伴奏のみが存在する。

【００２８】

また、図６に示される曲構造データにおいて、「ｉＰａｒｔＴｉｍｅ［Ｍ］」項目（「Ｍ」は「Ｍｅａｓｕｒｅ」項目の値）には、第Ｍレコードに対応する第Ｍ＋１小節の小節開始時間データが登録される。図６中では空欄になっているが、各レコードに実際の時間値が格納される。

以上の図６に示される曲構造データは例えば、ＭＩＤＩ規格のメタデータとして記憶される。

【００２９】

図２で前述したように、ユーザは例えば、図６の曲構造データで最初に現れるＡメロの例えば先頭２小節である第３、４レコード（第４、５小節）のメロディを、モチーフＡ（図２（ａ）参照）として、モチーフ入力部１０１（図１参照）から入力できる。または、ユーザは例えば、図６の曲構造データで最初に現れるＢメロの例えば先頭２小節である第１１、１２レコード（第１２、１３小節）のメロディを、モチーフＢ（図２（ｂ）参照）として、モチーフ入力部１０１から入力できる。あるいは、ユーザは例えば、図６の曲構造データで最初に現れるＣメロ（サビメロ）の例えば先頭２小節である第１９、２０レコード（第２０、２１小節）のメロディを、モチーフＣ（図２（ｃ）参照）として、モチーフ入力部１０１から入力できる。

【００３０】

コード進行選択部１０２は、伴奏・コード進行ＤＢ１０３に記憶されているコード進行

データごと（以下「評価対象のコード進行データ」と記載する）に、その評価対象のコード進行データがモチーフ入力部１０１から入力された入力モチーフ１０８にどの程度適合しているかを示す適合度を算出する。

【００３１】

本実施形態では、入力モチーフ１０８に対する評価対象のコード進行データの適合度を、音楽理論におけるアヴェイラブルノートスケールの概念を使って算出する。アヴェイラブルノートスケールは、コード進行が与えられたときに、メロディに使うことが可能な音を音階として表したものである。アヴェイラブルノートスケールを構成するノートの種類（以下、「ノートタイプ」と呼ぶ）としては、例えば、コードトーン、アヴェイラブルノート、スケールノート、テンションノート、アヴォイドノートがある。コードトーンは、スケールの元となるコードの構成音であって、メロディとして１音は用いることが望ましいノートタイプである。アヴェイラブルノートは、メロディに一般的に使用可能なノートタイプである。スケールノートは、スケールの構成音であり、その音を長い音などで加えると、元々のコードサウンドとぶつかってしまうので、取り扱いに注意を要するノートタイプである。テンションノートは、コード音にかぶせられる、コードのテンションで用いられている音で、高次のテンションほどサウンドの緊張感が増したり色彩豊かなサウンドになるノートタイプである。アヴォイドノートは、コードと不協和な音で、使用を避けるか、短い音符で用いることが望ましいとされるノートタイプである。本実施形態では、入力モチーフ１０８を構成する各ノート（図３（ａ）の各音符）について、そのノートの発音タイミングに対応する評価対象のコード進行データ中のキーおよびスケールとコードの根音およびコードタイプとに基づいて、そのノートの当該コード進行上でのノートタイプが算出される。

【００３２】

上述した、入力モチーフ１０８を構成する各ノート（図３（ａ）の各音符）のノートタイプを取得するために、本実施形態では、標準ピッチクラスセットテーブルが使用される。図７は、標準ピッチクラスセットテーブルのデータ構成例を示す図である。標準ピッチクラスセットテーブルはコード進行選択部１０２内のメモリ領域（例えば後述する図４のＲＯＭ１４０２内）に置かれる。標準ピッチクラステーブルは、図７（ａ）に例示されるコードトーンテーブル、図７（ｂ）に例示されるテンションノートテーブル、および図７（ｃ）に例示されるスケールノートテーブルから構成される。

【００３３】

図７（ａ）、（ｂ）、または（ｃ）のテーブルにおいて、その１行に対応する１組のピッチクラスセットは、コードまたはスケールの根音を第０音（第０ビット目）の音階構成音としたときの１オクターブ分の半音階を構成する第０音（第０ビット目）（図中の行の右端）から第１１音（第１１ビット目）（図中の行の左端）の音階構成音のそれぞれに対して、「０」または「１」の値が与えられる、合計１２ビットのデータで構成される。１組のピッチクラスセットにおいて、値「１」が与えられた音階構成音はそれがピッチクラスセットの構成要素に含まれ、値「０」が与えられた音階構成音はそれがピッチクラスセットの構成要素に含まれないことを示す。

【００３４】

図７（ａ）のコードトーンテーブル内の各行に対応するピッチクラスセット（以下、「コードトーンピッチクラスセット」と呼ぶ）は、その右端に記載されているコードタイプについて、そのコード根音が第０音（第０ビット目）の音階構成音として与えられたときに、どの音階構成音がそのコードタイプのコード構成音であるかを記憶する。例えば、図７（ａ）に例示されるコードトーンテーブルの１行目において、コードトーンピッチクラスセット「０００１００１０００１」は、第０音（第０ビット目）、第４音（第４ビット目）、および第７音（第７ビット目）の各音階構成音がコードタイプ「ＭＡＪ」のコード構成音であることを表わしている。

【００３５】

図１のコード進行選択部１０２は、入力モチーフ１０８を構成するノートごと（以下、

このノートを「現在ノート」と呼ぶ）に、その現在ノートのピッチが、その現在ノートの発音タイミングに対応する評価対象のコード進行データ中のコード根音に対して、どの音程（以下、これを「コード音程」と呼ぶ）を有するかを算出する。このとき、コード進行選択部１０２は、現在ノートのピッチを、その現在ノートの発音タイミングに対応する評価対象のコード進行データ中のコード根音を第０音の音階構成音としたときの、第０音から第１１音までの１オクターブ内の音階構成音のいずれかに写像させる演算を行い、その写像位置の音（第０音から第１１音のいずれか）を、上記コード音程として算出する。そして、コード進行選択部１０２は、上記発音タイミングにおける評価対象のコード進行データ中のコードタイプに対応する図７（ａ）に例示されるコードトーンテーブル上のコードトーンピッチクラスセットのコード構成音に、上記算出されたコード音程が含まれるか否かを判定する。

【００３６】

図７（ｂ）のテンションノートテーブル内の各行に対応するピッチクラスセット（以下、「テンションノートピッチクラスセット」と呼ぶ）は、その右端に記載されているコードタイプについて、そのコード根音が第０音（第０ビット目）の音階構成音として与えられたときに、どの音階構成音がそのコードタイプに対するテンションであるかを記憶する。例えば、図７（ｂ）に例示されるテンションノートテーブルの１行目において、テンションノートピッチクラスセット「００１００１０００１００」は、第２音（第２ビット目）、第６音（第６ビット目）、および第９音（第９ビット目）がコードタイプ「ＭＡＪ」（コード根音＝Ｃ）に対するテンションであることを表わしている。

【００３７】

図１のコード進行選択部１０２は、現在ノートの発音タイミングにおける評価対象のコード進行データ中のコードタイプに対応する図７（ｂ）に例示されるテンションノートテーブル上のテンションノートピッチクラスセットのテンションノートに、前述した現在ノートのピッチのコード根音に対するコード音程が含まれるか否かを判定する。

【００３８】

図７（ｃ）のスケールノートテーブル内の各行に対応するピッチクラスセット（以下、「スケールノートピッチクラスセット」と呼ぶ）は、その右端に記載されているスケールについて、そのスケールの根音が第０音（第０ビット目）の音階構成音として与えられたときに、どの音階構成音がそのスケールに対応するスケール構成音であるかを記憶する。例えば、図７（ｃ）に例示されるスケールノートテーブルの１行目において、スケールノートピッチクラスセット「１０１０１０１１０１０１」は、第０音（第０ビット目）、第２音（第２ビット目）、第４音（第４ビット目）、第５音（第５ビット目）、第７音（第７ビット目）、第９音（第９ビット目）、および第１１音（第１１ビット目）がスケール「ダイアトニック」のスケール構成音であることを表している。

【００３９】

図１のコード進行選択部１０２は、現在ノートのピッチが、その現在ノートの発音タイミングに対応する評価対象のコード進行データ中のキーに対して、どの音程（以下、これを「キー音程」と呼ぶ）を有するかを算出する。このとき、コード進行選択部１０２は、コード音程の算出の場合と同様に、現在ノートのピッチを、その現在ノートの発音タイミングに対応する評価対象のコード進行データ中のキーを第０音の音階構成音としたときの、第０音から第１１音までの１オクターブ内の音階構成音のいずれかに写像させる演算を行い、その写像位置の音を、上記キー音程として算出する。そして、コード進行選択部１０２は、上記発音タイミングにおける評価対象のコード進行データ中のスケールに対応する図７（ｃ）に例示されるスケールノートテーブル上のスケールノートピッチクラスセットのスケール構成音に、上記算出されたキー音程が含まれるか否かを判定する。

【００４０】

以上のようにして、コード進行選択部１０２は、入力モチーフ１０８の現在ノートの発音タイミングにおける評価対象のコード進行データ中のコードタイプに対応する図７（ａ）に例示されるコードトーンテーブル上のコードトーンピッチクラスセットのコード構成

音にコード音程が含まれるか否かを判定する。また、コード進行選択部１０２は、上記コードタイプに対応する図７（ｂ）に例示されるテンションノートテーブル上のテンションノートピッチクラスセットのテンションノートにコード音程が含まれるか否かを判定する。さらに、コード進行選択部１０２は、評価対象のコード進行データ中のスケールに対応する図７（ｃ）に例示されるスケールノートテーブル上のスケールノートピッチクラスセットのスケール構成音にキー音程が含まれるか否かを判定する。そして、コード進行選択部１０２は、これらの判定に基づいて、現在ノートが、コードトーン、アヴェイラブルノート、スケールノート、テンションノート、またはアヴォイドノートのいずれに該当するか、すなわちノートタイプの情報を取得する。ノートタイプ取得処理の詳細については、図２２の説明において詳述する。

【００４１】

図８（ａ）は、図３（ａ）に例示される入力モチーフ１０８の各ノートのピッチ（図８（ａ）中の灰色の部分）ごとに、図１の伴奏・コード進行ＤＢ１０３から読み出される図３（ｂ）に例示される＃０、＃１、＃２の３つの評価対象のコード進行データの例のそれぞれに対して、コード進行選択部１０２が取得するノートタイプの例を示す図である。図８（ａ）において、「Ｃ」はコードトーン、「Ａ」はアヴェイラブルノート、「Ｓ」はスケールノート、「Ｖ」はアヴォイドノートの、ノートタイプをそれぞれ示す値である。また、図示していないが、「Ｔ」はテンションノートのノートタイプを示す値である。なお、この図では、表記の簡略化のために、各ノートタイプを示す値をアルファベット１文字で表しているが、実際のメモリに記憶される各ノートタイプの値としては例えば、コードトーンを示す定数値としてｃｉ＿ＣｈｏｒｄＴｏｎｅ（表記「Ｃ」と等価）、アヴェイラブルノートを示す定数値としてｃｉ＿ＡｖａｉｌａｂｌｅＮｏｔｅ（表記「Ａ」と等価）、スケールノートを示す定数値としてｃｉ＿ＳｃａｌｅＮｏｔｅ（表記「Ｓ」と等価）、テンションノートを示す定数値としてｃｉ＿ＴｅｎｓｉｏｎＮｏｔｅ（表記「Ｔ」と等価）、アヴォイドノートを示す定数値としてｃｉ＿ＡｖｏｉｄＮｏｔｅ（表記「Ｖ」と等価）が用いられる（後述する図１５Ａ参照）。

【００４２】

次に、コード進行選択部１０２は、入力モチーフ１０８の各ノートのピッチごとに、隣接するピッチ間の半音単位の音程（以下、「隣接音程」と呼ぶ）を算出する。図８（ｂ）の「隣接音程」は、入力モチーフ１０８各ノートのピッチ（図８（ｂ）中の灰色の部分）間の音程の算出結果の例を示す図である。

【００４３】

コード進行選択部１０２は、評価対象のコード進行データに対して、上述のように算出したノートタイプと隣接音程が交互に格納された配列変数データ（以下、この配列変数データを「ｉｎｃｏｎ［ｉ］」（「ｉ」は配列番号）と記載する）を生成する。図８（ｃ）は、図１の伴奏・コード進行ＤＢ１０３から読み出される図３（ｂ）に例示される＃０、＃１、＃２の３つの評価対象のコード進行データの例のそれぞれに対して算出された配列変数データｉｎｃｏｎ［ｉ］の例を示す図である。図８（ｃ）のコード進行＃０、＃１、＃２のそれぞれの配列変数データｉｎｃｏｎ［ｉ］において、偶数番目の配列番号ｉ＝０、２、４、６、８、１０、１２、１４、１６、１８の各要素には、図８（ａ）のコード進行＃０、＃１、＃２のそれぞれのノートタイプが先頭から順次コピーされる。また、コード進行＃０、＃１、＃２のそれぞれの配列変数データｉｎｃｏｎ［ｉ］において、奇数番目の配列番号ｉ＝１、３、５、７、９、１１、１３、１５、１７の各要素にはともに、図８（ｂ）の隣接音程が先頭から順次コピーされる。

【００４４】

次に、コード進行選択部１０２は、現在の評価対象のコード進行データに対して上述のように算出した入力モチーフ１０８の各ノートのノートタイプと隣接音程を格納した配列変数データｉｎｃｏｎ［ｉ］（ｉ＝０、１、２、３、・・・）において、配列番号０から順に例えば４組ずつ、ノートタイプと隣接音程の組合せの規則（以下、この規則を「ノート接続ルール」と呼ぶ）を評価するノート接続性チェック処理を実行する。このノート接

続性チェック処理において、コード進行選択部１０２は、図１のルールＤＢ１０４に記憶されているノート接続ルールを参照する。

【００４５】

図９は、ルールＤＢ１０４に記憶されるノート接続ルールのデータ構成例を示す図である。ノート接続ルールには、３音のルールと４音のルールがあり、説明の便宜上、それぞれに例えば、「コードトーン」、「刺繍音」、「経過音」、「倚音」、「逸音」などの名称を付けてある。また、各ノート接続ルールには、メロディを形成する上でどの程度適合しているかを評価するための評価点が付与されている。さらに、本実施形態では、ノート接続ルールを示す変数として、ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［２ｋ］（０≦ｋ≦３）およびｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［２ｋ＋１］（０≦ｋ≦２）という配列変数データを用いる。ここで、変数データ「ｊ」は、ルールＤＢ１０４におけるｊ番目（図９中ではｊ行目）のノート接続ルールのデータを指す。また、変数データ「ｋ」は、０から３までの値をとる。そして、ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［２ｋ］＝ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［０］、ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［２］、ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［４］、ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［６］にはそれぞれ、ｊ番目のノート接続ルールにおける１ノート目（ノートタイプ＃０）、２ノート目（ノートタイプ＃１）、３ノート目（ノートタイプ＃２）、および４ノート目（ノートタイプ＃３）の各ノートタイプが格納される。なお、４ノート目（ノートタイプ＃３）が「ｃｉ＿ＮｕｌｌＮｏｔｅＴｙｐｅ」となっているｊ＝０からｊ＝８までのノート接続ルールは、４ノート目のノートタイプは無いことを示しており、実質的に３音からなるノート接続ルールであることを示している。また、ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［２ｋ＋１］＝ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［１］、ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［３］、ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［５］にはそれぞれ、ｊ番目のノート接続ルールにおける１ノート目（＃０）と２ノート目（＃１）の隣接音程、２ノート目（＃１）と３ノート目（＃２）の隣接音程、および３ノート目（＃２）と４ノート目（＃３）の隣接音程が格納される。隣接音程の数値は、半音単位の音程を示し、プラス値は音程が上がることを示し、マイナス値は音程が下がることを示す。また、値「９９」は、音程がどの値でもよいことを示し、値「０」は音程が変化しないことを示す。なお、４ノート目（ノートタイプ＃３）が「ｃｉ＿ＮｕｌｌＮｏｔｅＴｙｐｅ」となっているｊ＝０からｊ＝８までのノート接続ルールは、前述したように４ノート目のノートタイプは無い（値が「ｃｉ＿ＮｕｌｌＮｏｔｅＴｙｐｅ」である）ため、３ノート目（＃２）と４ノート目（＃３）の隣接音程が格納されるｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［５］の値は「０」とされる。最後の、ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［７］には、ｊ番目のノート接続ルールの評価点が格納される。

【００４６】

以上のようなデータ構成を有するノート接続ルールとして、図９に例示されるようにｊ＝０からｊ＝１７までの１８ルールが、図１のルールＤＢ１０４に予め登録されている。

【００４７】

コード進行選択部１０２は、上記構成を有するノート接続ルールを用いて、ノート接続性チェック処理を実行する。コード進行選択部１０２は、図１０（ａ）に例示される２小節分の入力モチーフ１０８の先頭のノートから順に、図１０（ｂ）のｉ＝０～６に示されるように４ノートずつ、各ノートに対応して配列変数データｉｎｃｏｎ［ｉ］に格納されているノートタイプと隣接音程の組と、ｊ＝０からｊ＝１７までのノート接続ルールのよりｊ＝０から順に選択した１組のノート接続ルールのノートタイプと隣接音程の組とが一致するか否かを比較する。

【００４８】

例えば、コード進行選択部１０２は、図１０（ｂ）のｉ＝０では、ｉ＝０の右横の矢印で示されるように、入力モチーフ１０８の第１、２、３、４ノート目（図中では１音、２音、３音、４音目）のノートタイプおよび隣接音程の各組が、図９に例示されるｊ＝０、１、２、３、・・・の各ノート接続ルールの４組のノートタイプおよび隣接音程の組と一

致するか否かを比較する。

【００４９】

まず、図９に例示されるｊ＝０のノート接続ルールでは、＃０、＃１、および＃２のノートタイプがともにコードトーン（ｃｉ＿ＣｈｏｒｄＴｏｎｅ）となる。これに対して、例えば評価対象のコード進行データが図３（ｂ）に例示される＃０のコード進行である場合には、図３（ａ）に対応する図１０（ａ）の入力モチーフ１０８に対応するノートタイプと隣接音程の配列変数データｉｎｃｏｎ［ｉ］は、図８の説明で前述したように、図１０（ｃ）のコード進行＃０の右横に示されるデータとなる。従って、入力モチーフ１０８の第１、２、３、４ノート目のノートタイプは、コードトーン（Ｃ）、アヴェイラブルノート（Ａ）、コードトーン（Ｃ）となって、ｊ＝０のノート接続ルールとは一致しない。この場合には、ｊ＝０のノート接続ルールの評価点は加算されない。

【００５０】

次に、図９に例示されるｊ＝１のノート接続ルールでは、＃０、＃１、および＃２のノートタイプが、コードトーン（ｃｉ＿ＣｈｏｒｄＴｏｎｅ）、アヴェイラブルノート（ｃｉ＿ＡｖａｉｌａｂｌｅＮｏｔｅ）、コードトーン（ｃｉ＿ＣｈｏｒｄＴｏｎｅ）となる。これに対して、例えば評価対象のコード進行データが図３（ｃ）に例示される＃０のコード進行である場合には、図１０（ｃ）のコード進行＃０の右横に示されるノートタイプと隣接音程の配列変数データｉｎｃｏｎ［ｉ］より得られる、入力モチーフ１０８の第１、２、３、４ノート目のノートタイプと一致する。しかし、ｊ＝１のノート接続ルールにおける第１音（＃０）と第２音（＃１）の隣接音程は「－１」、第２音（＃１）と第３音（＃２）の隣接音程は「１」であり、これは、図１０（ｃ）のコード進行＃０の右横に示されるノートタイプと隣接音程の配列変数データｉｎｃｏｎ［ｉ］より得られる、入力モチーフ１０８の第１音と第２音間の隣接音程「－２」および第２音と第３音間の隣接音程「２」と一致しない。従って、ｊ＝１の場合もｊ＝０の場合と同様に、ノート接続ルールの評価点は加算されない。

