(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6265931号
(P6265931)
(45)発行日　平成30年1月24日(2018.1.24)　(24)登録日　平成30年1月5日(2018.1.5)

(51)Int.Cl.　F I
G05B 13/04 (2006.01)　G05B 13/04

請求項の数 19　(全 31 頁)

(21)出願番号　特願2015-36590(P2015-36590)
(22)出願日　平成27年2月26日(2015.2.26)
(65)公開番号　特開2015-170361(P2015-170361A)
(43)公開日　平成27年9月28日(2015.9.28)
審査請求日　平成28年9月28日(2016.9.28)
(31)優先権主張番号　14/199,459
(32)優先日　平成26年3月6日(2014.3.6)
(33)優先権主張国　米国(US)

(73)特許権者　000006013
三菱電機株式会社
東京都千代田区丸の内二丁目７番３号
(74)代理人　100110423
弁理士　曾我　道治
(74)代理人　100111648
弁理士　梶並　順
(74)代理人　100122437
弁理士　大宅　一宏
(74)代理人　100147566
弁理士　上田　俊一
(74)代理人　100161171
弁理士　吉田　潤一郎
(74)代理人　100161115
弁理士　飯野　智史

最終頁に続く

(54)【発明の名称】システムの連続モデル予測制御のための方法およびシステムを制御するための連続モデル予測コントローラー

(57)【特許請求の範囲】
【請求項１】
変化する係数行列Ａおよび右辺のベクトルｂを有する行列方程式Ａｘ＝ｂを繰り返し解くことによって制御の解ベクトルｘを求める反復解法を用いたシステムの連続モデル予測制御（ＣＭＰＣ）の方法であって、
前記反復解法の中で制御の現在の時間ステップにて制御で用いられる前記行列方程式の左辺の行列ベクトル積を返す正確な係数関数を近似する近似的な係数関数を求め、求めた近似的な係数関数を用いてプリコンディショナーの少なくとも一部分を求めるステップと、
前記プリコンディショナーを有する前記反復解法を用いて、正確な係数関数により定義された係数行列を有する前記ＣＭＰＣの行列方程式を解くことによって解ベクトルｘを求めるステップと、
前記解ベクトルｘを用いて、前記システムを制御するための制御信号を生成するステップと
を含み、
前記近似的な係数関数は、前記正確な係数関数を求めるための算術計算の精度よりも低い精度を有する算術計算を用いて求められた係数関数または微分方程式の単純化された解法を用いて求められ、
該方法の各ステップは、少なくとも１つのプロセッサによって実行される、システムの連続モデル予測制御のための方法。

【請求項２】
前記単純化された解法は、前記正確な係数関数を求めるための解法よりも高速であるが、該解法よりも正確でない
請求項１に記載の方法。
【請求項３】
前記システムのモデルを用いて、前記正確な係数関数を求めるステップと、
前記システムの前記モデルを近似するステップと、
前記システムの前記近似的なモデルを用いて、前記近似的な係数関数を求めるステップと
をさらに含む、請求項１に記載の方法。
【請求項４】
前記近似的な係数関数は、前記制御の異なる時間ステップについて求められた前記正確な係数関数である
請求項１に記載の方法。
【請求項５】
複数の近似的な係数関数を用いて、前記プリコンディショナーを求めるステップ
をさらに含む、請求項１に記載の方法。
【請求項６】
前記近似的な係数関数および前記正確な係数関数のうちの一方またはそれらの組み合わせを用いて、近似的な係数行列の各エントリーを求めるステップであって、前記行列方程式の左辺のベクトルｘに適用された前記近似的な係数関数は、前記近似的な係数行列と前記左辺のベクトルｘとの行列ベクトル積を返し、前記左辺のベクトルｘに適用された前記正確な係数関数は、正確な係数行列と前記左辺のベクトルｘとの行列ベクトル積を返す、各エントリーを求めるステップと、
前記近似的な係数行列を用いて、前記プリコンディショナーを求めるステップと
をさらに含む、請求項１に記載の方法。
【請求項７】
前記解くことは、メイン制御ルーチンの間にコントローラープロセッサによって実行され、前記方法は、
少なくとも１つの追加のプロセッサを用いて、前記メイン制御ルーチンと並列に前記近似的な係数行列および前記プリコンディショナーのうちの一方またはそれらの組み合わせを求めるステップ
をさらに含む、請求項６に記載の方法。
【請求項８】
前記近似的な係数行列は、前記制御の以前の時間ステップの間に求められ、前記方法は、
前記制御の前記現在の時間ステップの間に前記近似的な係数行列の少なくとも一部分を更新するステップ
をさらに含む、請求項６に記載の方法。
【請求項９】
前記近似的な係数関数および前記正確な係数関数のうちの一方またはそれらの組み合わせを、前記正確な係数行列のサイズの行列の全ての個々の列または列のブロックに適用し、前記近似的な係数行列を生成するための行列を生成し、前記行列を変換し、対称形の特徴を有する前記近似的な係数行列を生成するステップと
をさらに含む、請求項６に記載の方法。
【請求項１０】
前記システムのモデルに基づいて予測された次の時間ステップのためのデータを用いて、前記制御の前記現在の時間ステップの間に前記制御の前記次の時間ステップのための前記近似的な係数行列の少なくとも一部分を求めるステップ
をさらに含む、請求項６に記載の方法。

【請求項１１】
前記プリコンディショナーを求めるステップは、
反三角形因数分解を用いて、前記近似的な係数行列を反転するステップ
を含む、請求項６に記載の方法。
【請求項１２】
前記プリコンディショナーを求めるステップは、
前記近似的な係数行列の固有値分解を用いて、対称正定値プリコンディショナーとして前記プリコンディショナーを求めるステップ
を含む、請求項６に記載の方法。
【請求項１３】
前記プリコンディショナーを求めるステップは、
前記プリコンディショナーを対称正定値プリコンディショナーとして求めるステップ
を含み、前記反復解法は、前処理付き最小残差方法である
請求項１に記載の方法。
【請求項１４】
前記プリコンディショナーを求めるステップは、
前記反復解法の収束を遅くする誤差伝播演算子の固有ベクトルの少なくとも一部分を求めるステップと、
デフレーションを用いて、前記プリコンディショナーにおける前記固有ベクトルを除去するステップと
を含む、請求項１に記載の方法。
【請求項１５】
前記反復解法の収束速度を求めるステップと、
前記収束速度が閾値未満である場合、前記プリコンディショナーを少なくとも部分的に更新するステップと
をさらに含む、請求項１に記載の方法。
【請求項１６】
前記更新するステップは、
前記反復解法の反復ベクトルのうちの少なくともいくつかへの前記正確な係数関数の適用の結果を用いて、前記プリコンディショナーを少なくとも部分的に更新するステップ
を含む、請求項１５に記載の方法。
【請求項１７】
前記現在の時間ステップの前記反復解法の実行の間に前記プリコンディショナーを少なくとも部分的に更新するステップ
をさらに含み、前記反復解法は、前記プリコンディショナーが可変のプリコンディショナーであるように、前記プリコンディショナーの更新と同期して再開されるか、または柔軟な反復解法である
請求項１に記載の方法。
【請求項１８】
前記解ベクトルｘを求めるステップは、
前記制御の以前の時間ステップから前記反復解法の反復を再開することなく、前記制御の前記現在の時間ステップにおいて、前記反復解法における前記行列方程式を更新するステップ
を含み、前記反復解法は、前記行列方程式が可変の行列方程式であるように柔軟な反復解法である
請求項１に記載の方法。
【請求項１９】
変化する係数行列Ａおよび右辺のベクトルｂを有する行列方程式Ａｘ＝ｂを反復解法を用いて繰り返し解くことによって制御の解ベクトルｘを求めて、各時間ステップにおいて生成される制御信号に従ってシステムを制御する連続モデル予測コントローラー（ＣＭＰ

Ｃ）であって、
前記システムのモデル、前記行列方程式の左辺のベクトルｘに適用すると制御の現在の時間ステップにて制御で用いられる前記行列方程式の左辺の行列ベクトル積を返す正確な係数関数および前記正確な係数関数を近似する近似的な係数関数を記憶するメモリと、
前記近似的な係数関数を用いてプリコンディショナーを求め、前記プリコンディショナーを有する反復解法を用いて前記正確な係数関数によって定義された係数行列を有する行列方程式を解いて解ベクトルｘを生成し、前記解ベクトルｘを用いて前記制御信号を生成する少なくとも１つのプロセッサと
を備え、
前記近似的な係数関数は、前記正確な係数関数を求めるための算術計算の精度よりも低い精度を有する算術計算を用いて求められた係数関数または微分方程式の単純化された解法を用いて求められ、
各時間ステップにおいて生成される制御信号に従ってシステムを制御する連続モデル予測コントローラー。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、包括的には、モデル予測制御に関し、より詳細には、動的システムの前処理付き連続モデル予測制御のための方法および装置に関する。
【背景技術】
【０００２】
モデル予測制御（ＭＰＣ）は、多くの用途において複合動的システムを制御するのに用いられる。そのようなシステムの例には、生産ライン、車両エンジン、ロボット、他の数値制御された機械加工、および発電機が含まれる。
【０００３】
ＭＰＣは、システムのモデルの反復的な有限計画対象期間最適化に基づいている。ＭＰＣは、将来の事象を予期し、適切な制御動作を取る能力を有する。これは、制約を条件として将来の有限計画対象期間にわたって、システムの動作を最適化し、現在の時間ステップにわたる制御のみを実施することによって達成される。
【０００４】
ＭＰＣは、独立変数の変化によって引き起こされるモデル化されたシステムの従属変数の変化を予測することができる。化学プロセスでは、例えば、コントローラーが調整することができる独立変数は、多くの場合、圧力、流量、温度、弁の開度、およびダンパーの剛性である。コントローラーが調整することができない独立変数は、外乱として扱われる。これらのプロセスにおける従属変数は、制御目標またはプロセス制約のいずれかを表す他の測定値である。
【０００５】
ＭＰＣは、システムのモデル、現在のシステム測定値、プロセスの現在の動的状態、ならびにプロセス変数の目標および制限を用いて、従属変数の将来の変化を計算する。これらの変化は、独立変数および従属変数の双方に対する制約を条件として、従属変数を目標の近くに保持するように計算される。ＭＰＣは、通常、実施される各独立変数の最初の変化のみを送出し、次の変化が必要とされるときに計算を繰り返す。
【０００６】
ＭＰＣ数値方法は、所与のシステム向けに特別に設計されることが多い種々の計算装置上で実施される。計算装置は、安価な固定小数点精度の組み込みコントローラーから高性能のマルチコアコンピュータープロセッサユニット（ＣＰＵ）、グラフィックス処理ユニット（ＧＰＵ）、フィールドプログラマブルゲートアレイ（ＦＰＧＡ）、または専用並列コンピュータークラスターにまで及ぶことができる。
【０００７】
モデル予測コントローラーは、プロセスの動的モデルに依拠し、最も多くは、線形経験

モデルに依拠しており、この場合、ＭＰＣは、線形である。非線形モデルを用いてシステムを記述する非線形ＭＰＣ（ＮＭＰＣ）が、多くの場合、線形ＭＰＣと比較して、より現実的であるが、計算がより難しくなる。線形ＭＰＣと同様に、ＮＭＰＣは、一般に凸ではない、有限予測計画対象期間に関する最適制御問題を解くことを要し、これは、数値解を取得するための難題を提起する。ＮＭＰＣ最適制御問題の数値解は、通常、ニュートン型最適化方式に基づいている。

【０００８】

正確なニュートン型最適化方式は、対応するヤコビ行列の解析式を必要とし、この解析式は、実際には稀にしか利用可能でなく、一般的には前進差分（ＦＤ）近似に置き換えられる。そのような近似的なニュートン型最適化方式は、制御のあらゆる時間ステップの間、元の非線形方程式のＦＤ近似を利用する。

【０００９】

ＮＭＰＣへの適用において、そのような近似的なニュートン型最適化は、非特許文献１に記載されている連続ＮＭＰＣ（ＣＮＭＰＣ）数値方法によって実行することができる。ＣＮＭＰＣを用いたオンライン計算における重要で反復的な最も計算集約的ステップのうちの１つが、行列方程式Ａｘ＝ｂの数値解法である。ここで、Ａは、係数行列であり、ｘは、解ベクトルであり、ｂは、ベクトルである。この解法は、制御の時間ステップごとに係数行列Ａおよびベクトルｂを変化させる。

【００１０】

ＣＮＭＰＣにおける係数行列Ａは、明示的には与えられず、行列ベクトル積を介してのみ利用可能である。具体的には、積Ａ＊Ｘを計算するのに、コンピューターコードが利用可能である。ここで、Ａは、係数行列であり、＊は、行列と行列との積であり、Ｘは、単一の列ベクトル、または行列に組み立てられるいくつかの列ベクトルである。この開示において、このコンピューターコードの動作は、「係数関数」の適用と呼ばれる。ベクトルに適用された係数関数は、係数行列とそのベクトルとの積を返す。内部的には、係数関数は、非特許文献１に記載されているように、例えば、複数の微分方程式を解くことによる状態方程式および共状態方程式の評価を必要とする。