【００５１】

次に、図９に例示されるｊ＝２のノート接続ルールでは、＃０、＃１、および＃２のノートタイプが、コードトーン（ｃｉ＿ＣｈｏｒｄＴｏｎｅ）、アヴェイラブルノート（ｃｉ＿ＡｖａｉｌａｂｌｅＮｏｔｅ）、コードトーン（ｃｉ＿ＣｈｏｒｄＴｏｎｅ）となる。これに対して、例えば評価対象のコード進行データが図３（ｃ）に例示される＃０のコード進行である場合には、図１０（ｃ）のコード進行＃０の右横に示されるノートタイプと隣接音程の配列変数データｉｎｃｏｎ［ｉ］より得られる、入力モチーフ１０８の第１、２、３、４ノート目のノートタイプと一致する。また、ｊ＝１のノート接続ルールにおける第１音（＃０）と第２音（＃１）の隣接音程は「－２」、第２音（＃１）と第３音（＃２）の隣接音程は「２」であり、これは、図１０（ｃ）のコード進行＃０の右横に示されるノートタイプと隣接音程の配列変数データｉｎｃｏｎ［ｉ］より得られる、入力モチーフ１０８の第１音と第２音間の隣接音程および第２音と第３音間の隣接音程と一致する。さらに、ｊ＝２のノート接続ルールの４ノート目（ノートタイプ＃３）は、ノートタイプが無いことを示す値「ｃｉ＿ＮｕｌｌＮｏｔｅＴｙｐｅ」であるため、入力モチーフ１０８の４ノート目は比較しなくてよい。以上より、評価対象のコード進行データが＃０である場合の入力モチーフ１０８の第１、２、３音が、図９のｊ＝２のノート接続ルールと適合することがわかり、ｊ＝２のノート接続ルールの評価点（ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［２］［７］）＝９０点が、評価対象のコード進行データ＃０に対応する総合評価点に加算される。図１０（ｃ）のコード進行＃０に記載されている「＜－　　Ｎｏ２：９０　　－＞」の表示が、その加算処理に対応する。

【００５２】

以上のようにして、ノート接続ルールが見つかると、そのノート接続ルール以降のノート接続ルールについては、図１０（ｂ）のｉ＝０の入力モチーフ１０８の第１、２、３、４ノートのノートタイプおよび隣接音程の組に対しての評価は実施されない。

【００５３】

図１０（ｂ）のｉ＝０の入力モチーフ１０８の第１、２、３、４ノートのノートタイプおよび隣接音程の組に対する評価が終了すると、入力モチーフ１０８上の評価対象のノートが１つ進められ、図１０（ｂ）のｉ＝１の状態になって、ｉ＝１の右横の矢印で示されるように、入力モチーフ１０８の第２、３、４、５ノート目のノートタイプおよび隣接音程の各組が、図９に例示されるｊ＝０、１、２、３、・・・の各ノート接続ルールの４組のノートタイプおよび隣接音程の組と一致するか否かが比較される。この結果、図１０（ｃ）の評価対象のコード進行データ＃０に対応する入力モチーフ１０８の第２、３、４、５ノート目のノートタイプおよび隣接音程の各組については、全てのノート接続ルールと一致せず、図１０（ｂ）のｉ＝１の入力モチーフ１０８の第２、３、４、５ノートのノートタイプおよび隣接音程の組に対する評価点は０点となって、評価対象のコード進行データ＃０に対応する総合評価点への加算は行われない。

【００５４】

図１０（ｂ）のｉ＝１の入力モチーフ１０８の第２、３、４、５ノートのノートタイプおよび隣接音程の組に対する評価が終了すると、入力モチーフ１０８上の評価対象のノートがさらに１つ進められ、図１０（ｂ）のｉ＝２の状態になって、ｉ＝２の右横の矢印で示されるように、入力モチーフ１０８の第３、４、５、６ノート目のノートタイプおよび隣接音程の各組が、図９に例示されるｊ＝０、１、２、３、・・・の各ノート接続ルールの４組のノートタイプおよび隣接音程の組と一致するか否かが比較される。この結果、図１０（ｃ）の評価対象のコード進行データ＃０に対応する入力モチーフ１０８の第３、４、５、６ノート目のノートタイプおよび隣接音程の各組については、図９のｊ＝３のノート接続ルールが適合することがわかり、ｊ＝３のノート接続ルールの評価点（ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［３］［７］）＝８０点が、評価対象のコード進行データ＃０に対応する総合評価点に加算される。図１０（ｃ）のコード進行＃０に記載されている「＜－　Ｎｏ３：８０　－＞」の表示が、その加算処理に対応する。この結果、総合評価点は、９０点＋８０点＝１７０点となる。

【００５５】

以降同様にして、図１０（ｂ）のｉ＝７の入力モチーフ１０８の第８、９、１０ノートのノートタイプおよび隣接音程の組に対する評価までが実行される。なお、本実施形態では評価は原則は４ノートずつ行われるが、最後のｉ＝７の場合のみ、入力モチーフ１０８の３ノートに対して、図９のｊ＝０からｊ＝８までのノートタイプ＃３が「ｃｉ＿ＮｕｌｌＮｏｔｅＴｙｐｅである」である３音のノート接続ルールが比較される。

【００５６】

以上のようにして、図１０（ｃ）の評価対象のコード進行データ＃０に対応する入力モチーフ１０８の各ノートごとの評価処理が終了すると、その時点で評価対象のコード進行データ＃０に対応して算出されている総合評価点が、その評価対象のコード進行データ＃０の入力モチーフ１０８に対する適合度とされる。

【００５７】

例えば評価対象のコード進行データが図３（ｃ）に例示される＃１または＃２の各コード進行である場合は、図３（ａ）に対応する図１０（ａ）の入力モチーフ１０８に対応するノートタイプと隣接音程の配列変数データｉｎｃｏｎ［ｉ］は、図８の説明で前述したように、図１０（ｃ）のコード進行＃１の右横に示されるデータまたは＃２の右横に示されるデータとなる。これらの配列変数データｉｎｃｏｎ［ｉ］についても上述したコード進行＃０の場合と同様の評価処理が実行される。例えば、コード進行＃１の場合は、図１０（ｃ）に示されるように、図９のノート接続ルールと適合する部分がないため、その総合評価点は０点となり、これがコード進行＃１の入力モチーフ１０８に対する適合度となる。また、コード進行＃２の場合は、図１０（ｃ）に示されるように、入力モチーフ１０８の第５、６、７ノート目のノートタイプおよび隣接音程の各組について、図９のｊ＝５のノート接続ルールが適合することがわかり、ｊ＝５のノート接続ルールの評価点（ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［５］［７］）＝９５点が、評価対象のコード進行データ＃２に対応する総合評価点に加算され、これがコード進行＃２の入力モチーフ１０８に対する

適合度となる。

【００５８】

図１のコード進行選択部１０２は、以上の適合度の算出処理を、伴奏・コード進行ＤＢ１０３に記憶されている複数のコード進行データに対して実行し、適合度が高かった例えば上位３個のコード進行データをそれぞれ指し示す＃０、＃１、＃２のコード進行候補指示データ１０９を出力する。なお、以上の処理において、入力モチーフ１０８と伴奏・コード進行ＤＢ１０３中の各コード進行データとは、キーが必ずしも一致しているとは限らないため、各コード進行データを１オクターブを構成する１２段階にキーシフトさせたデータが、入力モチーフ１０８と比較される。

【００５９】

次に、図１のメロディ生成部１０５の動作の概略について説明する。まず、図１１は、図１のフレーズセットＤＢ１０６のデータ構成例を示す図である。図１１（ａ）に示されるように、フレーズセットＤＢ１０６には、＃０、＃１、・・・という複数のフレーズセットデータのレコードが記憶され、最後に終端コードが記憶される。

【００６０】

１レコード分のフレーズセットデータは、図１１（ｂ）に示されるように、Ａメロデータ、Ｂメロデータ、Ｃメロ（サビメロディ）データ、エンディング１データ、エンディング２データの、複数のフレーズデータから構成される。

【００６１】

図１１（ｂ）の各フレーズデータは、図１１（ｃ）に示されるように、＃０、＃１、・・・という複数のノートデータによって構成され、最後に終端コードが記憶される。各ノートデータは、各フレーズを構成する１小節分以上の音符のそれぞれに対応し、各フレーズのメロディ音の発音を指示するデータである。図１１（ｄ）に示されるように、１つのノートデータは、そのノートデータに対応する音符の発音タイミングを例えばフレーズの先頭からの経過時間として示す「時間」データと、音符の長さを示す「長さ」データと、音符の強さを示す「強さ」データと、音符の音高を示す「ピッチ」データとから構成される。これらのデータによって、フレーズを構成する各音符が表現される。

【００６２】

図１のメロディ生成部１０５は、コード進行選択部１０２が出力した＃０、＃１、＃２のコード進行候補指示データ１０９に対応する３つのコード進行候補のうちの１つから、ユーザ指定によりまたは自動的に選択されたコード進行候補に対応する曲構造データ（図６参照）を、伴奏・コード進行ＤＢ１０３から読み込む。メロディ生成部１０５は、この曲構造データによって示される小節のフレーズごとに、入力モチーフ１０８とフレーズセットＤＢ１０６に登録されているフレーズセット（図１１参照）、およびルールＤＢ１０４（図９参照）を参照しながら、そのフレーズのメロディを自動生成する。

【００６３】

この場合、メロディ生成部１０５は、曲構造データによって示される小節のフレーズが、入力モチーフ１０８が入力されたフレーズであるか否かを判定し、入力モチーフ１０８のフレーズである場合は、その入力モチーフ１０８のメロディをそのまま図１のメロディ１１０の一部として出力する。

【００６４】

メロディ生成部１０５は、曲構造データによって示される小節のフレーズが、入力モチーフ１０８のフレーズでもなく、サビメロディの先頭フレーズでもない場合は、該当するフレーズのメロディが未だ生成されていなければ、フレーズセットＤＢ１０６から入力モチーフ１０８に対応するフレーズセットを抽出し、そのフレーズセット内の該当するフレーズのメロディをコピーし、生成済みであればその生成済みのフレーズからメロディをコピーする。そして、メロディ生成部１０５は、コピーしたメロディを変形する後述するメロディ変形処理と、さらにその変形したメロディを構成する各ノートのピッチを最適化する後述するメロディ最適化処理を実行して、曲構造データによって示される小節のフレーズのメロディを自動生成し、メロディ１１０の一部として出力する。既に生成済みのフレ

ーズからメロディをコピーする処理の詳細については、図２５の説明において後述する。
【００６５】
メロディ生成部１０５は、曲構造データによって示される小節のフレーズが、サビメロディの先頭フレーズである場合は、該当するサビメロディの先頭フレーズが生成済みでなければ、フレーズセットＤＢ１０６から入力モチーフ１０８に対応するフレーズセットを抽出し、そのフレーズセット内の該当するサビメロディ（Ｃメロ）の先頭フレーズのメロディをコピーし、そのメロディを構成する各ノートのピッチを最適化するメロディ最適化処理を実行して、サビメロディの先頭フレーズのメロディを自動生成し、メロディ１１０の一部として出力する。一方、該当するサビメロディの先頭フレーズが生成済みならば、その生成済みのフレーズからメロディをコピーし、メロディ１１０の一部として出力する。
【００６６】
図１２は、メロディ変形処理およびメロディ最適化処理の動作説明図である。予め生成されているメロディがあるときに、メロディ生成部１０５は、そのメロディをコピーして、例えば１２０１に示されるように、コピーしたメロディを構成する各ノートのピッチを、例えば2半音上にピッチシフトする処理を実行する。あるいは、メロディ生成部１０５は、例えば１２０２に示されるように、コピーしたメロディを構成する各ノートを、小節内で左右（再生順序）を反転させる処理を実行する。メロディ生成部１０５は、このようなメロディ変形処理を実行した小節のメロディに対して、さらに１２０３または１２０４として示されるメロディ最適化処理を実行して、最終的なメロディを自動生成する。
【００６７】
図１３は、メロディ最適化処理の詳細動作説明図である。いま、変数ｉＮｏｔｅＣｎｔには、メロディ変形処理を実行した小節のメロディを構成するノートの数が格納されており、配列データｎｏｔｅ［0］－＞ｉＰｉｔ、ｎｏｔｅ［1］－＞ｉＰｉｔ、ｎｏｔｅ［2］－＞ｉＰｉｔ、・・・、ｎｏｔｅ［ｉＮｏｔｅＣｎｔ－2］－＞ｉＰｉｔ、ｎｏｔｅ［ｉＮｏｔｅＣｎｔ－1］－＞ｉＰｉｔには、上記各ノートのピッチデータが格納されているとする。メロディ生成部１０５はまず、各ノートのピッチデータｎｏｔｅ［ｉ］－＞ｉＰｉｔ（0≦ｉ≦ｉＮｏｔｅＣｎｔ－1）をそれぞれ、ｉｐｉｔｄ［0］＝0、ｉｐｉｔｄ［1］＝1、ｉｐｉｔｄ［2］＝－1、ｉｐｉｔｄ［3］＝2、ｉｐｉｔｄ［4］＝－2という５段階の値だけピッチシフトさせ、合計$5^{iNoteCnt}$通りのピッチ列を生成する。そして、メロディ生成部１０５は、各ピッチ列ごとに、図７から図１０を用いて前述したのと同様の処理によって、コード進行選択部１０２が抽出しているコード進行データの上記小節に対応する部分について、ノートタイプの取得と、隣接音程の計算を実行し、ノート接続性チェック処理を実行する。この結果、メロディ生成部１０５は、合計$5^{iNoteCnt}$通りのピッチ列に対して算出した適合度のうち、もっとも適合度が高いピッチ列を、その小節の各ノートのピッチデータｎｏｔｅ［ｉ］－＞ｉＰｉｔ（0≦ｉ≦ｉＮｏｔｅＣｎｔ－1）として修正する。メロディ生成部１０５は、このようにして生成したピッチ列を含むその小節の各ノートのデータｎｏｔｅ［ｉ］（0≦ｉ≦ｉＮｏｔｅＣｎｔ－1）をメロディ１１０として出力する。
【００６８】
上述した自動作曲装置１００のさらに詳細な構成および動作について、以下に説明する。図１４は、図１の自動作曲装置１００のハードウェア構成例を示す図である。図１４に例示される自動作曲装置１００のハードウェア構成は、ＣＰＵ（中央演算処理装置）１４０１、ＲＯＭ（リードオンリーメモリ）１４０２、ＲＡＭ（ランダムアクセスメモリ）１４０３、入力手段１４０４、表示部１４０５、および音源部１４０６を備え、それらがシステムバス１４０８によって相互に接続された構成を有する。また、音源部１４０６の出力はサウンドシステム１４０７に入力する。
【００６９】
ＣＰＵ１４０１は、ＲＡＭ１４０３をワークメモリとして使用しながらＲＯＭ１４０２に記憶された自動作曲制御プログラムを実行することにより、図１の１０１～１０７の各

機能部分に対応する制御動作を実行する。

【００７０】

ＲＯＭ１４０２には、上記自動作曲制御プログラムのほか、図１の伴奏・コード進行ＤＢ１０３（図５、図６参照）、ルールＤＢ１０４（図９参照）、フレーズセットＤＢ１０６（図１１参照）、および標準ピッチクラスセットテーブル（図７参照）が予め記憶される。

【００７１】

ＲＡＭ１４０３は、モチーフ入力部１０１から入力された入力モチーフ１０８（図４参照）、コード進行選択部１０２が出力するコード進行候補データ１０９、メロディ生成部１０５が出力するメロディデータ１１０などを一時的に記憶する。このほか、ＲＡＭ１４０３には、後述する各種変数データ等が一時的に記憶される。

【００７２】

入力手段１４０４は、図１のモチーフ入力部１０１の一部の機能に対応し、例えば、鍵盤入力部１０１－１、音声入力部１０１－２、または音符入力部１０１－３に対応する。入力手段１４０４が鍵盤入力部１０１－１を備える場合には、演奏鍵盤と、当該演奏鍵盤の押鍵状態を検知しシステムバス１４０８を介してＣＰＵ１４０１に通知するキーマトリクス回路を備える。入力手段１４０４が音声入力部１０１－２を備える場合には、歌声入力用のマイクと、当該マイクから入力された音声信号をデジタル信号に変換した後、歌声のピッチ情報を抽出しシステムバス１４０８を介してＣＰＵ１４０１に通知するデジタル信号処理回路を備える。なお、ピッチ情報の抽出は、ＣＰＵ１４０１が実行してもよい。入力手段１４０４が音符入力部１０１－３を備える場合には、音符入力用のキーボードと、当該キーボードの音符入力状態を検知しシステムバス１４０８を介してＣＰＵ１４０１に通知するキーマトリクス回路を備える。ＣＰＵ１４０１は、図１のモチーフ入力部１０１の一部の機能に対応し、図１４の入力手段１４０４から入力した上記各種情報に基づいて、入力モチーフ１０８を検出してＲＡＭ１４０３に記憶する。

【００７３】

表示部１４０５は、ＣＰＵ１４０１による制御動作とともに、モチーフの入力を図１の出力部１０７が備える楽譜表示部１０７－１の機能を実現する。ＣＰＵ１４０１は、自動作曲されたメロディデータ１１０に対応する楽譜データを生成し、その楽譜データの表示を表示部１４０５に指示する。表示部１４０５は、例えば液晶ディスプレイ装置である。

【００７４】

音源部１４０６は、ＣＰＵ１４０１による制御動作とともに、図１の楽音再生部１０７－２の機能を実現する。ＣＰＵ１４０１は、自動生成されたメロディデータ１１０と伴奏・コード進行ＤＢ１０３から読み出された伴奏用ＭＩＤＩデータとに基づいて、メロディおよび伴奏を再生するための発音制御データを生成し、音源部１４０６に供給する。音源部１４０６は、この発音制御データに基づいて、メロディ音および伴奏音を生成し、サウンドシステム１４０７に出力する。サウンドシステム１４０７は、音源部１４０６から入力したメロディ音および伴奏音のデジタル楽音データをアナログ楽音信号に変換した後、そのアナログ楽音信号を内蔵のアンプで増幅して内蔵のスピーカから放音する。

【００７５】

図１５Ａおよび図１５Ｂは、ＲＯＭ１４０２またはＲＡＭ１４０３に記憶される各種変数データ、配列変数データ、および定数データのリストを示す図である。これらのデータは、後述する各種処理で使用される。

【００７６】

図１６は、本実施形態における自動作曲処理の例を示すフローチャートである。この処理は、自動作曲装置１００の電源が投入されることにより、ＣＰＵ１４０１がＲＯＭ１４０２に記憶されている自動作曲処理プログラムの実行を開始することによりスタートする。

【００７７】

ＣＰＵ１４０１はまず、ＲＡＭ１４０３および音源部１４０６に対して初期化を行う（

ステップＳ１６０１）。その後、ＣＰＵ１４０１は、ステップＳ１６０２からＳ１６０８までの一連の処理を繰返し実行する。

【００７８】

この繰返し処理において、ＣＰＵ１４０１はまず、ユーザが特には図示しない電源スイッチを押下したことにより自動作曲処理の終了を指示したか否かを判定し（ステップＳ１６０２）、終了を指示していなければ（ステップＳ１６０２の判定がＮＯ）、繰返し処理を継続し、終了を指示したならば（ステップＳ１６０２の判定がＹＥＳ）、図１６のフローチャートで例示される自動作曲処理を終了する。

【００７９】

ステップＳ１６０２の判定がＮＯの場合、ＣＰＵ１４０１は、ユーザが入力手段１４０４からモチーフ入力を指示したか否かを判定する（ステップＳ１６０３）。ユーザがモチーフ入力を指示した場合（ステップＳ１６０３の判定がＹＥＳの場合）、ＣＰＵ１４０１は、入力手段１４０４からのユーザによるモチーフ入力を受け付け、その結果入力手段１４０４から入力された入力モチーフ１０８を、例えば図４のデータ形式でＲＡＭ１４０３に記憶する（ステップＳ１６０６）。その後、ＣＰＵ１４０１は、ステップＳ１６０２の処理に戻る。

【００８０】

ユーザがモチーフ入力を指示していない場合（ステップＳ１６０３の判定がＮＯの場合）、ＣＰＵ１４０１は、ユーザが特には図示しないスイッチにより自動作曲を指示したか否かを判定する（ステップＳ１６０４）。ユーザが自動作曲を指示した場合（ステップＳ１６０４の判定がＹＥＳの場合）、ＣＰＵ１４０１は、コード進行選択処理（ステップＳ１６０７）、続いてメロディ生成処理（ステップＳ１６０８）を実行する。ステップＳ１６０７のコード進行選択処理は、図１のコード進行選択部１０２の機能を実現する。ステップＳ１６０８のメロディ生成処理は、図１のメロディ生成部１０５の機能を実現する。その後、ＣＰＵ１４０１は、ステップＳ１６０２の処理に戻る。

【００８１】

ユーザが自動作曲を指示していない場合（ステップＳ１６０４の判定がＮＯの場合）、ＣＰＵ１４０１は、ユーザが特には図示しないスイッチにより自動作曲されたメロディ１１０の再生を指示したか否かを判定する（ステップＳ１６０５）。ユーザがメロディ１１０の再生を指示した場合（ステップＳ１６０５の判定がＹＥＳの場合）、ＣＰＵ１４０１は、再生処理（ステップＳ１６０９）を実行する。この処理は、図１の出力部１０７内の楽譜表示部１０７－１および楽音再生部１０７－２の動作として前述した通りである。

【００８２】

ユーザが自動作曲を指示していない場合（ステップＳ１６０４の判定がＮＯの場合）、ＣＰＵ１４０１は、ステップＳ１６０２の処理に戻る。

【００８３】

図１７は、図１６のステップＳ１６０７のコード進行選択処理の詳細例を示すフローチャートである。

【００８４】

まず、ＣＰＵ１４０１は、ＲＡＭ１４０３上の変数データおよび配列変数データを初期化する（ステップＳ１７０１）。

【００８５】

次に、ＣＰＵ１４０１は、伴奏・コード進行ＤＢ１０３に記憶されている複数のコード進行データに対する繰返し処理を制御するためのＲＡＭ１４０３上の変数ｎを「０」に初期化する。その後、ＣＰＵ１４０１は、ステップＳ１７１４で変数ｎの値を＋１ずつインクリメントさせながら、ステップＳ１７０３で変数ｎの値がＲＯＭ１４０２に記憶されている定数データＭＡＸ＿ＣＨＯＲＤ＿ＰＲＯＧの値よりも小さいと判定される間、ステップＳ１７０４からＳ１７１３までの一連の処理を実行する。定数データＭＡＸ＿ＣＨＯＲＤ＿ＰＲＯＧの値は、伴奏・コード進行ＤＢ１０３に記憶されるコード進行データの数を示す定数データである。ＣＰＵ１４０１は、図５に示される伴奏・コード進行ＤＢ１０３

のレコード数の分だけ、ステップＳ１７０４からＳ１７１３までの一連の処理を繰り返し実行することにより、適合度の算出処理を、伴奏・コード進行ＤＢ１０３に記憶されている複数のコード進行データに対して実行し、入力モチーフ１０８との適合度が高かった例えば上位３個のコード進行データをそれぞれ指し示す＃０、＃１、＃２のコード進行候補指示データ１０９を出力する。

【００８６】

ステップＳ１７０３からＳ１７１３の繰返し処理において、ステップＳＣＰＵ１４０１はまず、変数ｎの値が定数データＭＡＸ＿ＣＨＯＲＤ＿ＰＲＯＧの値よりも小さいか否かを判定する（ステップＳ１７０３）。

【００８７】

ステップＳ１７０３の判定がＹＥＳならば、ＣＰＵ１４０１は、変数データｎが示すｎ番目のコード進行データ＃ｎ（図５（ａ）参照）を、伴奏・コード進行ＤＢ１０３からＲＡＭ１４０３内のコード進行データ領域に読み込む（ステップＳ１７０４）。このコード進行データ＃ｎのデータ形式は、例えば図５の（ｂ）、（ｃ）、（ｄ）で示されるフォーマットを有する。

【００８８】

次に、ＣＰＵ１４０１は、伴奏・コード進行ＤＢ１０３からＲＡＭ１４０３内のコード進行データ＃ｎ用の配列変数データ要素ｉＣｈｏｒｄＡｔｔｒｉｂｕｔｅ［ｎ］［０］に読み込まれた、コード進行データ＃ｎの楽曲ジャンルを示す値が、予め特には図示しないスイッチによりユーザによって設定され、ＲＡＭ１４０３内のる変数データｉＪｕｎｌｅＳｅｌｅｃｔに記憶されている楽曲ジャンルを示す値と等しいか否かを判定する（ステップＳ１７０５）。ステップＳ１７０５の判定がＮＯならば、そのコード進行データ＃ｎは、ユーザが望む楽曲ジャンルに合わないため、選択せずに、ステップＳ１７１４に進む。

【００８９】

ステップＳ１７０５の判定がＹＥＳならば、ＣＰＵ１４０１は、伴奏・コード進行ＤＢ１０３からＲＡＭ１４０３内のコード進行データ＃ｎ用の配列変数データ要素ｉＣｈｏｒｄＡｔｔｒｉｂｕｔｅ［ｎ］［１］に読み込まれた、コード進行データ＃ｎのコンセプトを示す値が、予め特には図示しないスイッチによりユーザによって設定され、ＲＡＭ１４０３内のる変数データｉＣｏｎｎｃｅｐｔＳｅｌｅｃｔに記憶されている楽曲のコンセプトを示す値と等しいか否かを判定する（ステップＳ１７０６）。ステップＳ１７０６の判定がＮＯならば、そのコード進行データ＃ｎは、ユーザが望む楽曲コンセプトに合わないため、選択せずに、ステップＳ１７１４に進む。

【００９０】

ステップＳ１７０６の判定がＹＥＳならば、ＣＰＵ１４０１は、コードデザインデータ作成処理を実行する（ステップＳ１７０７）。この処理において、ＣＰＵ１４０１は、コード進行データ＃ｎによって、時間経過に沿って順次指定されるコード進行の情報を、ＲＡＭ１４０３上の配列変数データである後述するコードデザインデータｃｄｅｓｉｇｎ［ｋ］に格納する処理を実行する。

【００９１】

次に、ＣＰＵ１４０１は、ＲＡＭ１４０３上の変数データｉＫｅｙＳｈｉｆｔに初期値「０」を格納する（ステップＳ１７０８）。この変数データｉＫｅｙＳｈｉｆｔは、１オクターブの半音階中で、初期値「０」からＲＯＭ１４０２に記憶されている定数データＰＩＴＣＨ＿ＣＬＡＳＳ＿Ｎより１小さい数までの範囲で、コード進行データ＃ｎに対する半音単位のキーシフト値を指定する。定数データＰＩＴＣＨ＿ＣＬＡＳＳ＿Ｎの値は、通常は１オクターブ内の半音数１２である。

【００９２】

次に、ＣＰＵ１４０１は、変数データｉＫｅｙＳｈｉｆｔの値が定数データＰＩＴＣＨ＿ＣＬＡＳＳ＿Ｎの値よりも小さいか否かを判定する（ステップＳ１７０９）。

【００９３】

ステップＳ１７０９の判定がＹＥＳならば、変数データｉＫｅｙＳｈｉｆｔが示すキー

シフト値だけコード進行データ＃ｎのキーをシフトさせた後、入力モチーフ１０８とコード進行＃ｎに対する適合度チェック処理を実行する（ステップＳ１７１０）。この処理により、入力モチーフ１０８に対するコード進行＃ｎの適合度がＲＡＭ１４０３上の変数データｄｏＶａｌｕｅに得られる。

【００９４】

次に、ＣＰＵ１４０１は、変数データｄｏＶａｌｕｅの値が、ＲＡＭ１４０３上の変数データｄｏＭａｘＶａｌｕｅよりも大きいか否かを判定する（ステップＳ１７１１）。変数データｄｏＭａｘＶａｌｕｅは、現時点で最も高い適合度の値を格納する変数で、ステップＳ１７０１で値「０」に初期化されている。

【００９５】

ステップＳ１７１１の判定がＹＥＳならば、ＣＰＵ１４０１は、変数データｄｏＭａｘＶａｌｕｅの値を変数データｄｏＶａｌｕｅの値で置き換える。また、ＣＰＵ１４０１は、ＲＡＭ１４０３内の配列変数データｉＢｅｓｔＫｅｙＳｈｉｆｔ［ｉＢｅｓｔＵｐｄａｔｅ］に、変数データｉＫｅｙＳｈｉｆｔの現在値を格納する。また、ＣＰＵ１４０１は、ＲＡＭ１４０３内の配列変数データｉＢｅｓｔＣｈｏｒｄＰｒｏｇ［ｉＢｅｓｔＵｐｄａｔｅ］に、伴奏・コード進行ＤＢ１０３上のコード進行データを指し示す変数データｎの現在値を格納する。その後、ＣＰＵ１４０１は、ＲＡＭ１４０３内の変数データｉＢｅｓｔＵｐｄａｔｅを＋１インクリメントする（以上、ステップＳ１７１２）。変数データｉＢｅｓｔＵｐｄａｔｅは、ステップＳ１７０１で値「０」に初期化された後、現時点で適合度が最も高いコード進行データが見つかるごとにインクリメントされるデータであり、その値が大きいほど上位の適合度であることを示す。配列変数データｉＢｅｓｔＫｅｙＳｈｉｆｔ［ｉＢｅｓｔＵｐｄａｔｅ］は、変数データｉＢｅｓｔＵｐｄａｔｅが示す順位におけるキーシフト値を保持する。配列変数データｉＢｅｓｔＣｈｏｒｄＰｒｏｇ［ｉＢｅｓｔＵｐｄａｔｅ］は、変数データｉＢｅｓｔＵｐｄａｔｅが示す順位における伴奏・コード進行ＤＢ１０３上のコード進行の番号を保持する。

【００９６】

ステップＳ１７１１の判定がＮＯならば、ＣＰＵ１４０１は、上記ステップＳ１７１２の処理はスキップして、今回のコード進行データ＃ｎは入力モチーフ１０８に対する自動作曲用のコード進行データとしては選択しない。

【００９７】

その後、ＣＰＵ１４０１は、変数データｉＫｅｙＳｈｉｆｔの値を＋１インクリメントする（ステップＳ１７１３）。その後、ＣＰＵ１４０１は、ステップＳ１７０９の処理に戻る。

【００９８】

ＣＰＵ１４０１は、変数データｉＫｅｙＳｈｉｆｔの値をインクリメントしながらステップＳ１７０９からＳ１７１３までの処理を繰り返し実行した後、１オクターブ分のキーシフト値の指定が終了してステップＳ１７０９の判定がＮＯになると、ステップＳ１７１４に処理を進める。ステップＳ１７１４で、ＣＰＵ１４０１は、伴奏・コード進行ＤＢ１０３上のコード進行データの選択用の変数データｎを＋１インクリメントする。その後、ＣＰＵ１４０１は、ステップＳ１７０３の処理に戻る。

【００９９】

ＣＰＵ１４０１は、変数データｎの値をインクリメントしながらステップＳ１７０３からＳ１７１４までの一連の処理を繰り返し実行した後、伴奏・コード進行ＤＢ１０３内の全てのコード進行データに対する処理を終了してステップＳ１７０３の判定がＮＯになると、図１７のフローチャートの処理すなわち図１６のステップＳ１６０７のコード進行選択処理を終了する。この結果、変数データｉＢｅｓｔＵｐｄａｔｅの現在値よりも１だけ小さい値「ｉＢｅｓｔＵｐｄａｔｅ－１」を要素番号とする配列変数データｉＢｅｓｔＫｅｙＳｈｉｆｔ［ｉＢｅｓｔＵｐｄａｔｅ－１］およびｉＢｅｓｔＣｈｏｒｄＰｒｏｇ［ｉＢｅｓｔＵｐｄａｔｅ－１］に、入力モチーフ１０８に対して最も適合性が高いキーシフト値とコード進行データの番号が格納される。また、配列変数データｉＢｅｓｔＫｅｙ

Ｓｈｉｆｔ［ｉＢｅｓｔＵｐｄａｔｅ－２］およびｉＢｅｓｔＣｈｏｒｄＰｒｏｇ［ｉＢｅｓｔＵｐｄａｔｅ－２］に、入力モチーフ１０８に対して２番目に適合性が高いキーシフト値とコード進行データの番号が格納される。さらに、配列変数データｉＢｅｓｔＫｅｙＳｈｉｆｔ［ｉＢｅｓｔＵｐｄａｔｅ－３］およびｉＢｅｓｔＣｈｏｒｄＰｒｏｇ［ｉＢｅｓｔＵｐｄａｔｅ－３］に、入力モチーフ１０８に対して３番目に適合性が高いキーシフト値とコード進行データの番号が格納される。これらのデータセットが、上位から順に図１の＃０、＃１、および＃２のコード進行候補指示データ１０９に対応する。

【０１００】

図１８は、図１７のステップＳ１７０７のコードデザインデータ作成処理の詳細例を示すフローチャートである。

【０１０１】

まず、ＣＰＵ１４０１は、コード進行情報の番号を示す変数データｉＣＤｅｓｉｇｎＣｎｔを初期値「０」に設定する（ステップＳ１８０１）。

【０１０２】

次に、ＣＰＵ１４０１は、図１７のステップＳ１７０４で伴奏・コード進行ＤＢ１０３からＲＡＭ１４０３に例えば図５（ｂ）、（ｃ）、（ｄ）のデータ形式で読み込まれたコード進行データ＃ｎの最初のメタイベント（図５（ｂ）のコードデータ＃０に対応）へのポインタを、ＲＡＭ１４０３内のポインタ変数データｍｔに格納する（ステップＳ１８０２）。

【０１０３】

次に、ＣＰＵ１４０１は、ステップＳ１８１１でポインタ変数データｍｔに順次次のメタイベント（図５（ｂ）のコードデータ＃１、＃２、・・・）へのポインタを格納しながら、ステップＳ１８０３で終端（図５（ｂ）の「終端」）に達したと判定するまで、ステップＳ１８０３からＳ１８１１の一連の処理を、コード進行データ＃ｎの各コードデータ（図５（ｂ）参照）に対して、繰り返し実行する。

【０１０４】

上記繰返し処理において、ＣＰＵ１４０１はまず、ポインタ変数データｍｔが終端を指しているか否かを判定する（ステップＳ１８０３）。

【０１０５】

ステップＳ１８０３の判定がＮＯならば、ＣＰＵ１４０１は、ポインタ変数データｍｔが指すコードデータ（図５（ｂ））中のコード根音（ルート）とコードタイプ（図５（ｄ）参照）を抽出して、ＲＡＭ１４０３内の変数データｒｏｏｔとｔｙｐｅに格納することを試みる（ステップＳ１８０４）。そして、ＣＰＵ１４０１は、ステップＳ１８０４での格納処理に成功したか否かを判定する（ステップＳ１８０５）。

【０１０６】

ステップＳ１８０４での格納処理に成功した場合（ステップＳ１８０５の判定がＹＥＳの場合）、ＣＰＵ１４０１は、ポインタ変数データｍｔが指す記憶エリアの時間情報ｍｔ－＞ｉＴｉｍｅ（図５（ｄ）の「時間」データ）を、変数データｉＣＤｅｓｉｇｎＣｎｔの現在値を要素番号とするコードデザインデータの時間項目ｃｄｅｓｉｇｎ［ｉＣＤｅｓｉｇｎＣｎｔ］－＞ｉＴｉｍｅに格納する。また、ＣＰＵ１４０１は、ステップＳ１８０４で変数データｒｏｏｔに格納されたコード根音情報を、変数データｉＣＤｅｓｉｇｎＣｎｔの現在値を要素番号とするコードデザインデータのコード根音項目ｃｄｅｓｉｇｎ［ｉＣＤｅｓｉｇｎＣｎｔ］－＞ｉＲｏｏｔに格納する。また、ＣＰＵ１４０１は、ステップＳ１８０４で変数データｔｙｐｅに格納されたコードタイプ情報を、変数データｉＣＤｅｓｉｇｎＣｎｔの現在値を要素番号とするコードデザインデータのコードタイプ項目ｃｄｅｓｉｇｎ［ｉＣＤｅｓｉｇｎＣｎｔ］－＞ｉＴｙｐｅに格納する。さらに、変数データｉＣＤｅｓｉｇｎＣｎｔの現在値を要素番号とするコードデザインデータのキー項目ｃｄｅｓｉｇｎ［ｉＣＤｅｓｉｇｎＣｎｔ］－＞ｉＫｅｙとスケール項目ｃｄｅｓｉｇｎ［ｉＣＤｅｓｉｇｎＣｎｔ］－＞ｉＳｃａｌｅには、無効値「－１」を格納する（以上、ステップＳ１８０６）。その後、ＣＰＵ１４０１は、ステップＳ１８１０の処理に移行して

、変数データｉＣＤｅｓｉｇｎＣｎｔの値を＋１インクリメントする。
【０１０７】
ステップＳ１８０４での格納処理に成功しなかった場合（ステップＳ１８０５の判定がＮＯの場合）、ＣＰＵ１４０１は、ポインタ変数データｍｔが指すコードデータ（図５（ｂ））中のスケールとキー（図５（ｃ）参照）を抽出して、ＲＡＭ１４０３内の変数データｓｃａｌｅとｋｅｙに格納することを試みる（ステップＳ１８０７）。そして、ＣＰＵ１４０１は、ステップＳ１８０７での格納処理に成功したか否かを判定する（ステップＳ１８０８）。
【０１０８】
ステップＳ１８０７での格納処理に成功した場合（ステップＳ１８０８の判定がＹＥＳの場合）、ＣＰＵ１４０１は、ポインタ変数データｍｔが指す記憶エリアの時間情報ｍｔ－＞ｉＴｉｍｅ（図５（ｃ）の「時間」データ）を、変数データｉＣＤｅｓｉｇｎＣｎｔの現在値を要素番号とするコードデザインデータの時間項目ｃｄｅｓｉｇｎ［ｉＣＤｅｓｉｇｎＣｎｔ］－＞ｉＴｉｍｅに格納する。また、ＣＰＵ１４０１は、ステップＳ１８０７で変数データｋｅｙに格納されたキー情報を、変数データｉＣＤｅｓｉｇｎＣｎｔの現在値を要素番号とするコードデザインデータのキー項目ｃｄｅｓｉｇｎ［ｉＣＤｅｓｉｇｎＣｎｔ］－＞ｉＫｅｙに格納する。また、ＣＰＵ１４０１は、ステップＳ１８０７で変数データｓｃａｌｅに格納されたスケール情報を、変数データｉＣＤｅｓｉｇｎＣｎｔの現在値を要素番号とするコードデザインデータのスケール項目ｃｄｅｓｉｇｎ［ｉＣＤｅｓｉｇｎＣｎｔ］－＞ｉＳｃａｌｅに格納する。さらに、変数データｉＣＤｅｓｉｇｎＣｎｔの現在値を要素番号とするコードデザインデータのコード根音項目ｃｄｅｓｉｇｎ［ｉＣＤｅｓｉｇｎＣｎｔ］－＞ｉＲｏｏｔとコードタイプ項目ｃｄｅｓｉｇｎ［ｉＣＤｅｓｉｇｎＣｎｔ］－＞ｉＴｙｐｅには、無効値「－１」を格納する（以上、ステップＳ１８０９）。その後、ＣＰＵ１４０１は、ステップＳ１８１０の処理に移行して、変数データｉＣＤｅｓｉｇｎＣｎｔの値を＋１インクリメントする。
【０１０９】
ＣＰＵ１４０１は、ステップＳ１８１０での変数データｉＣＤｅｓｉｇｎＣｎｔの値のインクリメント処理の後、またはステップＳ１８０７での格納処理に成功しなかった場合（ステップＳ１８０８の判定がＮＯの場合）、ポインタ変数データｍｔに次のメタイベント（図５（ｂ）のコードデータ＃１、＃２、・・・）へのポインタを格納し（ステップＳ１８１１）、ステップＳ１８０３の判定処理に戻る。
【０１１０】
上記ステップＳ１８０３からＳ１８１１までの繰返し処理の結果、ＣＰＵ１４０１は、現在のコード進行データ＃ｎに対するコードデータを終端（図５（ｂ）参照）まで読み込むと、ステップＳ１８０３の判定がＹＥＳとなって、図１８のフローチャートで例示される処理すなわち図１７のステップＳ１７０７のコードデザインデータ作成処理を終了する。この時点で、変数データｉＣＤｅｓｉｇｎＣｎｔに現在のコード進行データ＃ｎを構成するコード情報の数が得られ、コードデザインデータｃｄｅｓｉｇｎ［０］からｃｄｅｓｉｇｎ［ｉＣＤｅｓｉｇｎＣｎｔ－１］にそれぞれのコード情報が得られる。
【０１１１】
図１９は、図１７のステップＳ１７１０の入力モチーフ１０８とコード進行＃ｎに対する適合度チェック処理の詳細例を示すフローチャートである。
【０１１２】
まず、ＣＰＵ１４０１は、適合度を示す変数データｄｏＶａｌｕｅに初期値「０」をセットする（ステップＳ１９０１）。
【０１１３】
次に、ＣＰＵ１４０１は、伴奏・コード進行ＤＢ１０３から、ステップＳ１７０４で読み込んだコード進行データ＃ｎに対応する曲構造データ＃ｎ（図５（ａ）参照）を参照し、入力モチーフ１０８の入力時にユーザにより指定されたフレーズ種別と同じフレーズ種別が「ＰａｒｔＮａｍｅ［Ｍ］」項目（図６参照）に指定されている先頭の小節のレコー

ドに格納されている小節開始時間データｉＰａｒｔＴｉｍｅ［Ｍ］を読み込み、ＲＡＭ１４０３内の変数データｓＴｉｍｅに格納する（ステップＳ１９０２）。

【０１１４】

次に、ＣＰＵ１４０１は、入力モチーフ１０８を構成するノートの順番を指す変数データｉＮｏｔｅＣｎｔの値を初期値「０」に設定する（ステップＳ１９０３）。

【０１１５】

次に、ＣＰＵ１４０１は、図１６のステップＳ１６０６でＲＡＭ１４０３に図４のデータ形式で入力された入力モチーフ１０８の最初のノートデータ（図４（ａ）のノートデータ＃０に対応）へのポインタを、ＲＡＭ１４０３内のポインタ変数データｍｅに格納する（ステップＳ１９０４）。