【００１１】

係数行列Ａは、行列フォーマットではなく、係数関数を介してのみ利用可能であるので、行列エントリーへのアクセスを必要とする直接的な行列解法は、ＣＮＭＰＣでは用いられない。その代わり、係数行列Ａを有さず係数関数のみを有して動作することができる、いわゆる「行列なし（matrix-free）」反復方法が用いられる。ＣＮＭＰＣの行列方程式Ａｘ＝ｂを解くための１つの従来の行列なし反復方法は、一般化最小残差（ＧＭＲＥＳ）方法である。あらゆる時間ステップの間に非線形方程式のＦＤ近似にＧＭＲＥＳを用いることは、一般に、ＦＤＧＭＲＥＳと呼ばれ、不正確なニュートンの方法と解釈することができる。ＣＮＭＰＣに関して、ＦＤＧＭＲＥＳは、連続／ＧＭＲＥＳ（Ｃ／ＧＭＲＥＳ）という名称が付けられている。

【００１２】

ＦＤＧＭＲＥＳには、係数関数を伴う行列ベクトル乗算が必要とされる。計算の全体的なコストは、ＦＤＧＭＲＥＳステップの最大数に比例する。ＦＤＧＭＲＥＳの収束が遅い結果、制御のオンライン計算が低速化し、コントローラーにおけるリアルタイム計算において取得される制御品質が低下する。

【先行技術文献】

【非特許文献】

【００１３】

【非特許文献１】Ohtsuka, T.「A Continuation/GMRES Method for Fast Computation of Nonlinear Receding Horizon Control」（Automatica, Vol. 40, No. 4, pp. 563-574, Apr. 2004）

【発明の概要】

【発明が解決しようとする課題】

【００１４】
したがって、システムの連続モデル予測制御の効率性を高めることが所望されている。
【課題を解決するための手段】
【００１５】
モデル予測制御（ＭＰＣ）は、多くの用途において用いられる。非線形モデルを有する非線形ＭＰＣ（ＮＭＰＣ）制御システムは、多くの場合、線形ＭＰＣよりも現実的であるが、計算がより難しい。連続ＮＭＰＣ（ＣＮＭＰＣ）は、変化する係数行列Ａおよび右辺のベクトルｂを有する行列方程式Ａｘ＝ｂを繰り返し解くことによって制御の解ベクトルｘを求めるタイプのＮＭＰＣである。制御の時間ステップごとに、ＣＮＭＰＣは、解ベクトルｘから、システムを制御する制御動作を生成する。
【００１６】
コントローラーの計算リソースに対して潜在的に大きな行列サイズおよび係数行列Ａの個々のエントリーが容易に利用可能でないことに起因して、行列方程式Ａｘ＝ｂ用の直接的な行列因数分解に基づく解法は、実用的ではない。したがって、ＣＭＰＣは、例えば、プリコンディショナー（前処理器）を有する反復方法を用いて行列方程式を反復的に解く。
【００１７】
ＣＮＭＰＣの前処理付き反復方法は、係数行列Ａを伴う行列ベクトル乗算を必要とし、計算の全体的なコストは、最大反復回数に比例する。係数行列Ａの計算の複雑さ、その変化する性質、および行列ベクトル乗算の必要性に起因して、ＣＮＭＰＣは、従来、係数行列をエントリー単位で用いず、明示的に求めることさえない。
【００１８】
その代わり、ＣＭＰＣは、ベクトルｘに適用されたときに、係数行列Ａとベクトルｘとの積、すなわち、ａ（ｘ）＝Ａ＊ｘを返す係数関数ａ（）に依拠する。この係数関数は、制御の時間ステップごとに生成される。内部的には、係数関数の式の生成には、複数の微分方程式を解くことが必要とされる。
【００１９】
本発明の種々の実施の形態は、係数行列の計算の複雑さ、および係数関数の存在にもかかわらず、係数行列を明示的に求めることが有益である可能性がある、という認識に基づいている。これは、係数行列を用いて行列方程式を直接解くか、または前処理付き反復方法においてプリコンディショナーを求めて、反復回数を削減することができるからである。ＣＮＭＰＣの計算複雑さは、最大反復回数に比例するので、いくつかの実施の形態によるプリコンディショナーは、ＣＮＭＰＣの複雑さを低減する。
【００２０】
本発明の種々の実施の形態は、前処理のために、係数関数を用いて係数行列の各エントリーを求める必要があるが、係数関数は、正確である必要はなく、近似的な係数関数とすることができる、という別の認識に基づいている。近似的な形式であっても、係数行列の各エントリーを求めるのに、係数関数を用いることによって、係数行列全体のＣＭＰＣの枠組みは、維持され、プリコンディショナーを求めるのに用いられる係数行列の品質は、改善される。
【００２１】
正確な係数関数とは、本明細書において用いられるとき、制御の現在の時間ステップについて、制御の解ベクトルを求めるのにコントローラーによって用いられる係数関数である。それ以外のどの係数関数も、近似的である。例えば、近似的な係数関数は、制御の異なる時間ステップについて求められた異なる正確な係数関数、または正確な係数関数を求めるための算術計算の精度よりも低い精度を有する算術計算を用いて求められた係数関数、または微分方程式の単純化された解法を用いて求められた係数関数である。
【００２２】
本発明のいくつかの実施の形態は、近似的な係数関数および正確な係数関数のうちの一方またはそれらの組み合わせを用いて係数行列を求める。係数行列の個々のエントリーを

求める際のそのような柔軟性によって、時間、空間、および／または品質において、係数行列の異なるエントリーを求めることの分割が可能になる。これは、近似的な係数関数および正確な係数関数のうちの一方またはそれらの組み合わせの使用が、異なるプロセッサ上で、制御の異なる時間ステップにおいて、異なる正確さで、同じ係数行列の異なるエントリーを求めることを可能にするとともに、プリコンディショナーを求めるためのＣＮＭＰＣの枠組みを、依然として維持しているからである。

【００２３】

例えば、いくつかの実施の形態は、正確な係数関数を用いて係数行列の一部分を求め、近似的な係数関数を用いて係数行列の別の部分を求める。例えば、制御の以前の時間ステップの間に求められた係数行列、すなわち、制御の現在の時間ステップの観点から近似的な係数関数を用いて求められた係数行列は、現在の時間ステップにおいて、正確な係数関数または近似的な係数関数を用いて更新することができる。

【００２４】

例えば、１つの実施の形態は、係数行列から求められたプリコンディショナーをいくつかの後続の時間ステップに再利用する。この実施の形態は、プリコンディショナーの品質を監視し、この品質が低下したとき、係数行列およびプリコンディショナーを再計算することができる。例えば、この実施の形態は、前処理付き反復方法の収束速度が閾値未満であるとき、係数行列を更新または再計算することができる。

【００２５】

いくつかの実施の形態は、行列方程式用の反復解法の実行と非同期かつ並行に、例えば反復解法の実行と並列に、係数行列を求めることもできるし、更新することもできる、という認識にも基づいている。例えば、１つの実施の形態では、行列方程式を解くことは、メイン制御ルーチンの間にコントローラープロセッサによって実行され、係数行列の少なくとも一部分は、追加のプロセッサを用いて、メイン制御ルーチンと並列に求められる。

【００２６】

本発明のいくつかの実施の形態は、係数行列のサイズの単位行列の座標ベクトル、または係数行列のサイズのフルランク行列のベクトルに、係数関数を適用することによって、係数行列の少なくとも一部分を求める。それらの実施の形態は、複数のプロセッサ、例えば、ＧＰＵまたはＦＰＧＡを用いたマルチスレッド計算を利用することができる。なぜならば、ベクトルへの係数関数の１つの適用が別の適用から独立しているからである。したがって、それらの実施の形態は、係数行列の要素を並列に求めることができ、これによって、計算効率が高められる。

【００２７】

いくつかの実施の形態は、係数行列を用いてプリコンディショナーを求め、行列方程式を解く際に、このプリコンディショナーを用いる。例えば、反復解法は、ある反復ステップにおいて停止することができ、その後、新しく求められた、または更新されたプリコンディショナーとともに、解の現在の反復近似値を再開される方法の初期近似値として用いて、再開することができる。この場合、再開と再開との間の反復のプロセス中、プリコンディショナーは、固定される。

【００２８】

代替的に、異なる実施の形態は、反復解法を再開することなく、この解法におけるプリコンディショナーを更新し、これによって、あらゆる反復において変化することができる可変のプリコンディショナーが得られる。したがって、実施の形態は、可変の前処理を可能にする柔軟な反復方法を用いるように変更される。

【００２９】

加えてまたは代替的に、１つの実施の形態は、従来のＣ／ＧＭＲＥＳが、所与の時間ステップにおいて開始するとき、以前の時間ステップ（複数の時間ステップの場合もある）からの、おそらくは解の初期近似値から離れている情報を用いない、という認識に基づいている。この情報は、収束を加速することができるので、この実施の形態は、従来の枠組みと比較して、異なる枠組みを利用する。この枠組みでは、行列方程式Ａｘ＝ｂ用の反復

解法は、時間ステップ間で再開されず、逆に、更新された係数行列Ａ（更新された係数関数によって与えられる）および更新された右辺のベクトルｂのみを用いて実行を維持する。この実施の形態は、可変の前処理だけでなく、更新された係数行列Ａおよび右辺のベクトルｂに対しても、柔軟な反復方法を用いる。

【００３０】

いくつかの実施の形態は、係数関数が、方程式行列を解くために反復方法によって用いられ、その結果、あらゆる反復において、ベクトルｘおよびＡ＊ｘの新たな対が得られるという別の認識に基づいている。これらの対は、通常は、廃棄されるが、いくつかの実施の形態は、係数行列およびプリコンディショナーを更新するのに、その情報を用いる。

【００３１】

したがって、１つの実施の形態は、システムの連続モデル予測制御（ＣＭＰＣ）の方法を開示する。該方法は、近似的な係数関数を用いてプリコンディショナーの少なくとも一部分を求めることと、前記プリコンディショナーを有する反復方法を用いて、制御の現在の時間ステップにおける正確な係数関数によって定義された係数行列を有する前記ＣＭＰＣの行列方程式を解くことによって解ベクトルを求めることであって、ベクトルに適用される前記近似的な係数関数は、該ベクトルへの前記正確な係数関数の適用の結果を近似することと、前記解ベクトルを用いて、前記システムを制御するための制御信号を生成することと、を含み、該方法のステップは、少なくとも１つのプロセッサによって実行される。

【００３２】

別の実施の形態は、各時間ステップにおいて生成される制御信号に従ってシステムを制御する連続モデル予測コントローラー（ＣＭＰＣ）を開示する。該コントローラーは、前記システムのモデル、近似的な係数関数、および正確な係数関数を記憶するメモリであって、ベクトルに適用された前記近似的な係数関数は、近似的な係数行列と前記ベクトルとの積を返し、前記ベクトルに適用された前記正確な係数関数は、正確な係数行列と前記ベクトルとの積を返す、メモリと、前記近似的な係数関数を用いてプリコンディショナーを求め、前記プリコンディショナーを有する反復方法を用いて前記正確な係数関数によって定義された係数行列を有する行列方程式を解いて解ベクトルを生成し、前記解ベクトルを用いて前記制御信号を生成する少なくとも１つのプロセッサと、を備える。