【０１１６】

次に、ＣＰＵ１４０１は、ステップＳ１９０９でポインタ変数データｍｅに順次入力モチーフ１０８の次のノート（図４（ａ）のノートデータ＃１、＃２、・・・）へのポインタを格納しながら、ステップＳ１９０５で終端（図４（ｂ）の「終端」）に達したと判定するまで、ステップＳ１９０５からＳ１９０９の一連の処理を、入力モチーフ１０８の各ノートデータ（図４（ａ）参照）に対して、繰り返し実行する。

【０１１７】

上記繰返し処理において、ＣＰＵ１４０１はまず、ポインタ変数データｍｅが終端を指しているか否かを判定する（ステップＳ１９０５）。

【０１１８】

ステップＳ１９０５の判定がＮＯならば、ＣＰＵ１４０１は、ポインタ変数データｍｅが指すノートデータ（図４（ｂ））中の「時間」データであるｍｅ－＞ｉＴｉｍｅを参照し、これにステップＳ１９０２で得ている入力モチーフ１０８の該当小節の小節開始時間ｓＴｉｍｅを加算し、その結果を新たにｍｅ－＞ｉＴｉｍｅに上書きする（ステップＳ１９０６）。入力モチーフ１０８を構成する各ノートデータ中の「時間」データは、２小節からなる入力モチーフ１０８の先頭からの時間であるため、それを楽曲の先頭からの時間に変換するために、ステップＳ１９０２で曲構造データから得た入力モチーフ１０８の該当小節の小節開始時間ｓＴｉｍｅが加算される。

【０１１９】

次に、ＣＰＵ１４０１は、ポインタ変数データｍｅの値を、変数データｉＮｏｔｅＣｎｔの現在値を要素値とする配列変数データであるノートポインタ配列データｎｏｔｅ［ｉＮｏｔｅＣｎｔ］に格納する（ステップＳ１９０７）。

【０１２０】

その後、ＣＰＵ１４０１は、変数データｉＮｏｔｅＣｎｔの値を＋１インクリメントする（ステップＳ１９０８）。そして、ＣＰＵ１４０１は、ポインタ変数データｍｅに入力モチーフ１０８中の次のノートデータ（図４（ａ）のノートデータ＃１、＃２、・・・）へのポインタを格納し（ステップＳ１９０９）、ステップＳ１９０５の判定処理に戻る。

【０１２１】

上記ステップＳ１９０５からＳ１９０９までの繰返し処理の結果、ＣＰＵ１４０１は、入力モチーフ１０８中のノートデータを終端（図４（ａ）参照）まで読み込むと、ステップＳ１９０５の判定がＹＥＳとなって、ステップＳ１９１０のチェック処理に進む。このチェック処理では、入力モチーフ１０８に対するコード進行＃ｎの適合度を算出処理が実行され、この結果、適合度が変数データｄｏＶａｌｕｅに得られる。その後、図１９のフローチャートで例示される処理すなわち図１７のステップＳ１７１０の入力モチーフ１０８とコード進行＃ｎの適合度チェック処理を終了する。この時点で、変数データｉＮｏｔｅＣｎｔに入力モチーフ１０８を構成するノートの数（図３（ａ）の音符の数に対応）が得られ、ノートポインタ配列変数データｎｏｔｅ［０］～ｎｏｔｅ［ｉＮｏｔｅＣｎｔ－１］にそれぞれのノートデータへのポインタが得られる。

【０１２２】

図２０は、図１９のステップＳ１９１０のチェック処理の詳細例を示すフローチャート

である。

【０１２３】

まず、ＣＰＵ１４０１は、入力モチーフ１０８のノート数をカウントするＲＡＭ１４０３内の変数ｉに初期値「０」を格納する（ステップＳ２００１）。その後、ＣＰＵ１４０１は、ステップＳ２００８で変数ｉの値を＋１ずつインクリメントさせながら、ステップＳ２００２で変数ｉの値が図１９の処理で最終的に得られた入力モチーフ１０８のノート数を示す変数データｉＮｏｔｅＣｎｔの値よりも小さいと判定される間、ステップＳ２００２からＳ２００８までの一連の処理を実行する。

【０１２４】

ステップＳ２００２からＳ２００８の繰返し処理において、ステップＳＣＰＵ１４０１はまず、変数ｉの値が変数データｉＮｏｔｅＣｎｔの値よりも小さいか否かを判定する（ステップＳ２００２）。

【０１２５】

ステップＳ２００２の判定がＹＥＳならば、ＣＰＵ１４０１は、変数データｉによって指示されるｉ番目の処理対象ノートに対応するノートポインタ配列変数データｎｏｔｅ［ｉ］から、ピッチ項目値ｎｏｔｅ［ｉ］－＞ｉＰｉｔ（図４（ｂ）の「ピッチ」項目値を指す）を読み出し、それを変数データｉの値を要素値とするＲＡＭ１４０３内のピッチ情報列配列変数データｉｐｉｔ［ｉ］に格納する（ステップＳ２００３）。

【０１２６】

次に、ＣＰＵ１４０１は、入力モチーフ１０８の現在の処理対象ノートのタイミングに対応するコード情報の取得処理を実行する（ステップＳ２００４）。この処理では、入力モチーフ１０８の現在の処理対象ノートの発音タイミングにおいて指定されるべきコードのコード根音、コードタイプ、スケール、およびキーが、変数データｒｏｏｔ、ｔｙｐｅ、ｓｃａｌｅ、およびｋｅｙに得られる。

【０１２７】

続いて、ＣＰＵ１４０１は、ノートタイプの取得処理を実行する（ステップＳ２００５）。この処理では、図８の説明で前述した、ＲＡＭ１４０３内のノートタイプと隣接音程の配列変数データｉｎｃｏｎ［ｉ×２］（偶数番目の要素）に、入力モチーフ１０８の現在のｉ番目の処理対象ノートの、ピッチｉｐｉｔ［ｉ］の現在の評価対象のコード進行データ＃ｎに対するノートタイプが得られる。

【０１２８】

さらに、ＣＰＵ１４０１は、変数ｉの値が０よりも大きいか否か、すなわち処理対象ノートが先頭以外のノートであるか否かを判定する（ステップＳ２００６）。

【０１２９】

そして、ステップＳ２００６の判定がＹＥＳのときに、ＣＰＵ１４０１は、変数データｉによって指示されるｉ番目の処理対象ノートに対応するピッチ情報ｉｐｉｔ［ｉ］から、ｉ－１番目の処理対象ノートに対応するピッチ情報ｉｐｉｔ［ｉ－１］を減算することにより、ノートタイプと隣接音程の配列変数データｉｎｃｏｎ［ｉ×２－１］（奇数番目の要素）に、図８の説明で前述した隣接音程を得る（ステップＳ２００７）。

【０１３０】

ステップＳ２００６の判定がＮＯのとき（先頭のノートのとき）には、ＣＰＵ１４０１は、ステップＳ２００７の処理はスキップする。

【０１３１】

その後、ＣＰＵ１４０１は、変数ｉの値を＋１インクリメントし（ステップＳ２００８）、入力モチーフ１０８中の次のノートの処理に移行して、ステップＳ２００２の判定処理に戻る。

【０１３２】

ＣＰＵ１４０１は、変数データｉの値をインクリメントしながらステップＳ２００２からＳ２００８までの一連の処理を繰り返し実行した後、入力モチーフ１０８を構成する全てのノートデータに対する処理を終了してステップＳ２００２の判定がＮＯになると、ス

テップＳ２００９のノート接続性チェック処理に進む。この時点で、配列変数データｉｎｃｏｎ［ｉ×２］（０≦ｉ≦ｉＮｏｔｅＣｎｔ－１）およびｉｎｃｏｎ［ｉ×２－１］（１≦ｉ≦ｉＮｏｔｅＣｎｔ－１）に、図８の説明等で前述したノートタイプと隣接音程の集合が得られる。そして、ＣＰＵ１４０１は、このデータに基づいて、ステップＳ２００９のノート接続性チェック処理により、評価対象のコード進行データ＃ｎの入力モチーフ１０８に対する適合度を変数データｄｏＶａｌｕｅに得る。その後、ＣＰＵ１４０１は、図２０のフローチャートで例示される処理すなわち図１９のステップＳ１９１０のチェック処理を終了する。

【０１３３】

図２１は、図２０のステップＳ２００４の入力モチーフ１０８の現在のノートのタイミングnに対応するコード情報の取得処理の詳細例を示すフローチャートである。

【０１３４】

まず、ＣＰＵ１４０１は、コードデザインデータの情報数をカウントするＲＡＭ１４０３内の変数ｋに初期値「０」を格納する（ステップＳ２１０１）。その後、ＣＰＵ１４０１は、ステップＳ２１０７で変数ｋの値を＋１ずつインクリメントさせながら、ステップＳ２１０２で変数ｋの値が図１８の処理で最終的に得られた現在の評価対象のコード進行データ＃ｎを構成するコード情報の数を示す変数データｉＣＤｅｓｉｇｎＣｎｔの値よりも小さいと判定される間、ステップＳ２１０２からＳ２１０７までの一連の処理を実行する。

【０１３５】

ステップＳ２１０２からＳ２１０７の繰返し処理において、ステップＳＣＰＵ１４０１はまず、変数ｋの値が変数データｉＣＤｅｓｉｇｎＣｎｔの値よりも小さいか否かを判定する（ステップＳ２１０２）。

【０１３６】

ステップＳ２１０２の判定がＹＥＳならば、ＣＰＵ１４０１は、現在の処理対象のノートのノートポインタ配列データが指す時間項目値ｎｏｔｅ［ｉ］－＞ｉＴｉｍｅが、変数データｋによって指示されるｋ番目のコードデザインデータの時間項目ｃｄｅｓｉｇｎ［ｋ］－＞ｉＴｉｍｅの値よりも大きく、ｋ＋１番目のコードデザインデータの時間項目ｃｄｅｓｉｇｎ［ｋ＋１］－＞ｉＴｉｍｅの値よりも小さく、かつ、ｋ番目のコードデザインデータのキー項目ｃｄｅｓｉｇｎ［ｋ］－＞ｉＫｅｙとスケール項目ｃｄｅｓｉｇｎ［ｋ］－＞ｉＳｃａｌｅの各値が０以上で有意な値が設定されているか否か（図１８のステップＳ１８０６、Ｓ１８０８参照）を判定する（ステップＳ２１０３）。

【０１３７】

ステップＳ２１０３の判定がＹＥＳならば、入力モチーフ１０８の現在の処理対象のノートｎｏｔｅ［ｉ］の発音タイミングにおいてｋ番目のコードデザインデータｃｄｅｓｉｇｎ［ｋ］によるコード情報が指定されていると判定できる。そこで、ＣＰＵ１４０１は、変数データｋｅｙとｓｃａｌｅに、それぞれｋ番目のコードデザインデータのキー項目ｃｄｅｓｉｇｎ［ｋ］－＞ｉＫｅｙとスケール項目ｃｄｅｓｉｇｎ［ｋ］－＞ｉＳｃａｌｅの各値を格納する（ステップＳ２１０４）。

【０１３８】

ステップＳ２１０３の判定がＮＯならば、ＣＰＵ１４０１は、ステップＳ２１０４の処理はスキップする。

【０１３９】

続いて、ＣＰＵ１４０１は、現在の処理対象のノートのノートポインタ配列データが指す時間項目値ｎｏｔｅ［ｉ］－＞ｉＴｉｍｅが、変数データｋによって指示されるｋ番目のコードデザインデータの時間項目ｃｄｅｓｉｇｎ［ｋ］－＞ｉＴｉｍｅの値よりも大きく、ｋ＋１番目のコードデザインデータの時間項目ｃｄｅｓｉｇｎ［ｋ＋１］－＞ｉＴｉｍｅの値よりも小さく、かつ、ｋ番目のコードデザインデータのコード根音項目ｃｄｅｓｉｇｎ［ｋ］－＞ｉＲｏｏｔとコードタイプ項目ｃｄｅｓｉｇｎ［ｋ］－＞ｉＴｙｐｅの各値が０以上で有意な値が設定されているか否か（図１８のステップＳ１８０６、Ｓ１８

０８参照）を判定する（ステップＳ２１０５）。
【０１４０】
ステップＳ２１０５の判定がＹＥＳならば、入力モチーフ１０８の現在の処理対象のノートｎｏｔｅ［ｉ］の発音タイミングにおいてｋ番目のコードデザインデータｃｄｅｓｉｇｎ［ｋ］によるコード情報が指定されていると判定できる。そこで、ＣＰＵ１４０１は、変数データｒｏｏｔとｔｙｐｅに、それぞれｋ番目のコードデザインデータのコード根音項目ｃｄｅｓｉｇｎ［ｋ］－＞ｉＲｏｏｔとコードタイプ項目ｃｄｅｓｉｇｎ［ｋ］－＞ｉＴｙｐｅの各値を格納する（ステップＳ２１０６）。
【０１４１】
ステップＳ２１０５の判定がＮＯならば、ＣＰＵ１４０１は、ステップＳ２１０６の処理はスキップする。
【０１４２】
以上の処理の後、ＣＰＵ１４０１は、ＣＰＵ１４０１は、変数ｋの値を＋１インクリメントし（ステップＳ２１０７）、次のコードデザインデータｃｄｅｓｉｇｎ［ｋ］の処理に移行して、ステップＳ２１０２の判定処理に戻る。
【０１４３】
ＣＰＵ１４０１は、変数データｋの値をインクリメントしながらステップＳ２１０２からＳ２１０７までの一連の処理を繰り返し実行した後、全てのコードデザインデータに対する処理を終了してステップＳ２１０２の判定がＮＯになると、図２１のフローチャートで例示される処理すなわち図２０のステップＳ２００４の処理を終了する。この結果、変数データｒｏｏｔとｔｙｐｅおよび変数データｓｃａｌｅとｋｅｙに、入力モチーフ１０８の現在の処理対象ノートの発音タイミングに対応するコード情報が得られる。
【０１４４】
図２２は、図２０のステップＳ２００５のノートタイプ取得処理の詳細例を示すフローチャートである。この処理は、図７を用いて前述したように、図２０のステップＳ２００３で設定されている入力モチーフ１０８の現在のノートｎｏｔｅｓ［ｉ］に対応するピッチｉｐｉｔ［ｉ］と、図２０のステップＳ２００４で算出されている入力モチーフ１０８の現在のノートｎｏｔｅｓ［ｉ］の発音タイミングに対応するコード進行を構成するキーｋｅｙ、スケールｓｃａｌｅ、コード根音ｒｏｏｔ、およびコードタイプｔｙｐｅとに従って、入力モチーフ１０８の現在のノートｎｏｔｅｓ［ｉ］のノートタイプを取得する処理である。
【０１４５】
まず、ＣＰＵ１４０１は、ＲＯＭ１４０２に記憶されている標準ピッチクラスセットテーブル中の図７（ａ）に例示されるデータ構成を有するコードトーンテーブルから、図２０のステップＳ２００４で算出されたコードタイプｔｙｐｅに対応するコードトーンピッチクラスセットを取得し、ＲＡＭ１４０３上の変数データｐｃｓ１に格納する（ステップＳ２２０１）。以下、この変数データｐｃｓ１の値をコードトーンピッチクラスセットｐｃｓ１と呼ぶ。
【０１４６】
次に、ＣＰＵ１４０１は、ＲＯＭ１４０２に記憶されている標準ピッチクラスセットテーブル中の図７（ｂ）に例示されるデータ構成を有するテンションノートテーブルから、上記コードタイプｔｙｐｅに対応するテンションノートピッチクラスセットを取得し、ＲＡＭ１４０３上の変数データｐｃｓ２に格納する（ステップＳ２２０２）。以下、この変数データｐｃｓ２の値をテンションノートピッチクラスセットｐｃｓ２と呼ぶ。
【０１４７】
次に、ＣＰＵ１４０１は、ＲＯＭ１４０２に記憶されている標準ピッチクラスセットテーブル中の図７（ｃ）に例示されるデータ構成を有するスケールノートテーブルから、図２０のステップＳ２００４で得られているスケールｓｃａｌｅに対応するスケールノートピッチクラスセットを取得し、ＲＡＭ１４０３上の変数データｃｓ３に格納する（ステップＳ２２０３）。以下、この変数データｐｃｓ３の値をスケールノートピッチクラスセッ

トｐｃｓ３と呼ぶ。
【０１４８】
続いて、ＣＰＵ１４０１は、入力モチーフ１０８の現在の処理対象のノートｎｏｔｅｓ［ｉ］に対して図２０のステップＳ２００３で得られているピッチｉｐｉｔ［ｉ］を、コード根音ｒｏｏｔを第０音の音階構成音としたときの第０音から第１１音までの１オクターブ分の音階構成音のいずれかに写像させたときの、ピッチｉｐｉｔ［ｉ］のコード根音ｒｏｏｔに対する音程を、次式により算出し、ＲＡＭ１４０３上の変数データｐｃ１に格納する（ステップＳ２２０４）。以下、変数データｐｃ１の値を、入力モチーフピッチクラスｐｃ１と呼ぶ。
【０１４９】
ｐｃ１＝（ｉｐｉｔ［ｉ］－ｒｏｏｔ＋１２）ｍｏｄ１２　・・・（１）
【０１５０】
なお、「ｍｏｄ１２」は、その左側の括弧に対応する「値」を１２で割ったときの余りである。
【０１５１】
同様に、ＣＰＵ１４０１は、入力モチーフ１０８の現在のノートｎｏｔｅｓ［ｉ］に対して図２０のステップＳ２００４で得られているピッチｉｐｉｔ［ｉ］を、キーｋｅｙを第０音の音階構成音としたときの第０音から第１１音までの１オクターブ分の音階構成音のいずれかに写像させたときの、ピッチｉｐｉｔ［ｉ］のキーｋｅｙに対する音程を、次式により算出し、ＲＡＭ１４０３上の変数データｐｃ２に格納する（ステップＳ２２０５）。以下、変数データｐｃ２の値を、入力モチーフピッチクラスｐｃ２と呼ぶ。
【０１５２】
ｐｃ２＝（ｉｐｉｔ［ｉ］－ｋｅｙ＋１２）ｍｏｄ１２　・・・（２）
【０１５３】
次に、ＣＰＵ１４０１は、入力モチーフピッチクラスｐｃ１がコードトーンピッチクラスセットｐｃｓ１に含まれるか否かを判定する（ステップＳ２２０６）。この判定演算処理は、２のｐｃ１乗＝2^{pc1}とｐｃｓ１（図７（ａ）参照）のビット毎の論理積をとりそれが2^{pc1}と等しいか否かを比較する演算処理として実現される。
【０１５４】
ＣＰＵ１４０１は、ステップＳ２２０６の判定がＹＥＳならば、ノートタイプをコードトーンと決定し、ノートタイプと隣接音程の配列のノートタイプ要素の位置ｉｎｃｏｎ［ｉ×２］に、ＲＯＭ１４０２からコードトーンを示す定数データｃｉ＿ＣｈｏｒｄＴｏｎｅの値を読み出して格納する（ステップＳ２２０７）。その後、ＣＰＵ１４０１は、図２２のフローチャートで例示される処理すなわち図２０のステップＳ２００５のノートタイプ取得処理を終了する。
【０１５５】
ＣＰＵ１４０１は、ステップＳ２２０６の判定がＮＯならば、入力モチーフピッチクラスｐｃ１がテンションノートピッチクラスセットｐｃｓ２に含まれ、かつ入力モチーフピッチクラスｐｃ２がスケールノートピッチクラスセットｐｃｓ３に含まれるか否かを判定する（ステップＳ２２０８）。この判定演算処理は、２のｐｃ１乗＝2^{pc1}とｐｃｓ２（図７（ｂ）参照）のビット毎の論理積をとりそれが2^{pc1}と等しく、かつ２のｐｃ２乗＝2^{pc2}とｐｃｓ３（図７（ｃ）参照）のビット毎の論理積をとりそれが2^{pc2}と等しいか否かを比較する演算処理として実現される。
【０１５６】
ＣＰＵ１４０１は、ステップＳ２２０８の判定がＹＥＳならば、ノートタイプをアヴェイラブルノートと決定し、ノートタイプと隣接音程の配列のノートタイプ要素の位置ｉｎｃｏｎ［ｉ×２］に、ＲＯＭ１４０２からアヴェイラブルノートを示す定数データｃｉ＿ＡｖａｉｌａｂｌｅＮｏｔｅの値を読み出して格納する（ステップＳ２２０９）。その後、ＣＰＵ１４０１は、図２２のフローチャートで例示される処理すなわち図２０のステップＳ２００５のノートタイプ取得処理を終了する。

【０１５７】
ＣＰＵ１４０１は、ステップＳ２２０８の判定がＮＯならば、入力モチーフピッチクラスｐｃ２がスケールノートピッチクラスセットｐｃｓ３に含まれるか否かを判定する（ステップＳ２２１０）。この判定演算処理は、２のｐｃ２乗＝2^{pc2}とｐｃｓ３（図７（ｃ）参照）のビット毎の論理積をとりそれが2^{pc2}と等しいか否かを比較する演算処理として実現される。
【０１５８】
ＣＰＵ１４０１は、ステップＳ２２１０の判定がＹＥＳならば、ノートタイプをスケールノートと決定し、ノートタイプと隣接音程の配列のノートタイプ要素の位置ｉｎｃｏｎ［ｉ×２］に、ＲＯＭ１４０２からスケールノートを示す定数データｃｉ＿ＳｃａｌｅＮｏｔｅの値を読み出して格納する（ステップＳ２２１１）。その後、ＣＰＵ１４０１は、図２２のフローチャートで例示される処理すなわち図２０のステップＳ２００５のノートタイプ取得処理を終了する。
【０１５９】
ＣＰＵ１４０１は、ステップＳ２２１０の判定がＮＯならば、入力モチーフピッチクラスｐｃ１がテンションノートピッチクラスセットｐｃｓ２に含まれるか否かを判定する（ステップＳ２２１２）。この判定演算処理は、２のｐｃ１乗＝2^{pc1}とｐｃｓ２（図７（ｂ）参照）のビット毎の論理積をとりそれが2^{pc1}と等しいか否かを比較する演算処理として実現される。
【０１６０】
ＣＰＵ１４０１は、ステップＳ２２１２の判定がＹＥＳならば、ノートタイプをテンションノートと決定し、ノートタイプと隣接音程の配列のノートタイプ要素の位置ｉｎｃｏｎ［ｉ×２］に、ＲＯＭ１４０２からテンションノートを示す定数データｃｉ＿ＴｅｎｓｉｏｎＮｏｔｅの値を読み出して格納する（ステップＳ２２１３）。その後、ＣＰＵ１４０１は、図２２のフローチャートで例示される処理すなわち図２０のステップＳ２００５のノートタイプ取得処理を終了する。
【０１６１】
最後に、ＣＰＵ１４０１は、ステップＳ２２１２の判定もＮＯならば、ノートタイプをアヴォイドノートと決定し、ノートタイプと隣接音程の配列のノートタイプ要素の位置ｉｎｃｏｎ［ｉ×２］に、ＲＯＭ１４０２からアヴォイドノートを示す定数データｃｉ＿ＡｖｏｉｄＮｏｔｅの値を読み出して格納する（ステップＳ２２１４）。その後、ＣＰＵ１４０１は、図２２のフローチャートで例示される処理すなわち図２０のステップＳ２００５のノートタイプ取得処理を終了する。
【０１６２】
以上説明した図２２のフローチャートで例示される図２０のステップＳ２００５のノートタイプ取得処理により、入力モチーフ１０８の現在のノートｎｏｔｅｓ［ｉ］のノートタイプが、ノートタイプと隣接音程の配列のノートタイプ要素の位置ｉｎｃｏｎ［ｉ×２］（図７（ｂ）参照）に取得される。
【０１６３】
図２３は、図２０のノート接続性チェック処理の詳細例を示すフローチャートである。この処理は、図１０を用いて前述した処理を実現する。
【０１６４】
まず、ＣＰＵ１４０１は、ＲＡＭ１４０３内の変数データｉＴｏｔａｌＶａｌｕｅに初期値「０」を格納する（ステップＳ２３０１）。このデータは、現在の評価対象のコード進行データ＃ｎ（図１７のステップＳ１７０４参照）についての入力モチーフ１０８に対する適合度を算出するための総合評価点を保持する。
【０１６５】
次に、ＣＰＵ１４０１は、変数データｉについて、ステップＳ２３０２で初期値「０」を格納した後、ステップＳ２３２１で＋１ずつインクリメントしながら、ステップＳ２３０３の判定がＹＥＳ、すなわち変数データｉの値が変数データｉＮｏｔｅＣｎｔの値から

２を減算した値よりも小さい値であると判定される間、ステップＳ２３０３からＳ２３２１までの一連の処理を繰返し実行する。この繰返し処理が、図１０（ｂ）の入力モチーフ１０８中のノートごとのｉ＝０から７までの繰返し処理に対応する。

【０１６６】

入力モチーフ１０８中のｉ番目のノートごとに実行されるステップＳ２３０４からＳ２３２０までの一連の処理において、ＣＰＵ１４０１はまず、ＲＡＭ１４０３内の変数データｉＶａｌｕｅに初期値「０」を格納する（ステップＳ２３０４）。続いて、ＣＰＵ１４０１は、変数データｊについて、ステップＳ２３０６で初期値「０」を格納した後、ステップＳ２３１８で＋１ずつインクリメントしながら、ステップＳ２３０７の判定がＹＥＳ、すなわち変数データｊの値が終端値に達するまでの間、ステップＳ２３０７からＳ２３１９までの一連の処理を繰返し実行する。この繰返し処理が、ｉ番目のノートごとに、変数データｊの値で定まる図９の各ノート接続ルールをチェックする繰返し処理に対応する。

【０１６７】

入力モチーフ１０８中のｉ番目のノートごとに、ｊ番目のノート接続ルールをチェックするステップＳ２３０８からＳ２３１６までの一連の処理において、ＣＰＵ１４０１はＲＡＭ１４０３内の変数データｋについて、ステップＳ２３０８で初期値「０」を格納した後、ステップＳ２３１５で＋１ずつインクリメントしながら、ステップＳ２３０９からステップＳ２３１５の一連の処理を繰返し実行する。この繰返し処理により、入力モチーフ１０８中のｉ番目のノートから４つの連続するノートに対応する４つのノートタイプｉｎｃｏｎ［ｉ×２］、ｉｎｃｏｎ［ｉ×２＋２］、ｉｎｃｏｎ［ｉ×２＋４］、ｉｎｃｏｎ［ｉ×２＋６］のそれぞれと、図９に例示されるｊ番目のノート接続ルール内の４つのノートタイプｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［０］、ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［２］、ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［４］、ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［６］のそれぞれとの一致の有無が判定される。また、入力モチーフ１０８内のｉ番目のノートから４つの連続するノート間の３つの隣接音程ｉｎｃｏｎ［ｉ×２＋１］、ｉｎｃｏｎ［ｉ×２＋３］、ｉｎｃｏｎ［ｉ×２＋５］のそれぞれと、図９に例示されるｊ番目のノート接続ルール内の３つの隣接音程ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［１］、ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［３］、ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［５］のそれぞれとの一致の有無が判定される。

【０１６８】

入力モチーフ１０８中のｉ番目のノートから４つの連続するノートを図９のｊ番目のノート接続ルールと比較する処理として、変数データｋの値が０から３までインクリメントされながらステップＳ２３０９からステップＳ２３１５までの一連の処理が４回繰り返し実行されるうちで、ステップＳ２３１０、Ｓ２３１２、またはＳ２３１４のいずれか１つでも条件が成立すると、現在のｊ番目のノート接続ルールは入力モチーフ１０８に対して不適合となって、ステップＳ２３１９に移行し、変数データｊの値がインクリメントされ次のノート接続ルールの適合評価に処理が移行する。

【０１６９】

具体的には、ステップＳ２３１０で、ＣＰＵ１４０１は、入力モチーフ１０８のｉ＋ｋ番目のノートのノートタイプｉｎｃｏｎ［ｉ×２＋ｋ×２］と、ｊ番目のノート接続ルールのｋ番目のノートタイプｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［ｋ×２］とが不一致となったか否かを判定する。ステップＳ２３１０の判定がＹＥＳになると、ＣＰＵ１４０１は、そのノート接続ルールの少なくとも１つのノートタイプが入力モチーフ１０８内の現在の処理対象（ｉ番目）のノートから始まる４つのノートのノートタイプの少なくとも１つと一致しないため、ステップＳ２３１９に移行する。

【０１７０】

ステップＳ２３１０の判定がＮＯならば、ステップＳ２３１１およびステップＳ２３１２が実行されるが、これらについては後述する。ステップＳ２３１１およびＳ２３１２の判定がともにＮＯとなった後、ＣＰＵ１４０１は、変数データｋの値が３より小さい場合

に、ステップＳ２３１３の判定がＹＥＳとなって、ステップＳ２３１４で隣接音程に関する判定処理を実行する。ステップＳ２３１３の判定が行われるのは、ｋ＝３となる入力モチーフ１０８の４ノート目については、それ以降には隣接音程は存在しないため、変数データｋの値が０から２までの範囲でのみ、隣接音程の判定処理を実行するためである。ステップＳ２３１４において、ＣＰＵ１４０１は、入力モチーフ１０８のｉ＋ｋ番目のノートとｉ＋ｋ＋１番目のノートの間の隣接音程ｉｎｃｏｎ［ｉ×２＋ｋ×２＋１］と、ｊ番目のノート接続ルールのｋ番目のノートタイプとｋ＋１番目のノートタイプの間の隣接音程ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［ｋ×２＋１］とが不一致であり、かつ、ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［ｋ×２＋１］の値が「９９」と不一致であるか否かを判定する。隣接音程の値「９９」は、その隣接音程がどの値でもよいことを示している。ステップＳ２３１４の判定がＹＥＳになると、ＣＰＵ１４０１は、そのノート接続ルールの少なくとも１つの隣接音程が入力モチーフ１０８内の現在の処理対象（ｉ番目）のノートから始まる４つのノートの隣接ノート間の隣接音程の少なくとも１つと一致しないため、ステップＳ２３１９に移行する。

【０１７１】

上記一連の処理で、ステップＳ２３１０において、入力モチーフ１０８のｉ＋ｋ番目のノートのノートタイプｉｎｃｏｎ［ｉ×２＋ｋ×２］と、ｊ番目のノート接続ルールのｋ番目のノートタイプｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［ｋ×２］の一致が検出されてステップＳ２３１０の判定がＮＯとなった後、ＣＰＵ１４０１は、ｊ番目のノート接続ルールのｋ番目の次のｋ＋１番目のノートタイプｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［ｋ×２＋２］がｃｉ＿ＮｕｌｌＮｏｔｅＴｙｐｅであるか否かを判定する（ステップＳ２３１１）。

【０１７２】

ｃｉ＿ＮｕｌｌＮｏｔｅＴｙｐｅが設定されるのは、図９のｊ＝０から８までのノート接続ルールにおけるｋ＝３の場合のｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［６］に対してである。従って、ステップＳ２３１１の判定がＹＥＳとなるケースは、変数データｊの値の範囲が０から８の間であって、変数データｋの値が０、１、２の３音分についてノートタイプおよび隣接音程が一致して、ｋ＝２となっている場合である。前述したように、ｊ＝０～８の範囲のノート接続ルールは３音のルールであるため、４音目はｃｉ＿ＮｕｌｌＮｏｔｅＴｙｐｅとなって評価をする必要がない。従って、ステップＳ２３１１の判定がＹＥＳとなる場合には、そのときのノート接続ルールは入力モチーフ１０８内のｉ番目のノートから始まる３つのノートと適合する。このため、ステップＳ２３１１の判定がＹＥＳならば、ＣＰＵ１４０１は、ステップＳ２３１６に移行して、変数データｉＶａｌｕｅに、そのノート接続ルールの評価点ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［７］（図９参照）を累算する。

【０１７３】

一方、ステップＳ２３１１の判定がＮＯとなる場合は、ステップＳ２３１２およびＳ２３１３を経てステップＳ２３１４の隣接音程の評価処理に進む。ここで、ＣＰＵ１４０１は、ステップＳ２３１１の判定がＮＯとなった直後のステップＳ２３１２で、変数データｉの値が入力モチーフ１０８のノート数を示す変数データｉＮｏｔｅＣｎｔの値から３を減算した値に等しく、かつ変数データｋの値が２に等しいか否かを判定する。このケースでは、処理対象となる入力モチーフ１０８のノートは、ｉ＋ｋ番目、すなわちｉＮｏｔｅＣｎｔ－３＋２＝ｉＮｏｔｅＣｎｔ－１番目、つまり、入力モチーフ１０８中の最後のノートになる。この状態で、ステップＳ２３１１において、ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［ｋ×２＋２］＝ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［６］の値がｃｉ＿ＮｕｌｌＮｏｔｅＴｙｐｅにならない場合は、図９のｊの値が９以上のノート接続ルールが処理されている場合である。つまり、ノート接続ルールは、４音についてのものである。一方、この場合における入力モチーフ１０８中の処理対象のノートは、ｉ＝ｉＮｏｔｅＣｎｔ－３から始まり最終ノートのｉ＝ｉＮｏｔｅＣｎｔ－１までの３音である。従って、このケースでは、入力モチーフ１０８中の処理対象のノートの数とノート接続ルール中の音の

数が合わないため、そのノート接続ルールは入力モチーフ１０８に適合することはない。従って、ステップＳ２３１２の判定がＹＥＳとなる場合は、ＣＰＵ１４０１は、そのノート接続ルールに関する適合評価を行わずに、ステップＳ２３１９に移行する。

【０１７４】

上述したステップＳ２３１０、Ｓ２３１１、Ｓ２３１２、およびＳ２３１４のいずれの条件も成立せずに、ステップＳ２３０９からＳ２３１５までの一連の処理が４回繰り返されてステップＳ２３０９の判定がＮＯになると、入力モチーフ１０８中のｉ番目のノートから４つの連続するノートに関して、ノートタイプと隣接音程が全て現在のｊ番目のノート接続ルールのノートタイプおよび隣接音程と適合したことになる。この場合には、ＣＰＵ１４０１は、ステップＳ２３１６に移行して、変数データｉＶａｌｕｅに、現在のｊ番目のノート接続ルールの評価点ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［７］（図９参照）を累算する。

【０１７５】

なお、１つのノート接続ルールのみが入力モチーフ１０８に適合するとは限らず、例えば３音のノート接続ルールに適合しかつ４音のノート接続ルールにも適合する場合があり得る。

【０１７６】

そこで、ＣＰＵ１４０１は、ステップＳ２３１９で変数データｊの値がインクリメントされながらステップＳ２３０７で全てのノート接続ルールに関する評価が完了するまで、ステップＳ２３０９の判定がＮＯまたはステップＳ２３１１の判定がＹＥＳとなってノート接続ルールが適合するごとに、ステップＳ２３１６において、新たに適合したノート接続ルールの評価点ｃｉ＿ＮｏｔｅＣｏｎｎｅｃｔ［ｊ］［７］が変数データｉＶａｌｕｅに累算される。

【０１７７】

その後、ＣＰＵ１４０１は、変数データｊの値を＋１インクリメントして次のノート接続ルールの評価に移行し（ステップＳ２３１９）、ステップＳ２３０７の判定処理に戻る。

【０１７８】

ＣＰＵ１４０１は、全てのノート接続ルールに対する評価が完了してステップＳ２３０７の判定がＹＥＳになると、現在のコード進行データ＃ｎに対応する変数データｉＴｏｔａｌＶａｌｕｅに、変数データｉＶａｌｕｅに累算されている評価点を累算する（ステップＳ２３２０）。

【０１７９】

その後、ＣＰＵ１４０１は、変数ｉの値を＋１インクリメントし（ステップＳ２３２１）、ステップＳ２３０３の判定処理に戻って、入力モチーフ１０８中の次のノートに処理を移す（図１０（ｂ）参照）。

【０１８０】

ＣＰＵ１４０１は、入力モチーフ１０８中の全てのノートについて全てのノート接続ルールの適合評価の処理を終了すると、ステップＳ２３０３の判定がＮＯとなる。ここで、入力モチーフ１０８中の処理対象のノートの終了位置は、本来は入力モチーフ１０８中の最終ノートを含む４音手前のノートであり、それに対応する変数データｉの値は「（ｉＮｏｔｅＣｎｔ－１）－３＝ｉＮｏｔｅＣｎｔ－４」である。しかし、図１０（ｂ）のｉ＝７として例示されるように、最後の処理は３音で行われるため、終了位置に対応する変数データｉの値は、「ｉＮｏｔｅＣｎｔ－３」となる。よって、ステップＳ２３０３の終了判定は、「ｉ＜ｉＮｏｔｅＣｎｔ－２」がＮＯになる場合となる。

【０１８１】

ステップＳ２３０３の判定がＮＯになると、ＣＰＵ１４０１は、変数データｉＴｏｔａｌＶａｌｕｅの値を入力モチーフ１０８中の処理したノート数（ｉＮｏｔｅＣｎｔ－２）で除算して正規化し、その除算結果をコード進行＃ｎの入力モチーフ１０８に対する適合度として、変数データｄｏＶａｌｕｅに格納する（ステップＳ２３２２）。その後、ＣＰ

Ｕ１４０１は、図２３のフローチャートすなわち図２０のステップＳ２００９のノート接続性チェック処理を終了する。

【０１８２】

図２４は、図１６の自動作曲処理において、ステップＳ１６０７のコード進行選択処理の次に実行されるステップＳ１６０８のメロディ生成処理の詳細例を示すフローチャートである。

【０１８３】

まず、ＣＰＵ１４０１は、ＲＡＭ１４０３の変数領域を初期化する（ステップＳ２４０１）。

【０１８４】

次に、ＣＰＵ１４０１は、図１６のステップＳ１６０７のコード進行選択処理によって選択され例えばユーザによって指示されたコード進行候補に対応する曲構造データ（図６参照）を、伴奏・コード進行ＤＢ１０３から読み込む（ステップＳ２４０２）。

【０１８５】

その後、ＣＰＵ１４０１は、変数データｉの値を初期値「０」に設定した後（ステップＳ２４０３）、ステップＳ２４０９でｉの値をインクリメントしながら、ステップＳ２４０４で曲構造データの終端に達したと判定するまで、変数データｉによって指示される曲構造データ上の小節のフレーズごとに、入力モチーフ１０８と、ＲＯＭ１４０２に記憶されるフレーズセットＤＢ１０６に登録されているフレーズセット（図１１参照）、およびＲＯＭ１４０２に記憶されるルールＤＢ１０４（図９参照）を参照しながら、そのフレーズのメロディを自動生成する。変数データｉは、その値がステップＳ２４０９で０から＋１ずつインクリメントされることにより、図６に例示される曲構造データの「Ｍｅａｓｕｒｅ」項目の値を順次指定して、曲構造データ上の各レコードを指定してゆく。

【０１８６】

具体的には、まず、ＣＰＵ１４０１は、曲構造データの終端に達したか否かを判定する（ステップＳ２４０４）。

【０１８７】

ステップＳ２４０４の判定がＮＯならば、ＣＰＵ１４０１は、変数データｉによって指定される曲構造データの現在の小節が、入力モチーフ１０８が入力された小節と一致するか否かを判定する（ステップＳ２４０５）。

【０１８８】

ステップＳ２４０５の判定がＹＥＳならば、ＣＰＵ１４０１は、その入力モチーフ１０８をそのままメロディ１１０（図１参照）の一部として、例えばＲＡＭ１４０３上の出力メロディ領域に出力する。

【０１８９】

ステップＳ２４０５の判定がＮＯならば、ＣＰＵ１４０１は、現在の小節が、サビメロディの先頭小節であるか否かを判定する（ステップＳ２４０６）。

【０１９０】

ステップＳ２４０６の判定がＮＯならば、ＣＰＵ１４０１は、メロディ生成１処理を実行する（ステップＳ２４０７）。

【０１９１】

一方、ステップＳ２４０６の判定がＹＥＳならば、ＣＰＵ１４０１は、メロディ生成２処理を実行する（ステップＳ２４０８）。

【０１９２】

ステップＳ２４０７またはＳ２４０８の処理の後、ＣＰＵ１４０１は、変数データｉを＋１インクリメントする（ステップＳ２４０９）。その後、ＣＰＵ１４０１は、ステップＳ２４０４の判定処理に戻る。