【図面の簡単な説明】

【００３３】

【図１】本発明の実施の形態によるコントローラーおよびシステムのブロック図である。

【図２】本発明の実施の形態によるコントローラーのブロック図である。

【図３】本発明の実施の形態によるコントローラー上で実施される非線形ＭＰＣ方法のブロック図である。

【図４Ａ】本発明の１つの実施の形態によるシステムの連続モデル予測制御（ＣＭＰＣ）のための方法のブロック図である。

【図４Ｂ】本発明のいくつかの実施の形態による行列方程式用の前処理付き反復解法の一例を示す図である。

【図５】本発明のいくつかの実施の形態による非同期並行プリコンディショナーのセットアップのブロック図である。

【図６】本発明の１つの実施の形態に従って係数関数をベクトルに並行して適用することによって係数行列を求める本発明の１つの実施の形態の図である。

【図７】ベクトルのブロックに係数関数を適用する別の実施の形態の図である。

【図８】既に記憶されたベクトルのセットを完全な状態に完成させ、この完成したものに対して係数関数を評価する１つの実施の形態のブロック図である。

【図９】本発明の１つの実施の形態によるプリコンディショナーを求めるための反三角形分解／因数分解の図である。

【図１０】係数行列Ａの固有値分解を用いてプリコンディショナーを求める別の実施の形態の図である。

【図１１】固有値分解を用いてプリコンディショナー関数を変更する別の実施の形態の図である。
【発明を実施するための形態】
【００３４】
本発明のいくつかの実施の形態は、モデル予測制御（ＭＰＣ）を用いた、システムの動作を制御するためのシステムおよび方法またはシステムのモデルに従ったシステムを提供する。
【００３５】
図１は、ＭＰＣコントローラー１１０に接続された１つの例示したシステム１２０を示している。コントローラー１１０は、このシステムのモデル１０２に従ってプログラムされる。このモデルは、システム１２０の状態１２１および出力１０３の経時的な変化を、現在および以前の入力１１１ならびに以前の出力１０３の関数として表す一組の方程式とすることができる。このモデルは、システムの物理的限界および動作限界を表す制約１０４を含むことができる。動作中、コントローラーは、システムの所望の挙動を示すコマンド１０１を受信する。このコマンドは、例えば、動きコマンドとすることができる。コマンド１０１の受信に応答して、コントローラーは、システムの入力としての役割を果たす制御信号１１１を生成する。この入力に応答して、システムは、システムの出力１０３および状態１２１を更新する。
【００３６】
本明細書において言及するようなシステムは、場合によっては電圧、圧力、力等の物理量に関連付けられることがある、ある特定の操作入力信号（入力）によって制御され、場合によっては、電流、流量、速度、位置等の物理量に関連付けられることがあるいくつかの制御された出力信号（出力）を返す任意の機械またはデバイスとすることができる。出力値は、一部分は、システムの以前の出力値に関係し、一部分は、以前の入力値および現在の入力値に関係している。以前の入力と以前の出力との依存関係は、システムの状態にコード化される。システムの動作、例えば、システムの構成要素の動きは、ある特定の入力値の適用に続いて、システムによって生成される一連の出力値を含むことができる。
【００３７】
システムのモデルは、システム出力が現在の入力および以前の入力、ならびに以前の出力の関数として、経時的にどのように変化するのかを記述する一組の数学方程式を含むことができる。システムの状態は、システムのモデルおよび将来の入力とともに、システムの将来の動きを一意に規定することができる、一般に、時変の任意の情報セット、例えば、現在のおよび以前の入力および出力の適切なサブセットである。
【００３８】
システムは、出力、入力、および場合によっては、システムの状態の操作が認められる範囲を制限する物理制約および仕様制約を受けることがある。
【００３９】
コントローラーは、固定または可変の制御周期サンプリング間隔で、システム出力およびシステムの動きの所望の操作を受信し、これらの情報を用いて、システムを動作させるための入力、例えば、制御信号１１１を決定するハードウェア、またはプロセッサ、例えば、マイクロプロセッサにおいて実行されるソフトウェアプログラムで実施することができる。
【００４０】
図２は、システムの状態１２１および出力１０３が、コマンド１０１に従うようにシステムを作動させる本発明の実施の形態によるコントローラー１１０のブロック図を示している。コントローラー１１０は、モデル１０２およびシステムの動作に対する制約１０４を記憶するためのメモリ２０２に接続された単一のコンピュータープロセッサユニット（ＣＰＵ）、または複数のＣＰＵ２０１を備える。コントローラーは、システム出力および状態が、コマンド１０１に従うように、システムを作動させる。
【００４１】

いくつかの実施の形態は、システムの連続モデル予測制御（ＣＭＰＣ）のためのシステムおよび方法を提供する。ＣＭＰＣは、変化する係数行列Ａおよびベクトルｂを有するＣＭＰＣの行列方程式Ａｘ＝ｂを繰り返し解くことによって、制御の解ベクトルｘを求めるタイプのＭＰＣである。各時間ステップにおいて、ＣＭＰＣは、解ベクトルｘから、システムを制御する制御動作を生成する。数学的な詳細については、付録Ａを参照されたい。いくつかの実施の形態は、非線形モデルを用いてシステムを記述する非線形ＭＰＣ（ＮＭＰＣ）を用いる。それらの実施の形態では、ＣＭＰＣは、実際にはＣＮＭＰＣである。

【００４２】

コントローラーの計算リソースに対して潜在的に大きな行列サイズおよび行列Ａの個々のエントリーが容易に利用可能でないことに起因して、行列方程式Ａｘ＝ｂ用の従来の直接的な行列因数分解に基づく解法は、実用的でないと考えられる。したがって、ＣＭＰＣは、例えば、前処理付き一般化最小残差（ＧＭＲＥＳ）方法等の前処理付き反復方法を用いて行列方程式を反復的に解くことを含む。

【００４３】

図３は、本発明のいくつかの実施の形態によるシステムの連続モデル予測制御（ＣＭＰＣ）のための方法のブロック図を示している。本方法は、コントローラー１１０のコンピュータープロセッサ２０１によって、および／またはコントローラーのプロセッサと並行して動作する追加のプロセッサを用いることによって、実行することができる。この例では、制御の現在の時間ステップｔにおけるコントローラーは、測定された状態１２１によってアクティブ化される。システムのモデル１０２および制約１０４は、コントローラーのメモリ２０２に記憶されている。

【００４４】

図３の方法は、現在の状態１０３の測定された値と、制御の以前の時間ステップの計画対象期間にわたって求められた状態、制御、ラグランジュ乗数の値３１０とに基づいてシステムを制御するための制御信号１１１を生成するオンライン制御ステップを実行する。

【００４５】

例えば、本方法は、値１０３および値３１０を用いて微分方程式を解くことによって、システムのモデルに従って計画対象期間における予測された状態３２５を求め（３２０）、この予測された状態３２５から、カルーシューキューンータッカー（Ｋａｒｕｓｈ－Ｋｕｈｎ－Ｔｉｃｋｅｒ：ＫＫＴ）最適性必要条件に従って計画対象期間における予測された共状態３３５を求める（３３０）。

【００４６】

本方法は、モデル、ラグランジュ乗数、状態３２５および共状態３３５、ならびに値３１０を用いて、係数関数３４５を生成する（３４０）。本方法は、係数関数３４５を用いて、計画対象期間にわたる制御信号およびラグランジュ乗数の変化を含むベクトルｘの行列方程式Ａｘ＝ｂを解いて（３５０）、制御信号１１１の解ベクトル３５５を生成する。解３５０について、係数行列Ａは、係数関数３４０によって暗黙的に与えられる。解ベクトルが求められた後、本方法は、制御信号１１１を生成し（３６０）、計画対象期間にわたる状態、制御、およびラグランジュ乗数の値を更新する。更新された値は、制御の次の時間ステップにおいて、本方法によって用いられる。

【００４７】

いくつかの実施の形態では、本方法は、計画対象期間にわたる全時間間隔の制御を決定する（３６０）が、ＭＰＣによって規定されるような最も近い時間ステップの制御信号１１１しか出力しない。

【００４８】

係数行列Ａの計算の複雑さ、その変化する性質、および行列ベクトル乗算の必要性に起因して、ＣＭＰＣは、通常、係数行列Ａをエントリーのようには用いず、明示的に求めることさえない。その代わり、ＣＭＰＣは、係数関数３４５に依拠する。ベクトルｘに適用された係数関数ａ（ｘ）３４５は、係数行列Ａとベクトルｘとの積、すなわちａ（ｘ）＝Ａ＊ｘを返す。内部的には、係数関数３４５の式の生成３４０には、状態３２５について

微分方程式を解き、次いで、共状態３３５について微分方程式を解くことが必要とされる。数学的な詳細については、付録Ａを参照されたい。

【００４９】

本発明の様々な実施の形態は、行列方程式Ａｘ＝ｂを解く。ここで、係数行列Ａは、係数関数３４５によって定義される。コントローラーの計算リソースに対して潜在的に大きな行列サイズおよび行列Ａの個々のエントリーが容易に利用可能でないことに起因して、行列方程式Ａｘ＝ｂ用の従来の直接的な行列因数分解に基づく解法は、実用的でないと考えられる。したがって、本発明のいくつかの実施の形態は、プリコンディショナーを有する反復方法を用いてＣＭＰＣの行列方程式を解く。

【００５０】

本発明のいくつかの実施の形態は、係数行列の計算の複雑さおよび係数関数の存在にもかかわらず、係数行列自体を明示的に求めることが有益である可能性がある、という認識に基づいている。これは、係数行列を用いて行列方程式を直接解くか、または前処理付き反復方法においてプリコンディショナーを求めて、反復回数を削減することができるからである。ＣＭＰＣの計算コストは、最大反復回数に比例するので、係数関数に基づいて求められたプリコンディショナーの使用によって、ＣＭＰＣのコストを削減することができる。

【００５１】

本発明のいくつかの実施の形態は、前処理のために、係数関数を用いて係数行列の各エントリーを求める必要があるが、係数関数は、正確である必要はなく、近似的な係数関数とすることができる、という別の認識に基づいている。近似的な形式であっても、係数行列の各エントリーを求めるのに、係数関数を用いることによって、係数行列全体にわたるＣＭＰＣの枠組みは、維持され、プリコンディショナーを求めるのに用いられる係数行列の品質は、改善される。加えてまたは代替的に、近似的な係数関数は、係数行列のエントリーを明示的に求めることがなくても、プリコンディショナーを求めるのに用いることができる。

【００５２】

図４Ａは、本発明の１つの実施の形態によるシステムの連続モデル予測制御（ＣＭＰＣ）のための方法のブロック図を示している。この実施の形態は、近似的な係数関数４９０を用いてプリコンディショナー４００の少なくとも一部分を求め（４０１）、プリコンディショナーを有する反復方法を用い、制御の現在の時間ステップにおける正確な係数関数３４５によって定義された係数行列を有するＣＭＰＣの行列方程式を解くことによって、解ベクトル３５５を求める（４０２）。この実施の形態は、解ベクトル３５５を用いてシステムを制御するための制御信号１１１を生成する（４０３）。

【００５３】

正確な係数関数とは、本明細書において用いられるとき、制御の現在の時間ステップにおける制御の解ベクトルを求めるのに、コントローラーによって用いられる係数関数である。それ以外のどの係数関数も、近似的である。例えば、近似的な係数関数は、制御の異なる時間ステップについて求められた異なる正確な係数関数、または正確な係数関数を求めるための算術計算の精度よりも、低い精度を有する算術計算を用いて求められた係数関数、または微分方程式の単純化された解法を用いて求められた係数関数である。

【００５４】

特に、ベクトルに適用された近似的な係数関数は、そのベクトルへの正確な係数関数の適用の結果を近似する。具体的には、ベクトルに適用された近似的な係数関数は、近似的な係数行列とそのベクトルとの積を返し、そのベクトルに適用された正確な係数関数は、制御の現在の時間ステップの正確な係数行列とそのベクトルとの積を返す。

【００５５】

近似的な係数関数を用いて、プリコンディショナーの少なくとも一部分を求めることは、係数行列の個々のエントリーおよび／またはプリコンディショナーを求める際の柔軟性を提供し、空間、時間、および／または品質において、プリコンディショナーの種々の部

分を求めることの分割を可能にする。これは、近似的な係数関数および正確な係数関数のうちの一方またはそれらの組み合わせの使用が、異なるプロセッサ上で、制御の異なる時間ステップにおいて、異なる正確さで、同じ係数行列の異なるエントリーを求めることを可能にするとともに、プリコンディショナーを求めるためのＣＭＰＣの枠組みを依然として維持しているからである。

【００５６】

図４Ｂは、本発明の１つの実施の形態による正確な係数関数４１０によって定義された係数行列を用いて、ＣＭＰＣの行列方程式Ａｘ＝ｂを解くための前処理付き反復方法の一例を示している。この反復方法の一例は、前処理付き一般化最小残差（ＧＭＲＥＳ）方法である。

【００５７】

正確な係数関数４１０は、制御の現在の時間ステップごとに求められる。また、解ベクトルｘの初期近似が求められる。例えば、ゼロベクトルまたは制御の以前の時間ステップにおいて求められた解ベクトルは、いくつかの可能な選択肢である。

【００５８】

ＣＭＰＣの反復方法は、あらゆる反復において正確な係数関数ａ（　）を伴う行列ベクトル乗算４２０を必要とし、計算の全体的なコストは、最大反復回数に比例する。停止判定基準４３０は、現在の近似値ｘが所与の許容範囲を満たすか否か、または反復ステップの最大数k_{max}に達したか否かを調べる。停止判定基準が満たされている場合（４３５）、反復は、停止し、ベクトルｘの現在の近似値が、行列方程式Ａｘ＝ｂの近似解４８０として出力される。停止判定基準が満たされていない場合（４３６）、解の更新された近似値４４０が計算され、反復は、ステップ４２０へのサイクルを続け、反復インデックスｋが１つだけ増加される（４７０）。

【００５９】

いくつかの実施の形態は、近似的な係数関数４９０を用いて、プリコンディショナー４５５の少なくとも一部分を求める（４５０）。プリコンディショナーは、最初に生成することができ、その後、例えば、メイン反復サイクルのあらゆるステップに対して適用することができる。いくつかの実施の形態では、プリコンディショナーは、メイン反復サイクルの外部で生成された（４５０）前処理関数ｔ（）であり、ベクトルに適用されたこの前処理関数ｔ（ｘ）＝Ｔ＊ｘは、ベクトルｘに対するプリコンディショナーの行列形式Ｔの積Ｔ＊ｘを返す。

【００６０】

プリコンディショナーＴは、例えば、メイン反復サイクルのあらゆるステップの間に適用することができる（４６０）。プリコンディショナーの適用４６０は、例えば、プリコンディショナー関数ｔ（　）が、残差ｂ－Ａｘに作用する、すなわち、ｔ（ｂ－Ａｘ）＝Ｔ＊ｘを求めることを意味することができる。前処理の適用によって、反復あたりの計算コストを大幅に増加させることなく、反復方法の収束が加速され、したがって、コントローラーの全体的な性能改善がもたらされる。

【００６１】

近似的な係数行列Ａの近似的な逆行列が、プリコンディショナーの行列形式Ｔに用いられる場合、すなわち、Ｔ　A^{-1}である場合、前処理付き反復方法における反復回数は、根本的に削減される。所与の時間ステップにおけるプリコンディショナーＴが、係数行列の逆行列と一致する正確な場合、すなわち、$T=A^{-1}$である場合、前処理付き方法における反復回数は、１に削減される。しかしながら、この正確な場合は、計算的に高価である可能性があり、したがって、本発明のいくつかの実施の形態は、前処理更新またはセットアップ４５０および適用４６０を行う目的のために、近似的な係数行列および／または近似的な係数関数４９０が、それらの正確な対応するものと比較して、生成および適用がより容易で高速である可能性があるとともに、依然として高品質の前処理をもたらすことができる、という認識に基づいている。

【００６２】

例えば、１つの実施の形態は、正確な係数関数を求めるための算術計算の精度よりも低い精度を有する算術計算を用いて、近似的な係数関数を求める。この実施の形態は、プリコンディショナーＴを求める目的で係数行列Ａを作成および更新することは、完全な正確さを必要とせず、コンピューターベースのコントローラー上で削減された算術計算ビット数を用いて行うことができる、という認識に基づいている。加えてまたは代替的に、別の実施の形態は、既に計算された近似的な係数行列関数または近似的な係数行列から低い精度の算術計算でプリコンディショナーＴを求めるものである。

【００６３】

例えば、２倍の算術計算精度を有する従来のコンピューター処理ユニット（ＣＰＵ）と、単一の算術計算精度を有するＧＰＵとを有する混合タイプの計算環境では、制御の計算は、倍精度のＣＰＵ上で実行することができる一方、プリコンディショナーの更新またはセットアップ４５０は、以下で詳細に説明する近似的な係数行列Ａの並列計算を含めて、単一精度のＧＰＵ上で実行することができる。

【００６４】

加えてまたは代替的に、別の実施の形態は、微分方程式用の単純化された解法を用いて近似的な係数関数を求める。ここで、この単純化された解法は、正確な係数関数を求めるための解法よりも高速であるが、この解法よりも正確でない。例えば、ステップ３２０および３３０において微分方程式を解くための解法は、近似的な係数関数を求めるための解法の時間領域および／または空間領域において、場合によっては不均一な初期のグリッドと比較して、より大きなサイズを用いることができる。これは、オンライン計算の速度を増加させることができる一方、その結果削減された正確さは、プリコンディショナーの更新またはセットアップ４５０にのみ影響し、実際に計算される解４８０および制御信号１１１の正確さには影響しない。

【００６５】

加えてまたは代替的に、別の実施の形態は、システムの単純化された近似的なモデル１０２を用いて、プリコンディショナーの更新またはセットアップ４５０用の近似的な係数関数４９０を求める。この実施の形態は、正確な係数関数を求めるのに用いられるシステムのモデルを近似し、このシステムの近似的なモデルを用いて近似的な係数関数を求める。そのようなモデルの近似および／または単純化の例には、ＮＭＰＣの線形化、元の計画対象期間よりも短い期間の使用、および制約の単純化のうちの、１つまたはそれらの組み合わせが含まれるが、これらに限定されるものではない。

【００６６】

いくつかの実施の形態は、係数関数ａ（　）、したがって、係数行列Ａが、時間ステップごとに変化する可能性があるが、ＮＭＰＣにおける連続最適制御問題は、互いに類似している可能性があるので、この変化は、多くの場合、穏やかなものである、という別の認識に基づいている。

【００６７】

図５は、本発明のいくつかの実施の形態による、事前に求められたプリコンディショナーを更新するための方法のブロック図を示している。例えば、行列方程式を解く（５３０）のに用いられる係数関数は、正確な係数関数５５０である。しかしながら、制御の次の時間ステップでは、この正確な係数関数は、近似的な係数関数５４０となり、プリコンディショナーを更新する（５４０）のに用いることができる。いくつかの実施の形態では、更新５２０は、解く（５３０）こととは非同期に、例えば、並列に行われる。例えば、解くことが、プロセッサ５１０によって実行される場合、更新５２０は、１つのまたは複数の異なるプロセッサ５１５によって実行される。

【００６８】

いくつかの実施の形態では、近似的な係数関数５４０は、プリコンディショナーの一部分のみを更新する（５２０）。この更新は、制御の異なる時間ステップにおいて、繰り返し実行される。したがって、この実施の形態は、複数の近似的な係数関数を用いて、プリコンディショナーを求める。そのような更新は、プリコンディショナーを求めることを、

異なる時点において求められた異なる係数関数を用いることによって時間において分割することと、追加のプロセッサを用いてプリコンディショナーの同じ部分または異なる部分を更新することによって、空間において分割することとを可能にする。

【００６９】

いくつかの実施の形態は、近似的な係数関数および正確な係数関数のうちの一方またはそれらの組み合わせを用いて求められた係数行列に基づいて、プリコンディショナーを求める。例えば、係数行列が、プリコンディショナーをセットアップする時間ステップごとに正確に求められる場合、前処理付き反復方法における反復回数を１に削減することができる。係数行列が、プリコンディショナーのセットアップのためにのみ近似的であるとき、反復回数は、１よりも多くなる可能性があるが、それでも、前処理のない従来の場合と比較して、少なくすることができる。したがって、いくつかの実施の形態は、プリコンディショナーを求める目的で係数関数の正確なものの計算のコストと、前処理付き反復方法における反復回数の削減との間のトレードオフを提供する。

【００７０】

例えば、１つの実施の形態は、以前の時間ステップの正確な係数行列Ａから求められたプリコンディショナーを、いくつかの後続の時間ステップに再利用する。これらの後続の時間ステップにおいて、正確な係数行列Ａは、変化する場合があり、そのため、プリコンディショナーは、現在の正確な係数行列Ａの近似値から求められる。この実施の形態は、プリコンディショナーの品質を監視し、その品質が低下したとき、プリコンディショナーＴを再計算することができる。例えば、この実施の形態は、前処理付き反復方法の収束速度が閾値未満であるとき、プリコンディショナーＴを再計算することができる。

【００７１】

代替的に、１つの実施の形態は、現在の時間よりも前のいくつかの時間ステップにおいて、予測された将来の状態について計算された係数行列から求められたプリコンディショナーＴを、いくつかの後続の時間ステップに再利用する。この実施の形態は、ＭＰＣプロセスが、制御および予測される将来の状態を計画対象期間内に計算する、という認識に基づいている。ＭＰＣは、次の時間ステップの制御のみをシステムに出力し、他の全ての計算された量を廃棄するが、それらの量は、プリコンディショナーを求めるのに用いることができる。

【００７２】

他の実施の形態は、例えば、追加のＣＰＵまたはＧＰＵ上でのＡｘ＝ｂ用の反復解法の実行と非同期かつ同時に（すなわち、並列に）プリコンディショナーＴを求めることもできるし、更新することもできる、という認識に基づいている。

【００７３】

いくつかの実施の形態は、反復方法を停止して、プリコンディショナーを更新する。例えば、反復方法は、ある反復ステップにおいて停止することができ、その後、新しく再計算または更新されたプリコンディショナーとともに、解の現在の反復近似値を再開される方法の初期近似値として用いて再開することができる。この場合、再開と再開との間の反復のプロセス中、プリコンディショナーは、固定される。

【００７４】

代替的な実施の形態は、再開することなく、再計算または更新されたプリコンディショナーを反復解法において用い、これによって、可変のプリコンディショナー、すなわち、あらゆる反復において変化することができるプリコンディショナーが得られる。したがって、実施の形態は、柔軟な一般化最小残差（ＰＧＭＲＥＳ）方法、前処理付き最小残差（ＰＭＩＮＲＥＳ）方法、または前処理付き共役勾配（ＰＣＧ）方法等の、可変の前処理を可能にする柔軟な反復方法を用いるように変更される。

【００７５】

加えてまたは代替的に、１つの実施の形態は、通常、所与の時間ステップにおいて開始する反復方法は、以前の時間ステップ（複数の場合もある）からの、おそらくは、解の初期近似値から離れている情報を用いない、という認識に基づいている。この情報は、収束

を加速するのに利用価値がある場合があるので、この実施の形態は、時間ステップ間で反復方法を再開せず、逆に、更新された係数行列Ａ（更新された係数関数によって与えられる）および更新されたベクトルｂを用いて反復に実行を維持する。この実施の形態は、可変の前処理だけでなく、更新された係数行列Ａおよびベクトルｂに対しても、柔軟な反復方法を用いる。

【００７６】

本発明のいくつかの実施の形態は、プリコンディショナーを、コントローラーメモリに記憶されたシステム係数行列Ａのエントリーから直接求める。係数行列Ａのエントリーは、最初は明示的に利用可能でないので、いくつかの実施の形態は、近似的な係数行列または正確な係数行列の各エントリーを求める。例えば、１つの実施の形態は、近似的な係数関数および正確な係数関数のうちの、一方またはそれらの組み合わせを用いて、近似的な係数行列の各エントリーを求め、この近似的な係数行列を用いて、プリコンディショナーを求める。

【００７７】

図６は、係数関数ａ（　）６０１を座標ベクトルe_i、すなわち、係数行列Ａのサイズの単位行列Ｉの列または任意のフルランク行列Ｚの列z_iに適用することによって、係数行列を求める（６００）、本発明の１つの実施の形態のフローチャートを示している。係数関数６０１は、正確な係数関数または近似的な係数関数である。

【００７８】

例えば、いくつかの実施の形態は、係数関数６０１をベクトルz_iのそれぞれに並行して（６１０）適用して（６２０）、行列ＡＺ６３０を取得する。これらの実施の形態は、行列ＡＺに行列Ｚの逆行列を右側から乗算することによって、係数行列Ａを正確または近似的に求める（６４０）。いくつかの変形形態では、行列Ｚは、単位行列Ｚ＝Ｉである。それらの変形形態では、乗算６４０は、必要ではない。

【００７９】

係数行列Ａが適切に定式化された場合、この係数行列は、対称形である。したがって、いくつかの実施の形態は、この係数行列Ａの上三角部分または下三角部分のみを求める。それにもかかわらず、実施の形態のうちのいくつかでは、便宜上または速度のために、完全な係数行列Ａが計算される。

【００８０】

コントローラーのいくつかの実施態様では、計算された完全な係数行列Ａは、正確には対称形でない場合があり、これは、前処理および反復方法において、数値の誤りをもたらす可能性がある。したがって、１つの実施の形態は、係数行列をその対称平均（$A+A^T$）／２に置き換える（６５０）ことによって係数行列を正確に対称形にする。ループ６１０内の計算６２０は、互いに完全に独立しているので、この実施の形態は、複数のプロセッサ、例えば、ＧＰＵまたはＦＰＧＡを用いたマルチスレッド（並列）計算を利用することができる。したがって、この実施の形態は、係数行列の要素を並列に求めることができ、これによって、計算効率が高められる。

【００８１】

図７は、同じプロセッサ上で、単一のベクトルａ（ｚ）ではなく、ベクトルのブロックＺに係数関数ａ（Z_j）を適用することによって係数行列を求める（７００）本発明の１つの実施の形態のフローチャートを示している。係数関数６０１から係数行列Ａを生成するためにここで用いられる行列Ｚは、その列Z_jのブロックに事前に分割されている（７０１）必要がある。あらゆるブロックが、単一の列しか含まない場合、図７の実施の形態は、図６の実施の形態と同様である。そうでない場合、ステップ７１０、７２０、および７３０は、ステップ６１０、６２０、および６３０を一般化したものである。

【００８２】

同じプロセッサ上で係数関数をベクトルのブロックに適用することによって、１つの実施の形態は、高レベル基本線形代数サブプログラム（ＢＬＡＳ）、またはそれらの類似の代替のものを利用することが可能になる。ＢＬＡＳは、ベクトルおよび行列を用いた共通