【０１９３】

図２５は、図２４のステップＳ２４０７のメロディ生成１処理の詳細例を示すフローチャートである。

【０１９４】
ＣＰＵ１４０１は、現在の小節が含まれるフレーズの種別が、入力モチーフ１０８のフレーズの種別と同じであるか否かを判定する（ステップＳ２５０１）。現在の小節が含まれるフレーズの種別は、図６に例示される曲構造データ中で、変数データｉの値に対応する「Ｍｅａｓｕｒｅ」項目を有するレコード中の「ＰａｒｔＮａｍｅ［Ｍ］」項目または「ｉＰａｒｔＩＤ［Ｍ］」項目を参照することにより、判定することができる。入力モチーフ１０８のフレーズの種別は、ユーザが入力モチーフ１０８を入力するときに指定する。
【０１９５】
ステップＳ２５０１の判定がＹＥＳならば、ＣＰＵ１４０１は、入力モチーフ１０８のメロディを現在の小節のメロディとしてＲＡＭ１４０３の所定領域にコピーする。その後、ＣＰＵ１４０１は、ステップＳ２５０７のメロディ変形処理に移行する。
【０１９６】
ステップＳ２５０１の判定がＮＯならば、ＣＰＵ１４０１は、現在の小節が含まれるフレーズの種別に対して、既にメロディが生成済みで、かつ小節の偶数／奇数が一致するか否かを判定する（ステップＳ２５０３）。
【０１９７】
ステップＳ２５０３の判定がＹＥＳならば、ＣＰＵ１４０１は、生成済みのメロディを現在の小節のメロディとしてＲＡＭ１４０３の所定領域にコピーする（ステップＳ２５０４）。その後、ＣＰＵ１４０１は、ステップＳ２５０７のメロディ変形処理に移行する。
【０１９８】
該当するフレーズのメロディが未だ生成されていなければ（ステップＳ２５０３の判定がＮＯ）、ＣＰＵ１４０１は、フレーズセットＤＢ検索処理を実行する（ステップＳ２５０５）。フレーズセットＤＢ検索処理において、ＣＰＵ１４０１は、フレーズセットＤＢ１０６から入力モチーフ１０８に対応するフレーズセットを抽出する。
【０１９９】
ＣＰＵ１４０１は、ステップＳ２５０５で検索されたフレーズセット中の、現在の小節が含まれるフレーズの種別と同じ種別のフレーズのメロディを、ＲＡＭ１４０３の所定領域にコピーする（ステップＳ２５０６）。その後、ＣＰＵ１４０１は、ステップＳ２５０７のメロディ変形処理に移行する。
【０２００】
ステップＳ２５０２、Ｓ２５０４、またはＳ２５０６の処理の後、ＣＰＵ１４０１は、コピーしたメロディを変形するメロディ変形処理を実行する（ステップＳ２５０７）。
【０２０１】
さらに、ＣＰＵ１４０１は、ステップＳ２５０７で変形したメロディを構成する各ノートのピッチを最適化するメロディ最適化処理を実行する（ステップＳ２５０８）。この結果、ＣＰＵ１４０１は、曲構造データによって示される小節のフレーズのメロディを自動生成し、ＲＡＭ１４０３の出力メロディ領域に出力する。
【０２０２】
図２６は、図２５のステップＳ２５０５のフレーズセットＤＢ検索処理の詳細例を示すフローチャートである。
【０２０３】
まず、ＣＰＵ１４０１は、入力モチーフ１０８のピッチ列を取り出し、ＲＡＭ１４０３内の配列変数データｉＭｅｌｏｄｙＢ［０］～ｉＭｅｌｏｄｙＢ［ｉＬｅｎｇｔｈＢ－１］に格納する。ここで、変数データｉＬｅｎｇｔｈＢには、入力モチーフ１０８のピッチ列の長さが格納される。
【０２０４】
次に、ＣＰＵ１４０１は、変数データｋの値を初期値「０」に設定した後（ステップＳ２６０２）、ステップＳ２６０９でｋの値をインクリメントしながら、ステップＳ２６０３でフレーズセットＤＢ１０６の終端（図１１（ａ）参照）に達したと判定するまで、変

数データｋによって指示されるフレーズセット（図１１（ａ）参照）について、ステップＳ２６０３からＳ２６０９の一連の処理を繰り返し実行する。

【０２０５】

この一連の処理において、まず、ＣＰＵ１４０１は、変数データｋが示すｋ番目のフレーズセット内の、入力モチーフ１０８に対応するフレーズのピッチ列を取り出して、ＲＡＭ１４０３内の配列変数データｉＭｅｌｏｄｙＡ［０］～ｉＭｅｌｏｄｙＡ［ｉＬｅｎｇｔｈＡ－１］に格納する（ステップＳ２６０４）。ここで、変数データｉＬｅｎｇｔｈＡには、フレーズセットＤＢ１０６内のフレーズのピッチ列の長さが格納される。

【０２０６】

次に、ＣＰＵ１４０１は、ステップＳ２６０１で設定した入力モチーフ１０８のピッチ列の配列変数データｉＭｅｌｏｄｙＢ［０］～ｉＭｅｌｏｄｙＢ［ｉＬｅｎｇｔｈＢ－１］と、ステップＳ２６０４で設定したフレーズセットＤＢ１０６内のｋ番目のフレーズセット内の該当フレーズのピッチ列の配列変数データｉＭｅｌｏｄｙＡ［０］～ｉＭｅｌｏｄｙＡ［ｉＬｅｎｇｔｈＡ－１］との間で、ＤＰ（Ｄｙｎａｍｉｃ　Ｐｒｏｇｒａｍｍｉｎｇ：動的計画法）マッチング処理を実行し、その結果算出される両者間の距離評価値をＲＡＭ１４０３上の変数データｄｏＤｉｓｔａｎｃｅに格納する。

【０２０７】

次に、ＣＰＵ１４０１は、ＲＡＭ１４０３上の変数データｄｏＭｉｎが示す最小距離評価値のほうが、ステップＳ２６０５のＤＰマッチング処理により新たに算出した距離評価値ｄｏＤｉｓｔａｎｃｅよりも大きくなったか否かを判定する（ステップＳ２６０８）。

【０２０８】

ステップＳ２６０８の判定がＹＥＳならば、ＣＰＵ１４０１は、変数データｄｏＤｉｓｔａｎｃｅに格納されている新たな距離評価値を、変数データ変数データｄｏＭｉｎに格納する（ステップＳ２６０７）。

【０２０９】

また、ＣＰＵ１４０１は、変数データｋの値を、ＲＡＭ１４０３上の変数データｉＢｅｓｔＭｏｃｈｉｅｆに格納する（ステップＳ２６０８）。

【０２１０】

ステップＳ２６０８の判定がＹＥＳならば、ＣＰＵ１４０１は、ステップＳ２６０７およびＳ２６０８の処理はスキップする。

【０２１１】

その後、ＣＰＵ１４０１は、変数データｋの値を＋１インクリメントしてフレーズセットＤＢ１０６内の次のフレーズセット（図１１（ａ）参照）に対する処理に移行する。

【０２１２】

ＣＰＵ１４０１は、フレーズセットＤＢ１０６内の全てのフレーズセットに対する入力モチーフ１０８とのＤＰマッチング処理を終了し、ステップＳ２６０３の判定がＹＥＳになると、変数データｉＢｅｓｔＭｏｃｈｉｅｆが示す番号のフレーズセットＤＢ１０６内のフレーズセットを、ＲＡＭ１４０３上の所定の領域に出力する（ステップＳ２６１０）。その後、ＣＰＵ１４０１は、図２６に例示されるフローチャートの処理すなわち図２５のステップＳ２５０５のフレーズセットＤＢ検索処理を終了する。

【０２１３】

図２７は、図２５のステップＳ２５０７のメロディ変形処理の詳細例を示すフローチャートである。この処理は、図１２の説明で前述したピッチシフトまたは左右反転によるメロディ変形処理を実行する。

【０２１４】

まず、ＣＰＵ１４０１は、図２５のコピー処理により得られたメロディのノート数をカウントするＲＡＭ１４０３内の変数ｉに初期値「０」を格納する（ステップＳ２７０１）。その後、ＣＰＵ１４０１は、ステップＳ２７０９で変数ｉの値を＋１ずつインクリメントさせながら、ステップＳ２７０２で変数ｉの値がメロディのノート数を示す変数データｉＮｏｔｅＣｎｔの値よりも小さいと判定される間、ステップＳ２７０２からＳ２７０９

までの一連の処理を実行する。

【０２１５】

ステップＳ２７０２からＳ２７０９の繰返し処理において、ステップＳＣＰＵ１４０１はまず、変形タイプを取得する（ステップＳ２７０２）。変形タイプは、ピッチシフトまたは左右反転があり、ユーザが特には図示しないスイッチにより指定することができる。

【０２１６】

変形タイプがピッチシフトである場合には、ＣＰＵ１４０１は、配列変数データｎｏｔｅ［ｉ］のｉＰｉｔ項目に得られているピッチデータｎｏｔｅ［ｉ］－＞ｉＰｉｔに、所定値を加算することにより、例えば図１２の１２０１として説明したような例えば２半音上へのピッチシフトを実行する（ステップＳ２７０４）。

【０２１７】

変形タイプが左右反転である場合には、ＣＰＵ１４０１は、変数データｉの値が変数データｉＮｏｔｅＣｎｔの値を２で割った値よりも小さいか否かを判定する（ステップＳ２７０５）。

【０２１８】

ステップＳ２７０５の判定がＹＥＳの場合には、まず、ＣＰＵ１４０１は、配列変数データｎｏｔｅ［ｉ］のｉＰｉｔ項目に得られているピッチデータｎｏｔｅ［ｉ］－＞ｉＰｉｔを、ＲＡＭ１４０３上の変数ｉｐに退避させる（ステップＳ２７０６）。

【０２１９】

次に、ＣＰＵ１４０１は、（ｉＮｏｔｅＣｎｔ－ｉ－１）番目の配列要素のピッチ項目ｎｏｔｅ［ｉＮｏｔｅＣｎｔ－ｉ－１］－＞ｉＰｉｔの値を、ｉ番目の配列要素のピッチ項目ｎｏｔｅ［ｉ］－＞ｉＰｉｔに格納する（ステップＳ２７０７）。

【０２２０】

そして、ＣＰＵ１４０１は、変数データｉｐに退避させていた元のピッチ項目値を、（ｉＮｏｔｅＣｎｔ－ｉ－１）番目の配列要素のピッチ項目ｎｏｔｅ［ｉＮｏｔｅＣｎｔ－ｉ－１］－＞ｉＰｉｔに格納する（ステップＳ２７０８）。

【０２２１】

ステップＳ２７０５の判定がＮＯの場合には、ＣＰＵ１４０１は、ステップＳ２７０６、Ｓ２７０７、Ｓ２７０８の処理をスキップする。

【０２２２】

ステップＳ２７０４またはＳ２７０８の処理の後、あるいはステップＳ２７０５の判定がＮＯとなった後に、ＣＰＵ１４０１は、ステップＳ２７０９で、変数データｉの値を＋１インクリメントし、次のノートに対する処理に移行してステップＳ２７０２の判定処理に戻る。

【０２２３】

以上の処理により、図１２の１２０２で説明した左右反転処理が実現される。

【０２２４】

図２８は、図２５のステップＳ２５０８のメロディ最適化処理の詳細例を示すフローチャートである。この処理は、図１３の説明で前述したピッチの最適化処理を実現する。

【０２２５】

まず、ＣＰＵ１４０１は、次式により、別ピッチ候補の全組合せ数を算出する（ステップＳ２８０１）。

【０２２６】

ＩＷｎｕｍ＝ＭＡＸ＿ＮＯＴＥ＿ＣＡＮＤＩＤＡＴＥ＾ｉＮｏｔｅＣｎｔ

【０２２７】

ここで、演算子「＾」は、べき乗演算を示す。また、ＲＯＭ１４０２上の定数データＭＡＸ＿ＮＯＴＥ＿ＣＡＮＤＩＤＡＴＥは、図１３に示される１つのノートに対する別ピッチ候補ｉｐｉｔｄ［０］～ｉｐｉｔｄ［４］の候補数を示し、この例では５である。

【０２２８】

次に、ＣＰＵ１４０１は、別ピッチ候補のカウント用の変数データｉＣｎｔを初期値「

０」に設定した後（ステップＳ２８０２）、ステップＳ２８１８で変数データｉＣｎｔを＋１ずつインクリメントしながら、ステップＳ２８０３で変数データｉＣｎｔの値がステップＳ２８０１で算出した別ピッチ候補の全組合せ数より小さい範囲で、入力されたメロディのピッチを変更しながら、そのメロディの妥当性を評価する。

【０２２９】

ＣＰＵ１４０１は、変数データｉＣｎｔの値がインクリメントされるごとに、ステップＳ２８０５からＳ２８１７までの一連の処理を実行する。

【０２３０】

まず、ＣＰＵ１４０１は、図２５のコピー処理により得られたメロディのノート数をカウントするＲＡＭ１４０３内の変数ｉに初期値「０」を格納する（ステップＳ２８０５）。その後、ＣＰＵ１４０１は、ステップＳ２８１３で変数ｉの値を＋１ずつインクリメントさせながら、ステップＳ２８０６で変数ｉの値がメロディのノート数を示す変数データｉＮｏｔｅＣｎｔの値よりも小さいと判定される間、ステップＳ２８０６からＳ２８１３までの一連の処理を繰り返し実行する。この繰返し処理によって、メロディの全てのノートに対して、ステップＳ２８０７、Ｓ２８０８、およびＳ２８０９によって、ピッチ修正が行われる。

【０２３１】

まず、ＣＰＵ１４０１は、次式を演算することによって、ピッチ修正値をＲＡＭ１４０３上の変数データｉｐｉｔｄｅｖに得る（ステップＳ２８０７）。

【０２３２】

ｉｐｉｔｄｅｖ＝ｉｐｉｔｄ［（ｉＣｎｔ／ＭＡＸ＿ＮＯＴＥ＿ＣＡＮＤＩＤＡＴＥ＾ｉ）ｍｏｄＭＡＸ＿ＮＯＴＥ＿ＣＡＮＤＩＤＡＴＥ］

ここで、「ｍｏｄ」は、剰余演算を示す。

【０２３３】

次に、ＣＰＵ１４０１は、入力したメロディのピッチ項目値ｎｏｔｅ［ｉ］－＞ｉＰｉｔに、ステップＳ２８０７で算出された変数データｉｐｉｔｄｅｖの値を加算し、その結果をピッチ情報列を示す配列変数データｉｐｉｔ［ｉ］に格納する（ステップＳ８０９）。

【０２３４】

次に、前述した図２０のステップＳ２００５～Ｓ２００７と同様にして、ピッチ情報列を示す配列変数データｉｐｉｔ［ｉ］に対して、ノートタイプ取得処理（ステップＳ２８１０）と、隣接音程の算出処理（ステップＳ２８１１およびＳ２８１２）を実行する。

【０２３５】

ＣＰＵ１４０１は、入力メロディを構成する全てのノートに対して、現在の変数データｉＣｎｔの値に対応するピッチ修正が完了すると、ステップＳ２８０６の判定がＮＯなる。この結果、ＣＰＵ１４０１は、ステップＳ２８１４において、ステップＳ２８１０～Ｓ２８１２で算出されたメロディを構成するノートごとのノートタイプおよび隣接音程に対して、前述した図２３の処理と同じノート接続性チェック処理を実行する（ステップＳ２８１４）。なお、このとき、入力されたメロディの小節に該当するコード進行データ中のコード情報が抽出されて使用される。

【０２３６】

ＣＰＵ１４０１は、ステップＳ２８１４のノート接続性チェック処理で変数データｄｏＶａｌｕｅに新たに得られた適合度の値が、変数データｉＭａｘＶａｌｕｅに保持されている最良適合度の値よりも大きいか否かを判定する（ステップＳ２８１５）。

【０２３７】

ステップＳ２８１５の判定がＹＥＳならば、ＣＰＵ１４０１は、変数データｉＭａｘＶａｌｕｅの値を変数データｄｏＶａｌｕｅの値で置き換え（ステップＳ２８１６）、変数データｉＭａｘＣｎｔの値を変数データｉＣｎｔの値で置き換える（ステップＳ２８１７）。

【０２３８】

その後、ＣＰＵ１４０１は、変数データｉＣｎｔの値を＋１インクリメントし（ステップＳ２８１８）、ステップＳ２８０３の判定処理に戻る。

【０２３９】

以上の動作が、順次インクリメントされる変数データｉＣｎｔの値に対して繰返し実行された結果、別ピッチ候補の全ての組合せに対してノート接続性をチェックする処理が完了すると、ステップＳ２８０３の判定がＮＯとなる。

【０２４０】

この結果、ＣＰＵ１４０１は、変数ｉに初期値「０」を格納した後（ステップＳ２８１９）、ステップＳ２８２３で変数ｉの値を＋１ずつインクリメントさせながら、ステップＳ２８２０で変数ｉの値がメロディのノート数を示す変数データｉＮｏｔｅＣｎｔの値よりも小さいと判定される間、ステップＳ２８２０からＳ２８２３までの一連の処理を繰り返し実行する。この繰返し処理によって、メロディの全てのノートに対して、変数データｉＭａｘＣｎｔに得られている最良値を用いて、ピッチの修正すなわち最適化が実行される。

【０２４１】

具体的には、ＣＰＵ１４０１は、ステップＳ２８２０の終了判定を行った後、次式を演算することによって、最適なピッチ修正値を、ピッチ情報列の配列変数データｐｉｔ［ｉ］に得る（ステップＳ２８２１）。

【０２４２】

ｉｐｉｔ［ｉ］＝ｎｏｔｅ［ｉ］－＞ｉＰｉｔ＋ｉｐｉｔｄ［（ｉＭａｘＣｎｔ／（ＭＡＸ＿ＮＯＴＥ＿ＣＡＮＤＩＤＡＴＥ＾ｉ）ｍｏｄＭＡＸ＿ＮＯＴＥ＿ＣＡＮＤＩＤＡＴＥ）］

【０２４３】

そして、ＣＰＵ１４０１は、ピッチ情報列の配列変数データｐｉｔ［ｉ］の値を、入力されたメロディのノートデータのピッチ項目値ｎｏｔｅ［ｉ］－＞ｉＰｉｔに上書きコピーする（ステップＳ２８２２）。

【０２４４】

最後に、ＣＰＵ１４０１は、変数ｉの値をインクリメントし（ステップＳ２８２３）、その後ステップＳ２８２０の判定処理に戻る。

【０２４５】

ＣＰＵ１４０１は、入力されたメロディを構成する全てのノートデータに対する上記処理が完了すると、ステップＳ２８２０の判定がＮＯになって、図２８のフローチャートで例示される処理すなわち図２５のステップＳ２５０８のメロディ最適化処理を終了する。

【０２４６】

図２９は、図２４のメロディ生成２処理（サビ先頭メロディ生成処理）の詳細例を示すフローチャートである。

【０２４７】

まず、ＣＰＵ１４０１は、サビ先頭メロディは生成済みか否かを判定する（ステップＳ２９０１）。

【０２４８】

サビ先頭メロディはまだ生成されておらずステップＳ２９０１の判定がＮＯならば、ＣＰＵ１４０１は、フレーズセットＤＢ検索処理を実行する（ステップＳ２９０２）。この処理は、図２５のステップＳ２５０５に対応する図２６の処理と同じである。このフレーズセットＤＢ検索処理により、ＣＰＵ１４０１は、フレーズセットＤＢ１０６から入力モチーフ１０８に対応するフレーズセットを抽出する。

【０２４９】

次に、ＣＰＵ１４０１は、ステップＳ２９０２で検索されたフレーズセット中の、サビ先頭（Ｃメロ）のフレーズのメロディを、ＲＡＭ１４０３の所定領域にコピーする（ステップＳ２９０３）。

【０２５０】

続いて、ＣＰＵ１４０１は、ステップＳ２９０３で得たメロディに対して、図２５のステップＳ２５０８と同様の図２８で示されるメロディ最適化処理を実行する（ステップＳ２９０４）。

【０２５１】

ＣＰＵ１４０１は、ステップＳ２９０４で得られたピッチが最適化されたメロディデータを、メロディ１１０の一部として、ＲＡＭ１４０３の出力メロディ領域に格納する。その後、ＣＰＵ１４０１は、図２９のフローチャートで例示される処理すなわち図２４のメロディ生成２処理（サビ先頭メロディ生成処理）を終了する。

【０２５２】

サビ先頭メロディは生成されておりステップＳ２９０１の判定がＹＥＳならば、ＣＰＵ１４０１は、生成済みのサビ先頭メロディを現在の小節のメロディとして、ＲＡＭ１４０３の出力メロディ領域にコピーする（ステップＳ２９０５）。その後、ＣＰＵ１４０１は、図２９のフローチャートで例示される処理すなわち図２４のメロディ生成２処理（サビ先頭メロディ生成処理）を終了する。

【０２５３】

以上説明した実施形態によれば、ほとんどの場合で、コードやスケールに対して最適なメロディ（音列）を保障することが可能となる。また、コピーした音列や任意に変形したものや、またユーザが入力したものまで含めて、利用することが可能となる。

【０２５４】

以上の実施形態に関して、更に以下の付記を開示する。

（付記１）

入力されたフレーズに含まれる各ノートデータのピッチを順次ピッチシフトさせるノートピッチシフト部と、

前記ピッチシフトが行われるごとに、連続するノートタイプの接続関係を規定する複数種のノート接続ルールを参照しながら、指定されたコード進行データと前記ピッチシフトされたフレーズとの適合度を算出する適合度算出部と、