の線形代数演算を実行する低レベルカーネルサブルーチンのセットである。高レベルＢＬＡＳ、すなわち、ＢＬＡＳレベル２およびＢＬＡＳレベル３は、特に、データおよび命令のパイプライン化、いくつかの機能ユニットを用いた内部マルチスレッド化、およびマルチレベル内部キャッシュメモリを有するベクトルプロセッサ上でＢＬＡＳを用いてプログラムの性能を改善するように意図されている。いくつかの実施の形態では、所与のコントローラーコンピューターハードウェアが、一方では複数のプロセッサを用いたマルチスレッド計算の機会と、他方ではベクトルプロセッサが複数のデータおよび命令を取り扱う能力とをバランスさせるように、行列Ｚのブロックへの分割を最適化することができる。

【００８３】

いくつかの実施の形態は、正確な係数関数が反復方法のあらゆるステップにおいて、メイン解決ルーティングの一部として種々のベクトルに適用され、その結果、あらゆる反復においてベクトルｘおよびＡ＊ｘの新たな対が得られる、という別の認識に基づいている。これらの対は、この反復方法によってそれ以上用いられないが、プリコンディショナーをセットアップする目的で近似的な係数行列Ａを再構成することを助けることができる正確な係数行列Ａについての情報を提供することができる。したがって、いくつかの実施の形態は、反復方法の反復ベクトルのうちの少なくともいくつかへの正確な係数関数の適用の結果を用いて、プリコンディショナーの少なくとも一部分を更新する。

【００８４】

図８は、１つの実施の形態による、ベクトルの所与のブロックZ_1およびAZ_1から係数行列Ａを更新する一例を示している。最初に、この実施の形態は、任意の方法で行列Z_1をフルランク行列Ｚに完成させ（８０１）、例えば、図７におけるステップ７０１と同様の方法で、この補空間を分割する。積AZ_1は、以前の反復において既に計算されているので、この実施の形態は、値ｊ＝２から開始してループ８１０を実行する。この方法の他のステップは、図７における実施の形態のステップと同様である。したがって、この実施の形態は、既に記憶されたベクトルのセットを完全な状態に完成させ、この完成したものに対してのみ、係数関数を評価して、係数行列Ａを再構成するのに十分な情報を取得する。

【００８５】

係数行列Ａのエントリーを入手することができることによって、行列係数関数ａ（　）の逆関数をシミュレートする前処理関数ｔ（　）を求めることが可能になる。図９および図１０は、係数行列の様々な行列因数分解および固有値分解または特異値分解に基づくプリコンディショナーＴの作成および更新を具体的に説明する例示の実施の形態の図を示している。

【００８６】

図９は、直交変換Ｑを用いるとともに、対称反三角形行列Ｋを選択することによって、行列Ａの対称性を利用する反三角形分解$A＝QKQ^T$の図を示している。ここで、Ｔは、転置演算である。近似的な係数行列Ａ９００は、事前に求められ、メモリに記憶されている。プリコンディショナーのセットアップ９１０は、反三角形因数分解$A＝QKQ^T$を求める。ここで、行列Ｑは、直交性を有し、行列Ｋは、対称反三角形である。プリコンディショナー関数の適用ｔ（ｂ）９２０は、次のステップを含む。行列Ｑの逆行列は、行列方程式の右辺ｂに$y＝Q^T b$として適用される（９３０）。行列方程式Ｋｚ＝ｙは、行列Ｋが反三角形であることを用いて、後退代入によって解かれる（９４０）。最後に、プリコンディショナーは、ｔ（ｂ）＝Ｑｚとして求められる（９５０）。

【００８７】

この反三角形分解に基づく前処理の結果、行列Ａが正定値でない場合には、正定値でない対称プリコンディショナー$T＝A^{-1}$が得られる。これは、適切な反復解法、例えば、前処理付きＧＭＲＥＳを選択する際に考慮に入れられる。

【００８８】

図１０は、図９の実施の形態に加えてまたはこれに代えて用いることができる別の実施の形態のブロック図を示している。この実施の形態では、プリコンディショナーのセット

アップ１０１０は、行列Ａ＝ＶΣＶ^{T}の固有値分解を求める。ここで、Ｖは、列固有ベクトルの直交行列であり、Σは、固有値から作成された対角行列であり、演算^{T}は、行列転置を表す。固有値分解は、ステップ１０３０、１０４０、および１０５０に示すように、Ｔ＝Ｖ｜Σ｜$^{-1}$Ｖ^{T}によって与えられる対称正定値絶対値プリコンディショナー（ＡＶＰ）を作成するのに用いられる（１０２０）。直接的な解法１０４０では、この実施の形態は、対角行列Σの対角エントリーの絶対値を反転する。

【００８９】

対称性定値ＡＶＰ　Ｔの利点は、前処理付きＭＩＮＲＥＳ反復方法の使用が、行列方程式Ａｘ＝ｂを解くことを可能にする、ということである。ＭＩＮＲＥＳは、前処理付きＧＭＲＥＳと比較して、より確実かつより高速に収束することでき、したがって、高速に変化するシステムを制御することが可能な高クロックレートを有する信頼できるコントローラーをもたらす。

【００９０】

図１１は、行列Ａの固有値分解または特異値分解を用いることによって前処理関数ｔ（　）を変更する（１１２０）別の実施の形態を示している。この実施の形態は、プリコンディショナーがいくつかの時間ステップについて更新されていない場合、その品質が低下し、その結果、Ａｘ＝ｂを解くための前処理付き反復方法の収束が遅くなる、という認識に基づいている。数学的には、そのような低速化は、行列Ａの条件数の増加に関係しており、プリコンディショナーを変更して、遅い収束の原因である行列Ａのいくつかの（閾値を用いる）特異ベクトルまたは固有ベクトルＹ　１１００をデフレートまたは強化することによって、取り除くことができる。

【００９１】

例えば、プリコンディショナーＴをＡＶＰ等の対称正定値とし、Ｙを、ｔ（Ａ）の絶対値固有値によるいくつかの最小値およびいくつかの最大値に対応する固有ベクトルである列を有する行列１１００を表すものとする。これらの固有ベクトルは、例えば、以前の時間ステップにおいて計算することができる。１つの実施の形態は、Ｙに含まれる固有ベクトルを検討から除外し、したがって、前処理付き反復解法の収束を加速するデフレーションを用いて、前処理関数ｔ（　）を新たな前処理関数ｔ$_{Y}$（　）に変更する（１１２０）。例えば、この実施の形態は、ｔ$_{Y}$（　）をｔ$_{Y}$（）＝Ｐ$_{Y}$ｔ（）として数学的に定義する（１１３０）ことができる。ここで、Ｐ$_{Y}$は、Ｔ$^{-1}$直交補空間における行列Ｙの範囲へのプロジェクター１１１０である。

【００９２】

加えてまたは代替的に、いくつかの実施の形態は、行列方程式Ａｘ＝ｂの反復解法の遅い収束が、小次元部分空間に由来する場合がある、という認識に基づいている。プリコンディショナー関数において、この部分空間を検出し、これをデフレートすることによって、反復解法の収束を加速させることができる。この部分空間は、誤差伝播演算子の固有ベクトルに関係し、以前の時間ステップにおいてまたは反復の過程において、計算することができる。正確な係数関数は、あらゆる時間ステップにおいて変化するので、１つの実施の形態は、ＣＭＰＣ行列方程式Ａｘ＝ｂを解くための反復解法の実行中およびこの実行と同時に、すなわち、並列に、デフレートした部分空間の少なくとも一部分を更新する。

【００９３】

本発明の上記の実施の形態は、数多くの方法のいずれかにおいて実現することができる。例えば、それらの実施の形態は、ハードウェア、ソフトウェアまたはその組合せを用いて実現することができる。ソフトウェアにおいて実現されるとき、そのソフトウェアコードは、単一のコンピューター内に設けられるにしても、複数のコンピューター間に分散されるにしても、任意の適切なプロセッサ、またはプロセッサの集合体において実行することができる。そのようなプロセッサは、集積回路として実現することができ、集積回路構成要素内に、１つまたは複数のプロセッサが含まれる。しかしながら、プロセッサは、任意の適切な構成の回路を用いて実現することができる。

【００９４】

また、本明細書において概説される種々の方法またはプロセスは、種々のオペレーティングシステムまたはプラットフォームのいずれか１つを利用する１つまたは複数のプロセッサ上で実行可能であるソフトウェアとして、コード化することができる。さらに、そのようなソフトウェアは、いくつかの適切なプログラミング言語および／またはプログラミングツール若しくはスクリプト記述ツールのいずれかを用いて書くことができ、フレームワークまたは仮想機械上で実行される実行可能機械語コードまたは中間コードとしてコンパイルすることもできる。通常、プログラムモジュールの機能は、種々の実施の形態において、望ましいように組合せることもできるし、分散させることもできる。

【００９５】

また、本発明の実施の形態は、方法として具現することができ、その一例が提供されてきた。その方法の一部として実行される動作は、任意の適切な方法において順序化することができる。したがって、例示的な実施の形態において、順次の動作として示される場合であっても、例示されるのとは異なる順序において動作が実行される実施の形態を構成することもでき、異なる順序は、いくつかの動作を同時に実行することを含むことができる。

【００９６】

本発明は、好ましい実施の形態の例として説明されてきたが、本発明の趣旨および範囲内で様々な他の適応および変更を行うことができることを理解されたい。したがって、添付の特許請求の範囲の目的は、本発明の真の趣旨および範囲に入る全ての変形および変更を包含することである。

【００９７】

付録Ａ

ここでは、ＣＮＭＰＣの数学的定式化の特定の例として、非特許文献１における問題を僅かに拡張したものを検討する。

【００９８】

【数１】

本発明における数式　　対　　非特許文献１における数式

$$\min_{u,p} J.$$
$$J = \phi(t+T, x(t+T), p) + \int_t^{t+T} L(t', x(t'), u(t'), p)\,dt'$$
$$s.t.$$
$$\dot{x} = f(t, x(t), u(t), p)$$
$$C(t, x(t), u(t), p) = 0$$
$$\psi(t+T, x(t+T), p) = 0$$

$$\min_{u,p} J.$$
$$J = \phi(t+T, x(t+T)) + \int_t^{t+T} L(t', x(t'), u(t'))\,dt'$$
$$s.t.$$
$$\dot{x} = f(t, x(t), u(t))$$
$$C(t, x(t), u(t)) = 0$$

【００９９】

ここで、ｘ＝ｘ（ｔ）は、計画対象期間にわたる最適制御問題のための初期状態としての役割を果たすシステムの状態のベクトルを表す。ベクトルｕ＝ｕ（ｔ）は、計画対象期間にわたる最適制御問題のための入力制御としての役割を果たす制御入力ベクトルである。スカラー関数Ｊは、最小にしたいシステム性能コストであり、終端コスト（総和における第１項）および後退計画対象期間にわたるコスト（総和における第２項）を含む。状態ベクトルの一次時間導関数（「ドット付き」のｘによって表される）の方程式は、ｘおよび／またはｕが非線形の場合がある、システム動的モデルである。制約ベクトル関数Ｃを有する方程式は、状態ｘおよび制御ｕの等式制約を記述している。終端制約関数ψによって記述される１つの余分の制約、およびパラメーターｐを追加する。計画対象期間の長さの値Ｔは、原則として、ｔに依存する場合もあるが、表記を単純にするために、ここではそのような可能性は考えないことにする。

【０１００】

上記最適制御問題の連続式は、以下のように、コンピューター解法に適するように離散化される。計画対象期間をサイズΔｔのＮ個の計画対象期間時間ステップに分割することによって、均一な計画対象期間時間グリッドを導入し、時間連続ベクトル関数ｘ（ｔ）およびｕ（ｔ）をグリッド点におけるそれらのインデックス付き値ｘ$_{i}$およびｕ$_{i}$に置き換える。このように、Ｎは、計画対象期間にわたる最適制御問題のための複数の人工時間ステップである。後退計画対象期間にわたるコストの積分を単純な求積ルールによって近似する。状態ベクトルの一次時間導関数を有限差分式によって近似する。したがって、離散化された最適制御問題のこれらの定義をつなぎ合わせると、以下のようになる。

【０１０１】

【数２】

本発明における数式　　対　　非特許文献１における数式

$$\min_{u_i,p} J.$$

$$J=\phi(t_N,x_N,p)+\sum_{i=0}^{N-1}L(t_i,x_i,u_i,p)\Delta t,s.t.$$

$$x_{i+1}=x_i+f(t_i,x_i,u_i,p)\Delta t \quad (i=0\rightarrow N-1)$$

$$C(t_i,x_i,u_i,p)=0 \quad (i=0\rightarrow N-1)$$

$$\psi(t_N,x_N,p)=0$$

$$\min_{u_i,p} J.$$

$$J=\phi(t_N,x_N)+\sum_{i=0}^{N-1}L(t_i,x_i,u_i)\Delta t,s.t.$$

$$x_{i+1}=x_i+f(t_i,x_i,u_i)\Delta t \quad (i=0\rightarrow N-1)$$

$$C(t_i,x_i,u_i)=0 \quad (i=0\rightarrow N-1)$$

【０１０２】

これまでのところ、計画対象期間におけるＮＭＰＣ最適制御問題のみが離散化されていることに留意されたい。後に、提示を単純にするために、時間ステップサイズΔｔを有する同じ均一なグリッドを用いてシステム時間ｔを離散化する。より一般的には、計画対象期間における離散化されたＮＭＰＣ最適制御問題は、システムの時間離散化とは異なる場合がある。

【０１０３】

共状態（非特許文献１では、ラグランジュ乗数とも呼ばれる）ベクトルを、ここでは、計画対象期間にわたる最適制御問題の初期共状態としての役割を果たすλによって表し、等式制約に関連付けられたラグランジュ乗数ベクトルをμによって表し、制御理論において定義されているようないわゆるハミルトン関数を導入する。

【０１０４】

【数３】

$$H(t,x,\lambda,u,\mu,p)=L(t,x,u,p)+\lambda^{T}f(t,x,u,p)+\mu^{T}C(t,x,u,p)$$

【０１０５】

関数ψによって与えられる終端制約を緩和するために、ラグランジュ乗数νを導入する。制御入力ｕ、ラグランジュ乗数μ、ラグランジュ乗数ν、およびパラメーターｐを全て１つのベクトルに組み合わせるベクトル関数Ｕ（ｔ）も導入する必要がある。

【０１０６】

離散化されたＮＭＰＣ最適制御問題は、ここでは、Ｆ（ｔ，ｘ，Ｕ）＝０によって表される「オイラーラグランジュの方程式」、および代替的に「ＫＫＴ条件」と呼ばれることが多い従来の一次必要最適性基準を用いることによって解かれる。ＫＫＴ条件は、例えば、求められていない乗数のラグランジュ方法を適用することによって導出することができる。数学的には、Ｆ（ｔ，ｘ，Ｕ）＝０であり、解Ｕ＝Ｕ（ｔ）が一意である場合、この解は、最適点であり、したがって、所与の状態ｘ＝ｘ（ｔ）についての最適制御問題の解となる。制御ベクトル入力ｕは、それよりも大きなベクトルＵの一部分であるので、最適解Ｕは、最適制御入力ｕを与える。

【０１０７】

ＫＫＴ条件を形式的に導出するために、以下の拡張評価関数を導入することができる。

【０１０８】

【数４】

$$J_{ex}(X,\Lambda)=\phi(t_N,x_N,p)+\sum_{i=0}^{N-1}L(t_i,x_i,u_i,p)\Delta t$$
$$+\sum_{i=0}^{N-1}\lambda_{i+1}^{T}(x_i-x_{i+1}+f(t_i,x_i,u_i,p)\Delta t)$$
$$+\sum_{i=0}^{N-1}(\mu_i\Delta t)^{T}C(t_i,x_i,u_i,p)+\nu^{T}\psi(t_N,x_N,p)$$

【０１０９】

ここで、Ｘ＝［ｘ$_i$　ｕ$_i$　ｐ］^{T}であり、Λ＝［λ$_i$　μ$_i$　ν］^{T}である。Ｃ（ｔ$_i$，ｘ$_i$，ｕ$_i$，ｐ）＝０のラグランジュ乗数は、幾分一貫していない方法で選ばれることに留意されたい。一方では、Δｔを含めるために、μ$_i$Δｔのセットをラグランジュ乗数として取るが、他方では、この場合、μ$_i$Δｔではなく、μ$_i$のみをラグランジュ乗数として考える。

【０１１０】

【数５】

$$\frac{\partial J_{ex}^{T}}{\partial X}(X,\Lambda)=0$$

および

【数６】

$$\frac{\partial J_{ex}^{T}}{\partial \Lambda}(X,\Lambda)=0$$

を計算することによってＫＫＴ条件を導出する。例えば、

【数７】

$$\frac{\partial J_{ex}^{T}}{\partial u_i}(X,\Lambda)=0$$

である、ｕ$_i$に関する導関数を取ると、以下の方程式が取得される。

【０１１１】

【数８】

$$\frac{\partial J_{ex}^{T}}{\partial u_i}(X,\Lambda)=\frac{\partial L}{\partial u_i}(t_i,x_i,u_i,p)\Delta t+\lambda_{i+1}^{T}\frac{\partial f}{\partial u_i}(t_i,x_i,u_i,p)\Delta t$$
$$+\mu_i^{T}C(t_i,x_i,u_i,p)\Delta t=0$$

【０１１２】

ハミルトン表記を用いると、以下の式となる。

【０１１３】

【数９】

$$\frac{\partial H}{\partial u_i}(t_i,x_i,u_i,p)\Delta t=0$$

【０１１４】

【数１０】

$$\frac{\partial J_{ex}^{T}}{\partial \mu_i}(X,\Lambda)=0$$

である、μ_iに関する導関数を取ると、以下の方程式が取得され、この方程式は、結局のところ乗数Δｔも有する。

【０１１５】

【数１１】

$$C(t_i, x_i, u_i, p)\Delta t = 0$$

【０１１６】

したがって、理論上、正確にするために、ＫＫＴ条件における

【数１２】

$$\frac{\partial H}{\partial u_i}$$

およびＣは、結局のところΔｔを有するべきであり、これによって、行列F_Uは対称形になる。

【０１１７】

Ｕは、共状態ベクトルλを含まず、したがって、λは、方程式Ｆ（ｔ，ｘ，Ｕ）＝０における未知数のうちの１つでない、すなわち、独立して削除される必要があることに留意されたい。次に、共状態ベクトルλの削除手順から開始して、シンボリック行列関数Ｆの作成について説明する。Ｆ（ｔ，ｘ，Ｕ）の独立変数のベクトルｘは、状態ベクトルを表す。現在の測定された状態ｘは、以下のアルゴリズムの初期ベクトルx_0としての役割を果たす。

【０１１８】

１．最初に、以下のように全てのx_iおよびλ_iを計算する。

【０１１９】

【数１３】

x_0 は、現在の測定された状態として与えられる

↓

$$x_{i+1} = x_i + f(t_i, x_i, u_i, p)\Delta t \quad (\text{Note: } i = 0 \to N-1)$$

↓

$$\lambda_N = \frac{\partial \phi^T}{\partial x}(t_N, x_N, p) + \frac{\partial \psi^T}{\partial x}(t_N, x_N, p) \cdot \nu$$

↓

$$\lambda_i = \lambda_{i+1} + \frac{\partial H^T}{\partial x}(t_i, x_i, \lambda_{i+1}, u_i, \mu_i, p)\Delta t \quad (\text{Note: } i = N-1 \to 1)$$

【０１２０】

２．次に、上記で計算されたx_iおよびλ_iを用いて、以下のようにＦ（ｔ，ｘ，Ｕ）を計算する。

【０１２１】

【数１４】

$$F(t,x,U)$$

$$= \begin{bmatrix} \vdots \\ \dfrac{\partial H^T}{\partial u}(t_i,x_i,\lambda_{i+1},u_i,\mu_i,p)\Delta t \\ \vdots \\ (\text{Note: } i=0 \to N-1) \end{bmatrix} \in \mathbb{R}^{(n_u \cdot N)\times 1}$$

$$\begin{bmatrix} \vdots \\ C(t_i,x_i,u_i,p)\Delta t \\ \vdots \\ (\text{Note: } i=0 \to N-1) \end{bmatrix} \in \mathbb{R}^{(n_C \cdot N)\times 1}$$

$$\psi(t_N,x_N,p)] \in \mathbb{R}^{n_\psi \times 1}$$

$$\left[\frac{\partial \phi_N}{\partial p}^T (t_N,x_N,p) + \frac{\partial \psi_N}{\partial p}^T (t_N,x_N,p)\cdot \nu + \sum_{i=0}^{N-1} \frac{\partial H_i}{\partial p}^T (t_i,x_i,\lambda_{i+1},u_i,\mu_i,p)\Delta t \right] \in \mathbb{R}^{n_p \times 1}$$

【０１２２】

上記式において、次元は、次のとおりである。

【０１２３】

【数１５】

$$x_i \text{ and } f(t_i,x_i,u_i,p) \in \mathbb{R}^{n_x \times 1}$$

$$u_i \in \mathbb{R}^{n_u \times 1}$$

$$C(t_i,x_i,u_i,p) \text{ and } \mu_i \in \mathbb{R}^{n_C \times 1}$$

$$\psi(t_N,x_N,p) \text{ and } \nu \in \mathbb{R}^{n_\psi \times 1}$$

$$p \in \mathbb{R}^{n_p \times 1}$$

$$F(t,x,U) \in \mathbb{R}^{n_U \times 1}$$

$$n_U = (n_u + n_C)N + n_\psi + n_p$$

【０１２４】

ここで、n_xはベクトルｘのサイズであり、n_uはベクトルｕのサイズであり、n_pはベクトルｐのサイズであり、n_Cは条件Ｃの数であり、n_ψは終端条件ψの数である。ベクトル

【数１６】

$$U \in \mathbb{R}^{n_U \times 1}$$

は、

【数１７】

$$U = [\ldots u_i^T \ldots, \ldots \mu_i^T \ldots, \nu^T, p^T]^T$$

である。スカラー関数Ｈは、

【数１８】

$$H = L(t_i,x_i,u_i,p) + \lambda_i^T f(t_i,x_i,u_i,p) + \mu_i^T C(t_i,x_i,u_i,p)$$

である。ここにおける全ての関数および変数は、一般に時間依存性を有する場合がある。

【０１２５】

この方程式は、コントローラーボード上でコンピュータープロセッサがＭＰＣの各時間ステップにおいてリアルタイムで数値的に解く必要がある。これは、ＮＭＰＣの実施の最も難しく、能力を必要とする部分である。ハミルトン関数Ｈおよび制約関数Ｃは、それらの変数に非線形に従属する場合があるので、方程式Ｆ（ｔ，ｘ，Ｕ）＝０は、ＮＭＰＣに対応して、一般に非線形である。

【０１２６】
各時間ステップにおける図１の非線形方程式系１１３　Ｆ（ｔ，ｘ，Ｕ）＝０は、ホモトピー連続の技法を用いて、近似的なニュートンの方法によって解かれる。この方法では、解Ｕ（ｔ）は、連続パラメーターを用いて微分方程式を積分することによってトレースされる。ここで、非特許文献１に従った表記を乱用して、時間ｔを、初期値０を有する連続パラメーターとみなすことにする。また、この付録の残りの部分では、変数の順序をＦ（ｔ，ｘ，Ｕ）からＦ（Ｕ，ｘ，ｔ）に入れ換えることにする。

【０１２７】
Ｆ（Ｕ（０），ｘ（０），０）＝０となるようなＵ（０）を選び、Ｆ（Ｕ（ｔ），ｘ（ｔ），ｔ）＝０が全く同様に満たされるように、時間に関するＵの導関数を求める。したがって、あるスカラーζに関して、以下の式が得られる。

【０１２８】
【数１９】

$$\dot{F}(U(t),x(t),t) = -\zeta F(U(t),x(t),t) \qquad (1)$$

【０１２９】
スカラーζを正のζ＞０に選ぶと、方程式（１）の零解Ｆ（Ｕ（ｔ），ｘ（ｔ），ｔ）＝０は、安定する。これによって、非零のＦ（Ｕ（０），ｘ（０），０）を用いることが可能になり、連続パラメーターに関して、方程式（１）を、その離散化を用いて近似することが可能になり、さらに、解Ｆ（Ｕ（ｔ），ｘ（ｔ），ｔ）がほぼ零になることを予想することが可能になる。

【０１３０】
方程式（１）は、ＣＮＭＰＣの主要な微分方程式である。数値的に解くために、方程式（１）は、以下の式によって近似される。

【０１３１】
【数２０】

$$D_h F(U,x,t:V,y,1) = (F(U+hV,x+hy,t+h) - F(U,x,t))/h = -\zeta F(U(t),x(t),t), \quad (2)$$