前記算出された適合度に基づいて選択される前記ピッチシフトされたフレーズに基づいてメロディを生成するメロディ生成部と、

を備えた自動作曲装置。

（付記２）

前記メロディ生成部は、前記算出された適合度が最も高くなるシフト量だけピッチシフトさせたフレーズを選択するとともに、当該選択されたフレーズに基づいてメロディを生成する、付記１に記載の自動作曲装置。

（付記３）

前記自動作曲装置はさらに、前記複数のノート接続ルールを記憶するルールデータベースを有する、付記１または２に記載の自動作曲装置。

（付記４）

前記自動作曲装置はさらに、

夫々種別の異なるフレーズを組み合わせたフレーズセットを複数種記憶するフレーズセットデータベースと、

外部より入力されたフレーズの種別と同じ種別のフレーズであって、前記入力モチーフとして入力されたフレーズに類似するフレーズを含むフレーズセットであって、当該フレーズセット内の予め指定された種別のフレーズを検索する検索部と、

を有する付記１乃至３のいずれかに記載の自動作曲装置。

（付記５）

前記フレーズセットは、それぞれ種別の異なるフレーズとして、Ａメロ、Ｂメロ、及びサビメロのいずれかを含むフレーズを有する、付記４に記載の自動作曲装置。

（付記６）

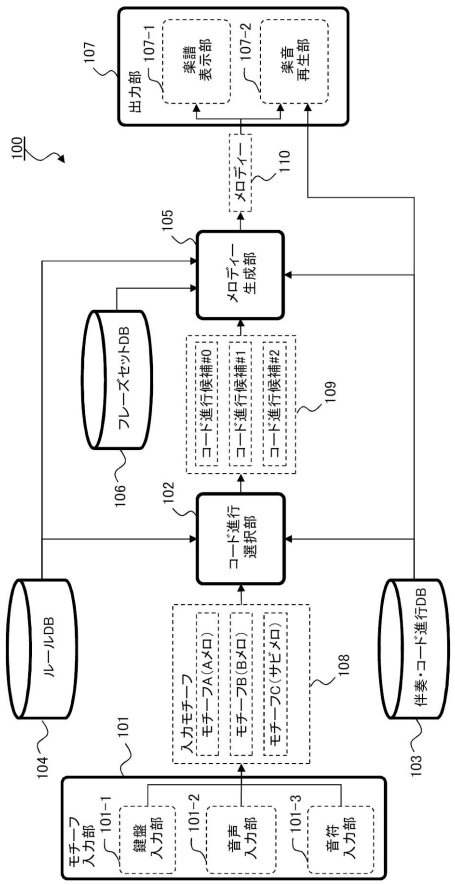
前記自動作曲装置はさらに、前記検索されたフレーズを変形させる変形部を有し、

前記ピッチシフト部は、前記変形させたフレーズを構成する各ノートデータのピッチを

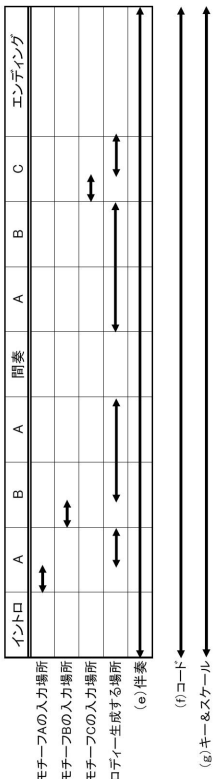
予め定められた範囲内で順次ピッチシフトさせる、付記４に記載の自動作曲装置。
（付記７）
前記変形部は、前記フレーズを構成する各ノートデータに含まれるピッチを予め定められた値だけシフトする、付記６に記載の自動作曲装置。
（付記８）
前記変形部は、前記フレーズを構成するノートデータの並び順を変更する、付記６に記載の自動作曲装置。
（付記９）
前記ノート接続ルールは、複数の連続するノートタイプの接続関係を規定するとともに、さらに隣接する当該ノートタイプ間の音程を規定し、
前記メロディ生成部は、前記モチーフを構成する各ノートデータについて、当該ノートデータの発音タイミングに対応する前記コード進行データ上でのノートタイプと、隣接する当該ノート間の音程とを算出するとともに、当該ノートタイプおよび音程を、前記ノート接続ルールを構成するノートタイプおよび音程と比較することにより、前記コード進行データの前記モチーフに対する適合度を算出する付記１に記載の自動作曲装置。
（付記１０）
前記自動作曲装置はさらに、前記メロディ生成部により生成されたメロディに基づいた楽曲を再生する再生部、及び当該楽曲を表す楽譜を表示する楽譜表示部の少なくとも一方を有する付記１乃至９のいずれかに記載の自動作曲装置。
（付記１１）
自動作曲装置が、
入力されたフレーズに含まれる各ノートデータのピッチを順次ピッチシフトさせ、
前記ピッチシフトが行われるごとに、連続するノートタイプの接続関係を規定する複数種のノート接続ルールを参照しながら、指定されたコード進行データと前記ピッチシフトされたフレーズとの適合度を算出し、
前記算出された適合度に基づいて選択される前記ピッチシフトされたフレーズに基づいてメロディを生成する、自動作曲方法。
（付記１２）
自動作曲装置として用いられるコンピュータに、
入力されたフレーズに含まれる各ノートデータのピッチを順次ピッチシフトさせるステップと、
前記ピッチシフトが行われるごとに、連続するノートタイプの接続関係を規定する複数種のノート接続ルールを参照しながら、指定されたコード進行データと前記ピッチシフトされたフレーズとの適合度を算出するステップと、
前記算出された適合度に基づいて選択される前記ピッチシフトされたフレーズに基づいてメロディを生成するステップと、
を実行させるプログラム。
【符号の説明】
【０２５５】
１００　自動作曲装置
１０１　モチーフ入力部
　１０１－１　鍵盤入力部
　１０１－２　音声入力部
　１０１－３　音符入力部
１０２　モチーフ／コード進行適正評価部
１０３　伴奏・コード進行ＤＢ
１０４　ルールＤＢ
１０５　メロディ生成部
１０６　フレーズセットＤＢ
１０７　出力部

１０７－１　楽譜表示部
１０７－２　楽音生成部
１０８　入力モチーフ
１０９　コード進行候補
１１０　メロディ
１４０１　ＣＰＵ
１４０２　ＲＯＭ
１４０３　ＲＡＭ
１４０４　入力手段
１４０５　表示手段
１４０６　音源部
１４０７　サウンドシステム

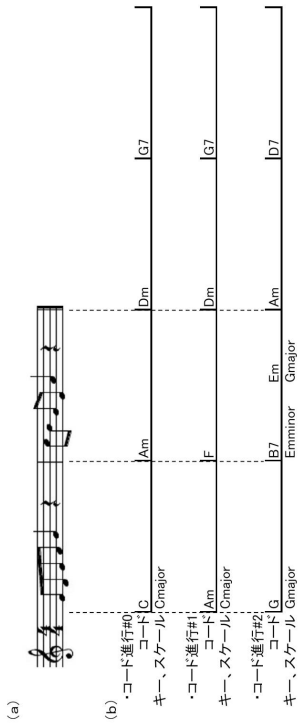
【図１】



【図２】



【図３】



【図４】

(a)

ノートデータ#0
ノートデータ#1
・
・
・
終端

(b)

時間
長さ
強さ
ピッチ

【図５】

(a)

コード進行データ／伴奏用MIDIデータ／曲構造データ #0
コード進行データ／伴奏用MIDIデータ／曲構造データ #1
・
・
・
終端

(b)

コードデータ#0
コードデータ#1
・
・
・
終端

(c)

時間
キー
スケール

(d)

時間
ルート
タイプ

【図６】

Measure	PartName[M]	iPartID[M]	ExistMelody[M]	iPartTime[M]
0	Null	0	0	
1	Intro	1	0	
2	Intro	1	0	
3	A	11	1	
4	A	11	1	
5	A	11	1	
6	A	11	1	
7	A	11	1	
8	A	11	1	
9	A	11	1	
10	A	11	1	
11	B	12	1	
12	B	12	1	
13	B	12	1	
14	B	12	1	
15	B	12	1	
16	B	12	1	
17	B	12	1	
18	B	12	1	
19	C	13	1	
20	C	13	1	
21	C	13	1	
22	C	13	1	
23	C	13	1	
24	C	13	1	
25	C	13	1	
26	C	13	1	
27	C	13	1	
28	A	11	1	
29	A	11	1	
30	A	11	1	
31	A	11	1	
32	A	11	1	
33	A	11	1	
34	A	11	1	
35	Ending	3	0	

【図７】

(a)

11	10	9	8	7	6	5	4	3	2	1	0	音階音（ビット位置）
0	0	0	0	1	0	0	1	0	0	0	1	MAJ
0	0	0	0	1	0	0	0	1	0	0	1	MIN
0	1	0	0	1	0	0	1	0	0	0	1	7TH
1	0	0	0	1	0	0	1	0	0	0	1	M7

(b)

11	10	9	8	7	6	5	4	3	2	1	0	音階音（ビット位置）
0	0	1	0	0	1	0	0	0	1	0	0	MAJ
0	0	0	0	0	0	1	0	0	1	0	0	MIN
0	0	1	1	0	0	0	0	1	1	1	0	7TH
0	0	1	0	0	1	0	0	0	1	0	0	M7

(c)

11	10	9	8	7	6	5	4	3	2	1	0	音階音（ビット位置）
1	0	1	0	1	0	1	1	0	1	0	1	ダイアトニック
0	1	1	0	1	0	1	0	1	1	0	1	ドリアン

【図８】

(a)ノートタイプ

	ノートのピッチ	E4		D4		E4		F4		G4		C5		B4		G4		E4		A4				
	コード進行#0	C		A		C		S		C		C		A		A		C		C				
	コード進行#1	C		A		C		S		A		C		A		A		A		C				
	コード進行#2	A		C		A		V		C		A		C		S		S		A				

(b)隣接音程

	ノートのピッチ	E4		D4		E4		F4		G4		C5		B4		G4		E4		A4				
	隣接音程		-2		2		1		2		6		-4		-1		5		-3					

(c)ノートタイプと隣接音程の配列変数データincon[i]

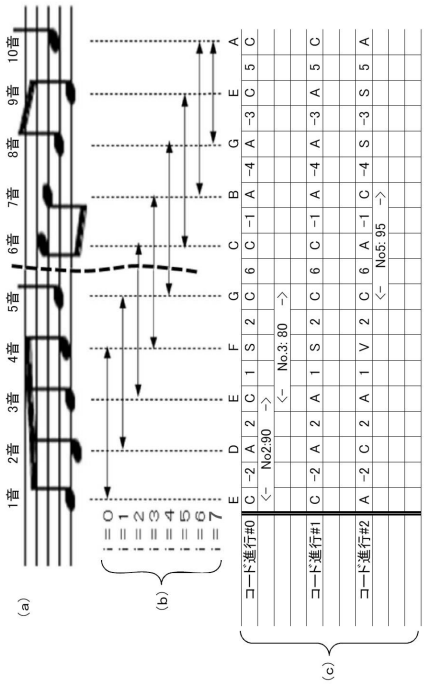
	コード進行#0	C	-2	A	2	C	1	S	2	C	1	S	2	C	6	C	-1	A	-4	A	-3	C	5	C
	コード進行#1	C	-2	A	2	A	1	S	2	A	1	S	2	A	6	C	-1	A	-4	A	-3	A	5	C
	コード進行#2	A	-2	C	2	A	1	V	2	C	6	A	-1	C	-4	S	-3	S	5	A				
	配列番号	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18				

【図９】

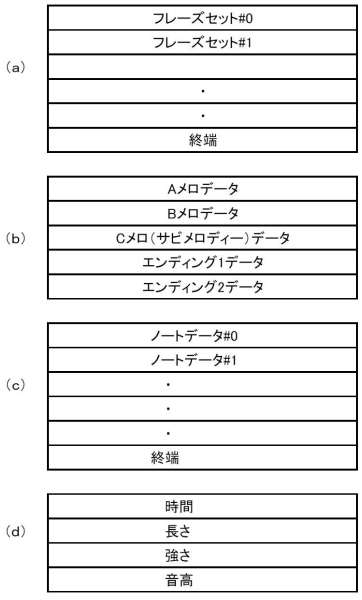
	j	ノートタイプ#0	隣接音程#0	ノートタイプ#1	隣接音程#1	ノートタイプ#2	隣接音程#2	ノートタイプ#3	評価点
3音	コードトーン 0	ci_ChordTone	99	ci_ChordTone	99	ci_ChordTone	0	ci_NullNoteType	100
	刺繍音 1	ci_ChordTone	-1	ci_AvailableNote	1	ci_ChordTone	0	ci_NullNoteType	100
	刺繍音 2	ci_ChordTone	-2	ci_AvailableNote	2	ci_ChordTone	0	ci_NullNoteType	90
	経過音 3	ci_ChordTone	1	ci_ScaleNote	2	ci_ChordTone	0	ci_NullNoteType	80
	経過音 4	ci_ChordTone	2	ci_AvailableNote	2	ci_ChordTone	0	ci_NullNoteType	100
	倚音 5	ci_ChordTone	99	ci_AvailableNote	-1	ci_ChordTone	0	ci_NullNoteType	95
	倚音 6	ci_ChordTone	99	ci_AvailableNote	-2	ci_ChordTone	0	ci_NullNoteType	95
	逆音 7	ci_ChordTone	1	ci_AvailableNote	99	ci_ChordTone	0	ci_NullNoteType	60
	逆音 8	ci_ChordTone	2	ci_AvailableNote	99	ci_ChordTone	0	ci_NullNoteType	80
4音	コードトーン 9	ci_ChordTone	99	ci_ChordTone	99	ci_ChordTone	99	ci_ChordTone	100
	刺繍音 10	ci_ChordTone	2	ci_AvailableNote	-3	ci_AvailableNote	1	ci_ChordTone	90
	刺繍音 11	ci_ChordTone	1	ci_AvailableNote	-3	ci_AvailableNote	2	ci_ChordTone	80
	経過音 12	ci_ChordTone	2	ci_AvailableNote	2	ci_AvailableNote	2	ci_ChordTone	95
	経過音 13	ci_ChordTone	2	ci_AvailableNote	1	ci_AvailableNote	2	ci_ChordTone	95
	経過音 14	ci_ChordTone	-2	ci_AvailableNote	-2	ci_AvailableNote	-2	ci_ChordTone	95
	経過音 15	ci_ChordTone	-2	ci_AvailableNote	-1	ci_AvailableNote	-2	ci_ChordTone	95
	倚音 16	ci_ChordTone	99	ci_AvailableNote	-3	ci_AvailableNote	1	ci_ChordTone	70
	倚音 17	ci_ChordTone	99	ci_AvailableNote	-3	ci_AvailableNote	2	ci_ChordTone	80

変数名 : ci_NoteConnect[j][0] ci_NoteConnect[j][1] ci_NoteConnect[j][2] ci_NoteConnect[j][3] ci_NoteConnect[j][4] ci_NoteConnect[j][5] ci_NoteConnect[j][6] ci_NoteConnect[j][7]

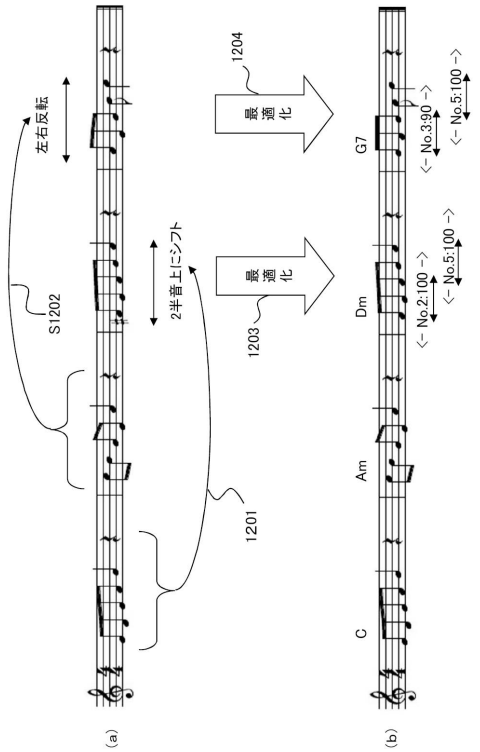
【図１０】



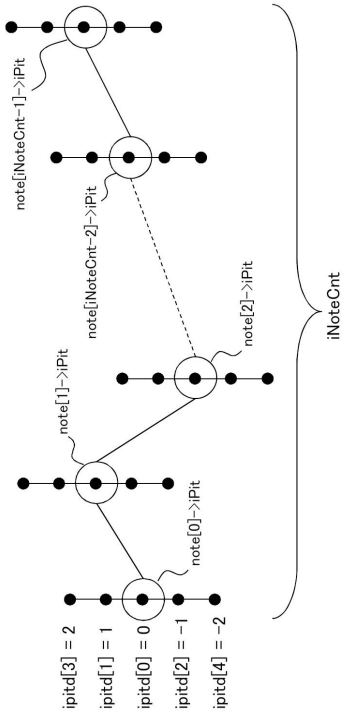
【図１１】



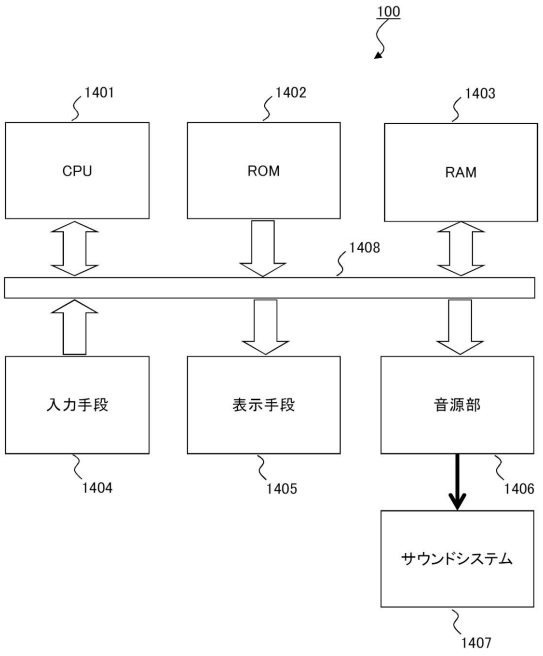
【図１２】



【図１３】



【図１４】



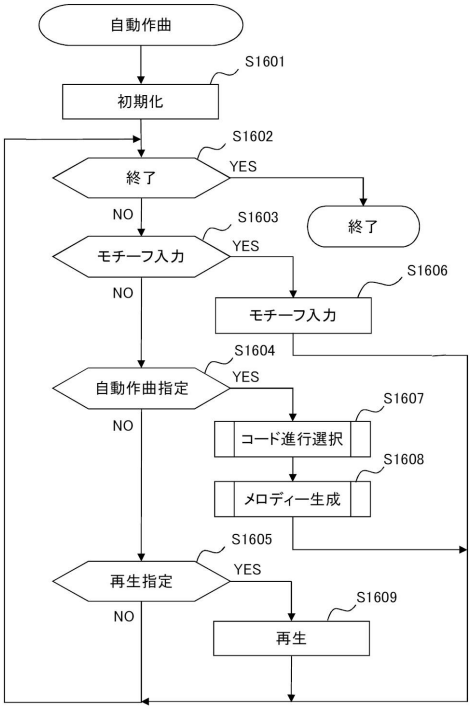
【図１５Ａ】

変数名	意味
n, i, j, k	繰返し処理制御用変数データ
MAX_CHORD_PROG	コード進行データの数を示す定数データ
iJunleSelect	楽曲ジャンル選択用変数データ
iChordAttribute[n][0]	コード進行#nの楽曲ジャンルを示す配列変数データ
iConceptSelect	楽曲コンセプト選択用変数データ
iChordAttribute[n][1]	コード進行#nの楽曲コンセプトを示す配列変数データ
iKeyShift	キーシフト値を示す変数データ
PITCH_CLASS_N	キーシフトの数を示す定数データ
doValue	適合度を示す変数データ
doMaxValue	適合度の最大値を示す変数データ
iBestUpdate	n番目の時点での最良コード進行を指す変数データ
iBestKeyShift[n]	n番目の時点での最良コード進行のキーシフト値
iBestChordProg[n]	n番目の時点での最良コード進行の番号
iCDesignCnt	コード進行内の情報番号を示す変数データ
cdesign[iCDesignCnt]	ICDesignCnt番目のコードデザインデータ
cdesign[iCDesignCnt]->iTime	コードデザインデータの時間情報
cdesign[iCDesignCnt]->iRoot	コードデザインデータのコード根音情報
cdesign[iCDesignCnt]->iType	コードデザインデータのコードタイプ情報
cdesign[iCDesignCnt]->iKey	コードデザインデータのキー情報
cdesign[iCDesignCnt]->iScale	コードデザインデータのスケール情報
mt	メタイベントを指すポインタ変数データ
root, type, scale, key	コード根音、コードタイプ、スケール、キーを示す変数データ
sTime	小節開始時間を示す変数データ
iNoteCnt	音列のノート番号を示す変数データ
me, me->iTime,	ノートを指すポインタ変数データ、その時間項目
notes[iNoteCnt]	ノートポインタ配列変数データ
iPit	ノートのピッチ項目値
ipit[i]	ピッチ情報列配列変数データ
incon[i×2], incon[i×2-1]	i番目のノートのノートタイプと隣接音程の配列変数データ
pcs1	コードトーンのピッチクラスセットを格納する変数データ
pcs2	テンションノートのピッチクラスセットを格納する変数データ
pcs3	スケールノートのピッチクラスセットを格納する変数データ
pc1, pc2	候補ピッチクラス1, 2を示す変数データ
ci_ChordTone	コードトーンを示す定数データ
ci_AvailableNote	アヴェイラブルノートを示す定数データ
ci_ScaleNote	スケールノートを示す定数データ
ci_TensionNote	テンションノートを示す定数データ
ci_AvoidNote	アヴォイドノートを示す定数データ
iTotalValue	総合評価点を示す変数データ
iValue	評価点を示す変数データ
iMaxValue	最大評価点を示す変数データ
ci_NoteConnect[j][k×2] ci_NoteConnect[j][k×2-1]	j番目のノート接続ルールのk番目の要素