【０１３２】
ここで、
【数２１】

$$V = \dot{U}$$

および
【数２２】

$$y = \dot{x}$$

は、それらに対応して、「ドット付き」のＵおよびｘ（Ｕおよびｘの時間ｔに関する導関数）を表し、（１）における左辺
【数２３】

$$\dot{F}(U(t),x(t),t)$$

は、（２）においては、前進差分方向導関数の以下の定義を用いてD_hＦ（Ｕ，ｘ，ｔ：Ｖ，ｙ，１）によって近似される。

【０１３３】
【数２４】

$$F_U(U(t),x(t),t)W + F_x(U(t),x(t),t)w + F_t(U(t),x(t),t)s \approx D_h F(U,xw,t:W,w,s) =$$

$$(F(U+hW,x+hw,t+hs) - F(U,x,t))/h$$

【０１３４】
連鎖法から以下のようになる。

【０１３５】

【数２５】

$$\dot{F}(U,x,t) = \frac{\partial F}{\partial U}\dot{U} + \frac{\partial F}{\partial x}\dot{x} + \frac{\partial F}{\partial t}$$

$$= F_U\dot{U} + F_x\dot{x} + F_t$$

【０１３６】

$D_h F$（Ｕ，ｘ，ｔ：Ｖ，ｙ，１）の定義を方程式（２）に挿入すると、その結果、以下のように、時間に関するＵの導関数である、Ｖについての方程式系が得られる。

【０１３７】

【数２６】

$$F(U+hV,x+hy,t+h) - F(U,x+hy,t+h) = \qquad (3)$$

$$- \zeta h\, F(U(t),x(t),t) - F(U,x+hy,t+h) + F(U(t),x(t),t)$$

【０１３８】

サンプリング周期Δｔ＝ｈを導入し、（３）においてｘ（ｔ＋Δｔ）＝ｘ＋ｈｙと表すと、（３）から以下の式が得られる。

【０１３９】

【数２７】

$$F(U(t)+ \Delta t\, V,\ x(t+\Delta t),\ t+\Delta t) - F(U(t),\ x(t+\Delta t),\ t+\Delta t) = \qquad (4)$$

$$- \zeta\, \Delta t\, F(U(t),x(t),t) - F(U(t),\ x(t+ \Delta t),t+ \Delta t) + F(U(t),x(t),t)$$

【０１４０】

ここで、

【数２８】

$$y = \dot{x}(t) = f(t,x,u)$$

である。

【０１４１】

状態ｘ（ｔ）は、測定から取得することができ、

【数２９】

$$\dot{x}(t) = f(t,x,u)$$

は、既に取得された状態ｘ（ｔ）を用いるとともに、以前の時間ステップにおいて解明された制御ｕ（ｔ）を用いて計算することができる。同じ表記Δｔを用いるが、（４）では、値Δｔ＝ｈを、時間ステップサイズおよび計画対象期間時間ステップサイズと比較して異なって選ぶことができることに留意されたい。

【０１４２】

方程式（４）の左辺の表式は、以下の係数関数である。

【０１４３】

【数３０】

$$a(V) = F(U(t)+ \Delta t\, V,\ x(t+\Delta t),\ t+\Delta t) - F(U(t),\ x(t+\Delta t),\ t+\Delta t) \approx \Delta t\, F_U{}^{*}V \qquad (5)$$

【０１４４】

これは、解く必要がある行列方程式Ａｘ＝ｂにおける係数行列Ａ＝F_Uを暗黙的かつ近似的に求めている。方程式（４）の右辺の表式は、行列方程式Ａｘ＝ｂにおける右辺のベクトル、すなわち、以下の式である。

【０１４５】

【数３１】

$$b = -\zeta \Delta t\, F(U(t),x(t),t) - F(U(t), x(t+\Delta t),t+\Delta t) + F(U(t),x(t),t)$$

【０１４６】

解ベクトルｘは、制御の現在の時間ステップにおける正確な（５）の係数関数ａ（ｘ）＝Ａ＊ｘによって定義された係数行列Ａを有するＣＭＰＣの行列方程式Ａｘ＝ｂを解くことによって求められる。

【０１４７】

いくつかの実施の形態の場合、行列Ａ＝F_Uが対称形であるか否かは重要である。非特許文献１のＦの形式的定義では、ＫＫＴ条件は、上位２つのベクトル方程式として、以下のものを含む。

【０１４８】

【数３２】

$$\frac{\partial H}{\partial u}(t,\vec{x},\vec{\lambda},\vec{u},\vec{\mu},\vec{p}) = 0, \qquad (6)$$

$$C(t,\vec{x},\vec{u},\vec{p}) = 0,$$

【０１４９】

これは、技術的には、行列Ａ＝F_Uを非常に非対称なものにする可能性がある。Ａ＝F_Uを対称形にするために、ヘッセ行列について予想されるように、Ｆ（ｔ，ｘ，Ｕ）の定義において、以下のように、Δｔを乗算する必要がある。

【０１５０】

【数３３】

$$\frac{\partial H}{\partial u}(t,\vec{x},\vec{\lambda},\vec{u},\vec{\mu},\vec{p})\Delta t = 0,$$

$$C(t,\vec{x},\vec{u},\vec{p})\Delta t = 0,$$

【０１５１】

これは、上記Ｆの本発明者らの定義に見ることができる。Δｔによる乗算は、方程式（６）に対応する、行列Ａ＝F_Uのブロックの未処理のスケーリングと同等である。

【０１５２】

数学的に言うと、Ｖの方程式（４）は、Ｖに関して正確には線形でない場合があること、すなわち、（５）における係数関数ａ（Ｖ）によって暗黙的に求められた係数行列Ａは、実際にはＶに従属する場合があること、を認識することも重要である。特に、これは、プリコンディショナーのセットアップの目的でａ（　）からＡを明示的に計算する本発明者らのアルゴリズムが、結果的に非対称行列をもたらす場合がある理由を説明している。そのような従属および対称性からのずれは、自由に選ぶことができるサンプリング周期Δｔ＝ｈを用いると小さくなる。Δｔ＝ｈの選択の際の重要な制限は、キャンセル誤差が、コントローラープロセッサの不正確な算術計算に起因して係数関数ａ（　）の有限差分評価において上昇し始めるということに由来している。これは、導関数の有限差分近似を用いることの回避不能な副作用である。Δｔ＝ｈの値の推奨された下限は、例えば、倍精度（10^{-16}）算術計算において、10^{-8}とすることができるが、最適な値は、Ｆに依存する。

【０１５３】

いくつかの実施の形態について、行列Ａ＝F_U用のプリコンディショナーＴを導入する。プリコンディショナーＴを用いた前処理は、解く必要がある行列方程式Ａｘ－ｂ＝０に直接適用することができ、次のようになる。すなわち、Ｔ（Ａｘ－ｂ）＝０となる。代替的に、方程式（１）は、以下のように前処理することができる。

【０１５４】

【数３４】

$$T\dot{F}(U(t),x(t),t) = -\zeta F(U(t),x(t),t) \qquad (7)$$

【０１５５】

この結果、上記と同じ操作を繰り返した後、前処理付き行列方程式はＴＡｘ－ｃ＝０となる。ここで、ベクトルの第１項

【数３５】

$$c = -\zeta \Delta t\, F(U(t),x(t),t) - T\, F(U(t), x(t+\Delta t),t+\Delta t) + T\, F(U(t),x(t),t)$$

は、ベクトルＴｂの第１項と僅かに異なる。前処理に関係した本発明者らの全ての実施の形態は、方程式（１）または（７）のいずれにも等しく適用可能である。

【０１５６】

プリコンディショナーＴの少なくとも一部分は、近似的な係数関数をa_{approx}（ｘ）　T^{-1}ｘとして用いて（暗黙的に）求めることができる。ここで、ベクトルｘに適用される近似的な係数関数をa_{approx}（　）は、ベクトルｘへの正確な係数関数ａ（　）の適用の結果ａ（ｘ）を近似する、すなわち、a_{approx}（ｘ）　ａ（ｘ）である。例えば、近似的な係数関数a_{approx}（　）は、正確な係数関数ａ（　）と同じ式（５）によって求めることができるが、正確な係数関数を求めるための算術計算の精度よりも、低い精度を有する算術計算を用いて計算することができる。

【０１５７】

加えてまたは代替的に、近似的な係数関数a_{approx}（　）は、Ｆ（ｔ，ｘ，Ｕ）の値を計算するのに必要とされる微分方程式用の単純化された解法を用いて求めることができ、この単純化された解法は、正確な係数関数を求めるための解法よりも高速ではあるが、この解法よりも正確でない。例えば、Ｆ（ｔ，ｘ，Ｕ）の評価は、状態ｘを記述する微分方程式

【数３６】

$$\dot{x} = f(t, x(t), u(t), p)$$

用の解法を必要とする。Ｆ（ｔ，ｘ，Ｕ）の定義において用いられるこの解法は、以下のように記述される陽的オイラー法である。

【０１５８】

【数３７】

$$x_{i+1} = x_i + f(t_i, x_i, u_i, p)\Delta t \quad (\text{Note: } i = 0 \to N-1)$$

【０１５９】

この方法は、値x_1，．．．，x_Nを計算し、関数ｆ（t_i，x_i，u_i，ｐ）のＮ個の評価値を必要とする。近似的な係数関数a_{approx}（　）を求める単純化された解法の一例は、この場合も、陽的オイラー法とすることができるが、２倍の大きさのΔｔを有する陽的オイラー法である。すなわち、以下の式となる。

【０１６０】

【数３８】

$$y_{j+1} = y_j + f(t_j, y_j, u_j, p)2\Delta t \quad (\text{Note: } j = 0 \to N/2-1)$$

【０１６１】

ここで、Ｎは、簡単にするために偶数と仮定され、$y_0 = x_0$である。この単純化された解法は、関数ｆ（t_i，x_i，u_i，ｐ）のＮ／２個の評価値しか必要としない。すなわち、元の陽的オイラー法と比較して２倍も高速にすることができる。この解法は、必要とされる値の半分しか与えず、そのため、例えば、以下のものがさらに定義される。

【０１６２】

【数３９】

$$z_{2j} = y_j, \quad z_{2j-1} = (y_j + y_{j-1})/2, \quad (\text{Note: j} = 1 \rightarrow N/2 - 1)$$

【０１６３】

したがって、この単純化された解法は、値z_1，．．．，z_Nを計算する。これらの値は、値x_1，．．．，x_Nしか近似することができないが、値z_1，．．．，z_Nの計算は、コントローラー上で、値x_1，．．．，x_Nの計算と比較して２倍も高速にすることができる。

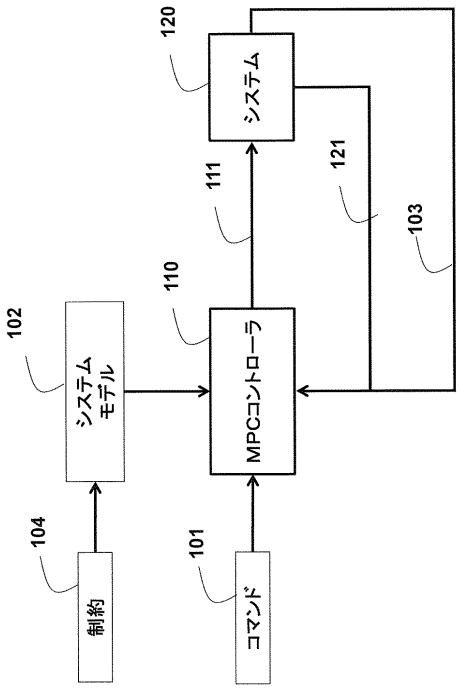
【０１６４】

別の例は、システムの近似的なモデルならびに制約関数Ｃおよび終端制約関数ψに基づいて、関数Ｆ（ｔ，ｘ，Ｕ）の近似を用いて求められた近似的な係数関数a_{approx}（　）である。ここで、関数ｆによって与えられるこのシステムの動的モデルは、近似することができる。このシステムの近似的なモデルは、正確な係数関数を求めるのに用いられる正確なモデルと比較して、コントローラー上での評価がより高速になるように求められるが、正確なモデルよりも正確ではない。例えば、関数ｆ、Ｃ、およびψのうちの１つまたはそれらの組み合わせは、短いテイラー展開によって求められたその近似と置き換えることができる。例えば、ｆ（ｘ，ｘ，ｕ，ｐ）＝ｅｘｐ（ｔ）とすると、ｆ＝ｅｘｐ（ｐ）は、小さなｔについて$f_{approx} = 1 + t + t^2/2$によって近似することができる。

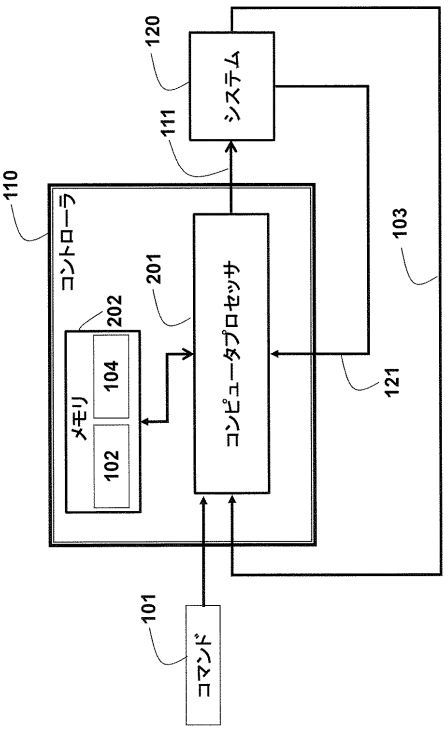
【０１６５】

加えてまたは代替的に、近似的な係数関数a_{approx}（　）は、制御の異なる時間ステップについて求められた正確な係数関数ａ（　）または制御のいくつかの異なる時間ステップについて求められた正確な係数関数ａ（　）の組み合わせとすることができる。過去の時間ステップの場合、正確な係数関数ａ（　）は、既に計算されており、コントローラーメモリにおいて利用可能な場合がある。制御の将来の時間ステップの場合、制御の現在の時間ステップの間、近似的な係数関数a_{approx}（　）を求めるのに必要とされる将来の時間ステップのためのデータ、例えば、状態ｘは、システムのモデルに基づいて予測することができる。制御のいくつかの異なる時間ステップについて求められた正確な係数関数ａ（　）の組み合わせは、例えば、それらの平均とすることができる。