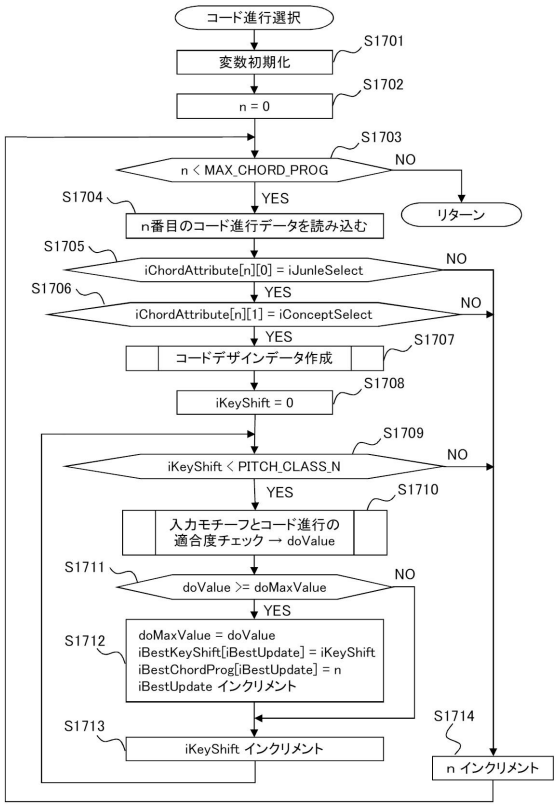
【図１５Ｂ】

変数名	意味
iMelodyA[0]～iMelodyA[iLengthA-1]	モチーフDB上のフレーズのピッチ列配列変数データ
iMelodyB[0]～iMelodyB[iLengthB-1]	入力モチーフのピッチ列
iLengthA	モチーフDB上のフレーズのピッチ列の長さ変数データ
iLengthB	入力モチーフのピッチ列の長さ変数データ
doDistance	距離評価値を示す変数データ
doMin	最小距離評価値を示す変数データ
iBestMochief	最良のフレーズセットを示す変数データ
MAX_NOTE_CANDIDATE	あるノートに対する別ピッチ候補の数
iWnum	音列のノート数すべての別ピッチ候補の数
ipitd[　]	或るノートに対する別ピッチ候補（差分）
ipitdev	ピッチ修正値を示す変数データ
iCnt	別ピッチ候補カウント用の変数データ
iMaxValue	最良適合度を示す変数データ
iMaxCnt	最良カウンタを示す変数データ

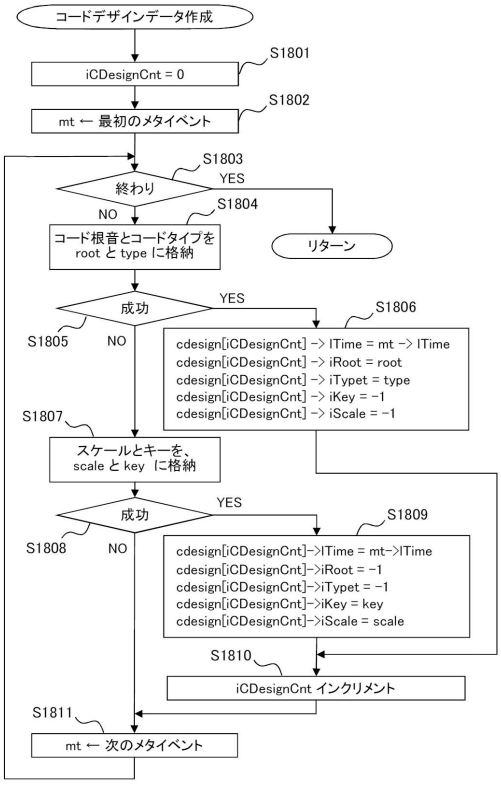
【図１６】



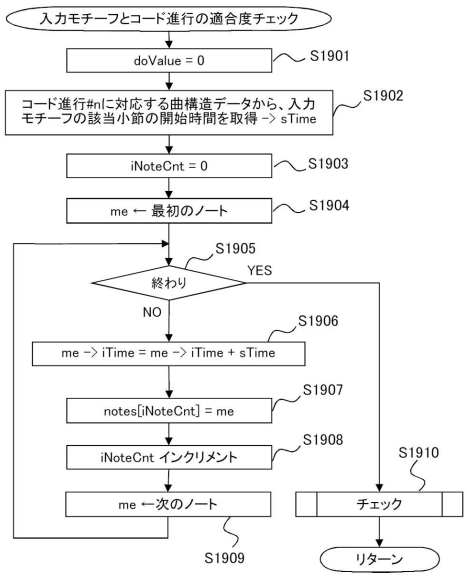
【図１７】



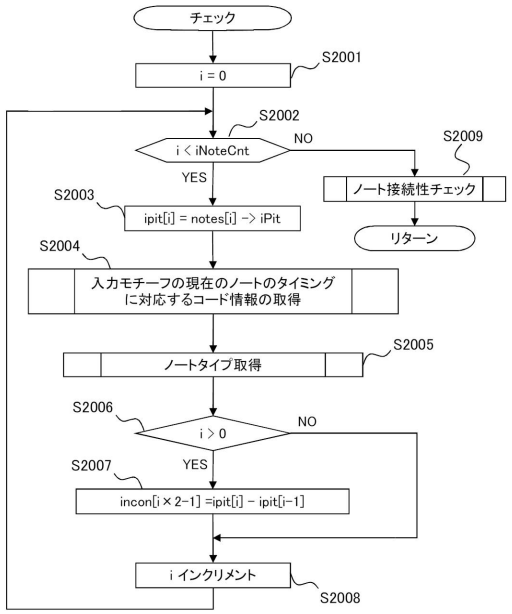
【図１８】



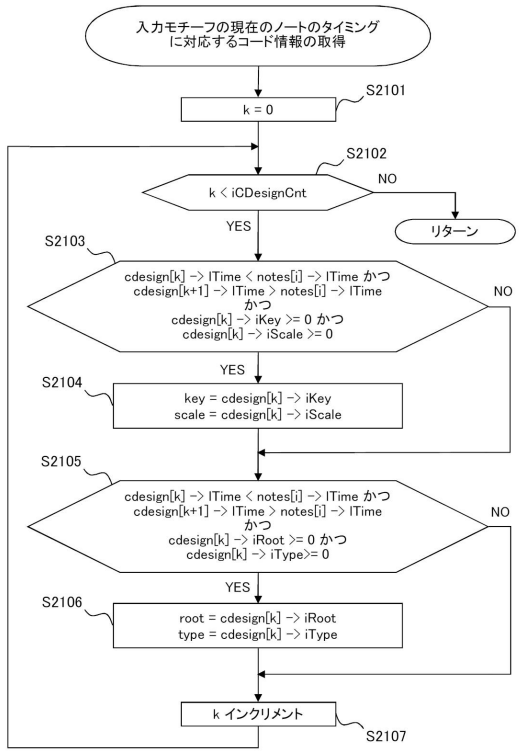
【図１９】



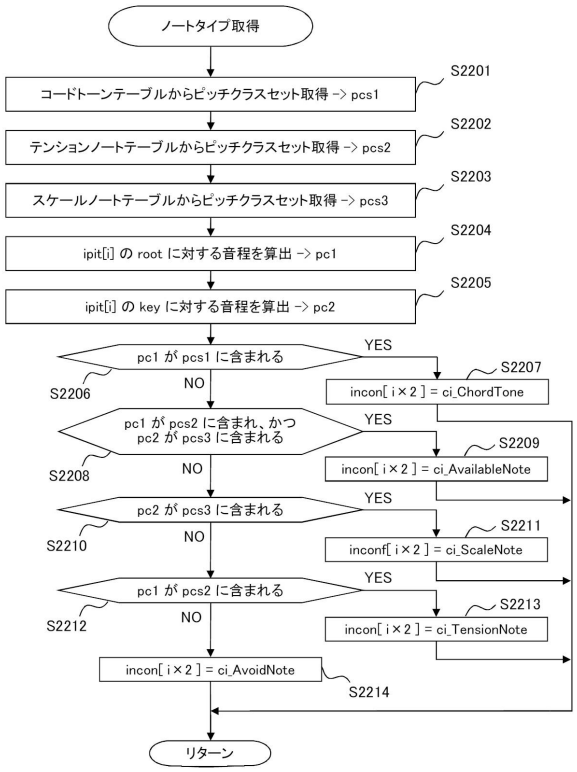
【図２０】



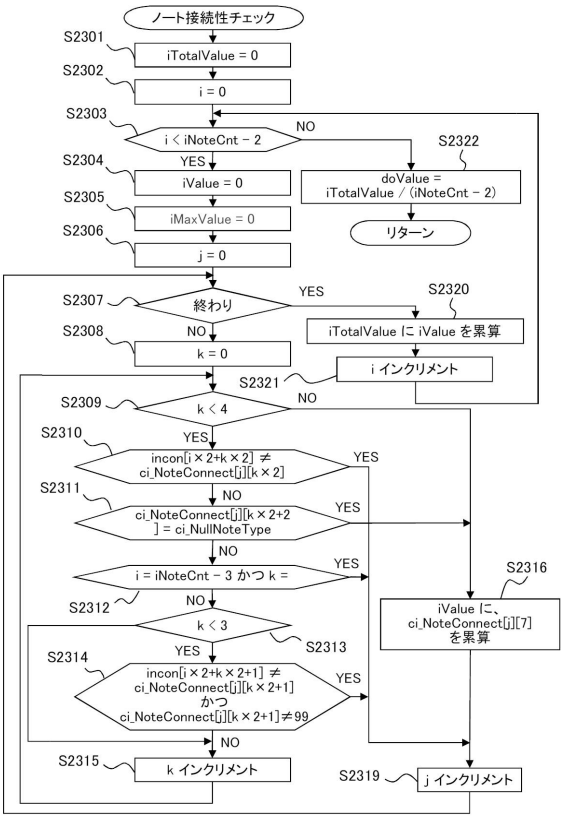
【図２１】



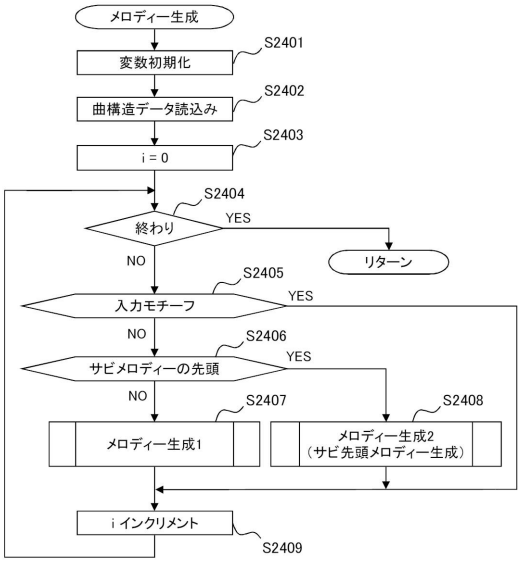
【図２２】



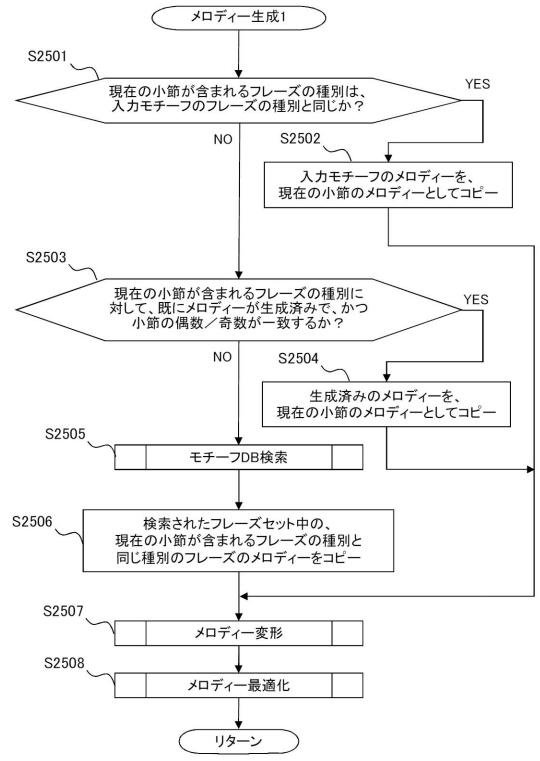
【図２３】



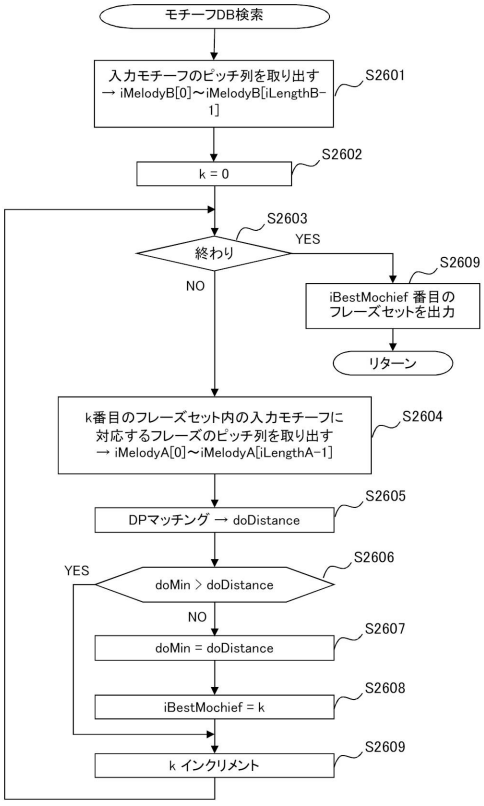
【図２４】



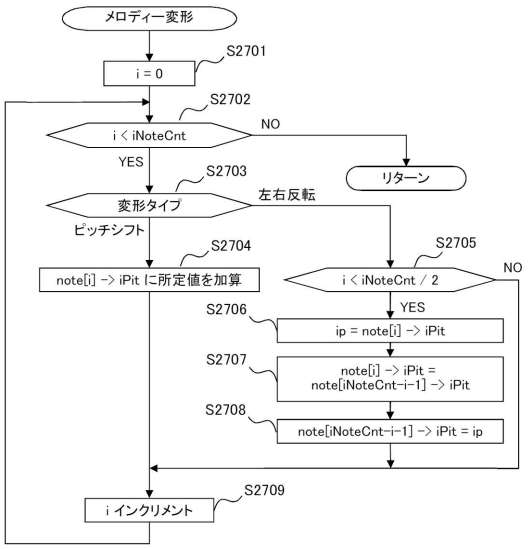
【図２５】



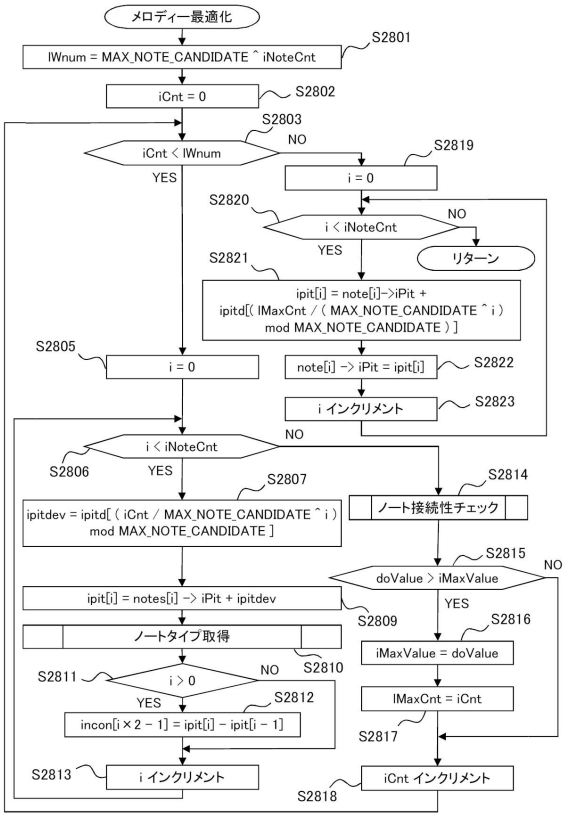
【図２６】



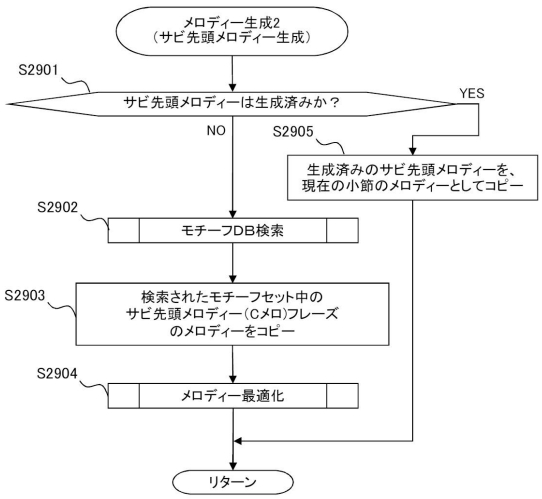
【図２７】


メロディー変形
i = 0
S2701
S2702
i < iNoteCnt
NO
YES
リターン
S2703
変形タイプ
左右反転
ピッチシフト
S2704
note[i] -> iPit に所定値を加算
S2705
i < iNoteCnt / 2
NO
YES
S2706
ip = note[i] -> iPit
S2707
note[i] -> iPit =
note[iNoteCnt-i-1] -> iPit
S2708
note[iNoteCnt-i-1] -> iPit = ip
S2709
i インクリメント

【図２８】


メロディー最適化
S2801
IWnum = MAX_NOTE_CANDIDATE ^ iNoteCnt
S2802
iCnt = 0
S2803
iCnt < IWnum
NO
YES
S2819
i = 0
S2820
i < iNoteCnt
NO
YES
リターン
S2821
ipit[i] = note[i]->iPit +
ipitd[(IMaxCnt / (MAX_NOTE_CANDIDATE ^ i)
mod MAX_NOTE_CANDIDATE)]
S2805
i = 0
note[i] -> iPit = ipit[i]
S2822
i インクリメント
S2823
NO
i < iNoteCnt
S2806
YES
S2807
ipitdev = ipitd[(iCnt / MAX_NOTE_CANDIDATE ^ i)
mod MAX_NOTE_CANDIDATE]
S2814
ノート接続性チェック
S2815
doValue > iMaxValue
NO
YES
ipit[i] = notes[i] -> iPit + ipitdev
S2809
S2816
ノートタイプ取得
S2810
iMaxValue = doValue
S2811
i > 0
NO
YES
S2812
incon[i × 2 - 1] = ipit[i] - ipit[i - 1]
IMaxCnt = iCnt
S2817
S2813
i インクリメント
iCnt インクリメント
S2818

【図２９】


メロディー生成2
（サビ先頭メロディー生成）
S2901
サビ先頭メロディーは生成済みか？
YES
NO
S2905
生成済みのサビ先頭メロディーを、
現在の小節のメロディーとしてコピー
S2902
モチーフDB検索
S2903
検索されたモチーフセット中の
サビ先頭メロディー（Cメロ）フレーズ
のメロディーをコピー
S2904
メロディー最適化
リターン

フロントページの続き

(56)参考文献 特開平１－１６７７８１（ＪＰ，Ａ）
特開２０００－１４８１４１（ＪＰ，Ａ）
特開２００２－３２０７７（ＪＰ，Ａ）
特開２００２－３２０７８（ＪＰ，Ａ）
特開２０１３－１０４８７８（ＪＰ，Ａ）
米国特許第５７５３８４３（ＵＳ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ１０Ｈ　　１／００－１／４６
Ｇ１０Ｇ　　１／００－３／０４