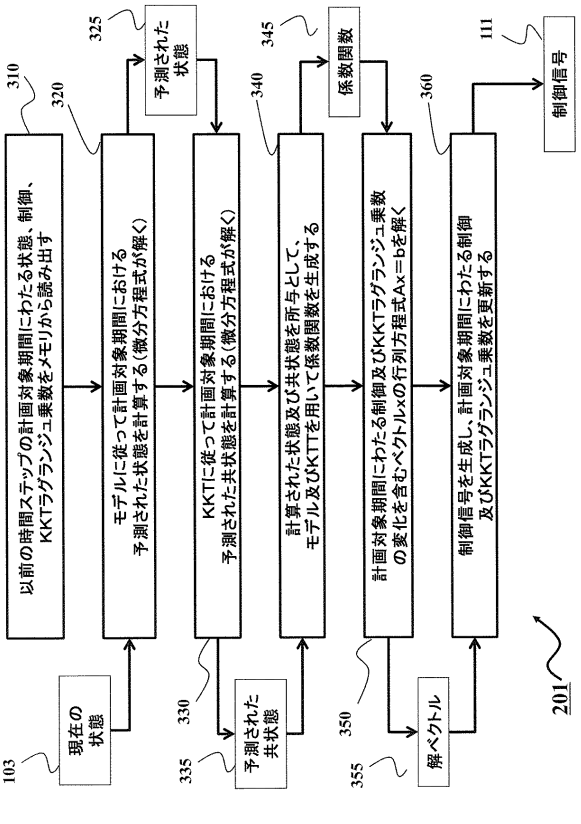
【図１】


システム
モデル
MPCコントローラ
システム
制約
コマンド
102
110
111
120
121
103
104
101

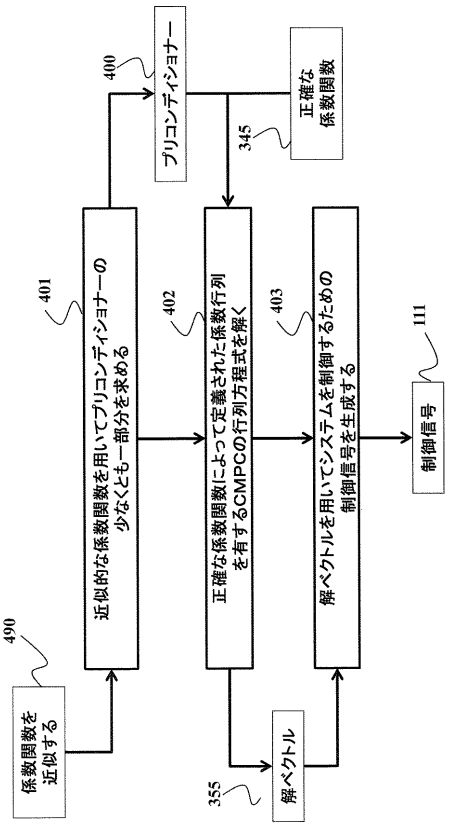
【図２】


コントローラ
メモリ
102
104
202
201
コンピュータプロセッサ
システム
コマンド
110
111
120
121
103
101

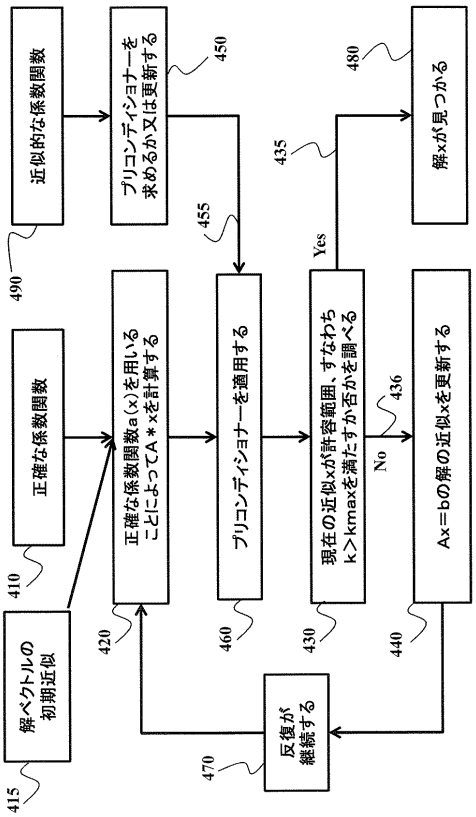
【図３】


以前の時間ステップの計画対象期間にわたる状態、制御、KKTラグランジュ乗数をメモリから読み出す
モデルに従って計画対象期間における予測された状態を計算する（微分方程式が解く）
KKTに従って計画対象期間における予測された状態を計算する（微分方程式が解く）
計算された状態及び共状態を所与として、モデル及びKTTを用いて係数関数を生成する
計画対象期間にわたる制御及びKKTラグランジュ乗数の変化を含むベクトルxの行列方程式Ax=bを解く
制御信号を生成し、計画対象期間にわたる制御及びKKTラグランジュ乗数を更新する
現在の状態
予測された状態
予測された状態
係数関数
解ベクトル
制御信号
310
320
325
330
335
340
345
350
355
360
111
103
201

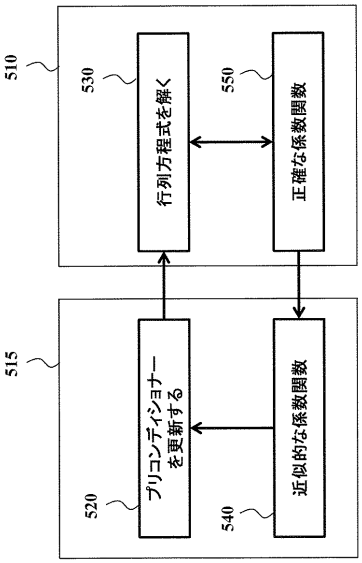
【図４Ａ】


係数関数を近似する
近似的な係数関数を用いてプリコンディショナーの少なくとも一部分を求める
プリコンディショナー
正確な係数関数によって定義された係数行列を有するCMPCの行列方程式を解く
正確な係数関数
解ベクトルを用いてシステムを制御するための制御信号を生成する
解ベクトル
制御信号
490
401
402
403
400
345
355
111

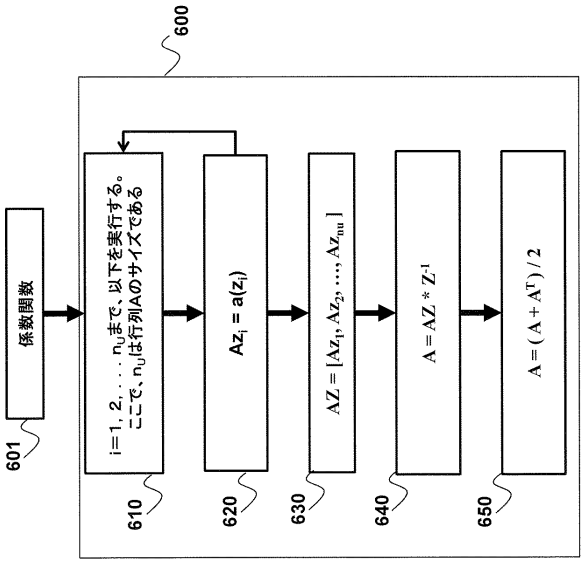
【図４Ｂ】


近似的な係数関数
490
プリコンディショナーを求めるか又は更新する
450
480
解xが見つかる
435
455
Yes
正確な係数関数
410
正確な係数関数a(x)を用いることによってA＊xを計算する
420
プリコンディショナーを適用する
460
現在の近似xが許容範囲、すなわちk>kmaxを満たすか否かを調べる
430
No
436
Ax＝bの解の近似xを更新する
440
解ベクトルの初期近似
415
反復が継続する
470

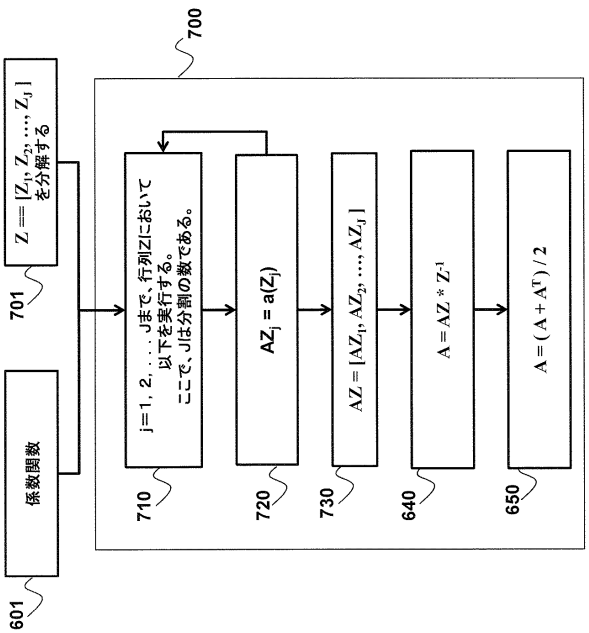
【図５】


510
530
行列方程式を解く
550
正確な係数関数
515
プリコンディショナーを更新する
520
540
近似的な係数関数

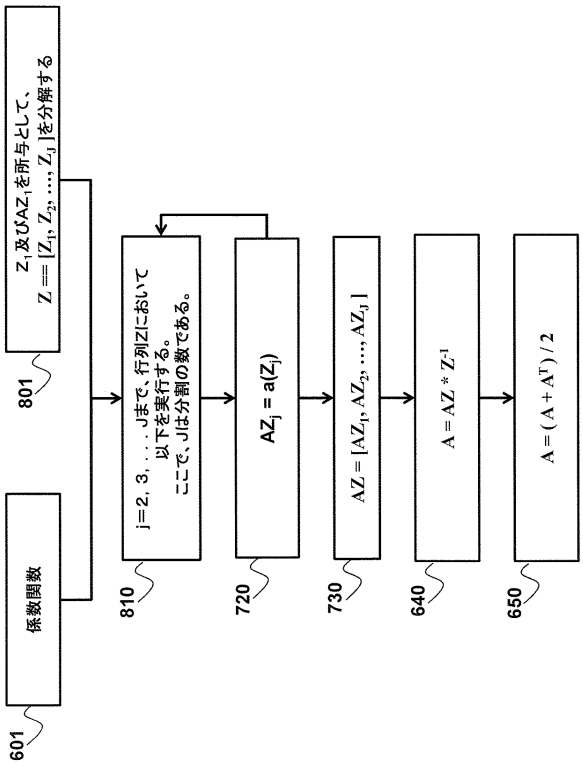
【図６】


600
係数関数
601
i=1, 2, . . . nUまで、以下を実行する。ここで、nUは行列Aのサイズである
610
Azi = a(zi)
620
AZ = [Az1, Az2, ..., AznU]
630
A = AZ * Z-1
640
A = (A + AT) / 2
650

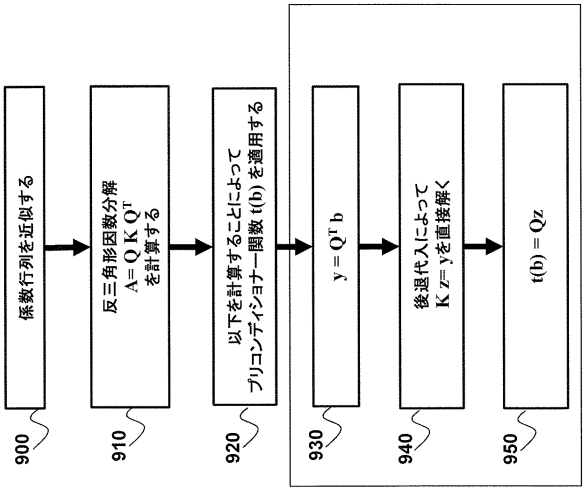
【図７】


700
Z == [Z1, Z2, ..., ZJ]を分解する
701
係数関数
601
j=1, 2, . . . Jまで、行列Zにおいて以下を実行する。ここで、Jは分割の数である。
710
AZj = a(Zj)
720
AZ = [AZ1, AZ2, ..., AZJ]
730
A = AZ * Z-1
640
A = (A + AT) / 2
650

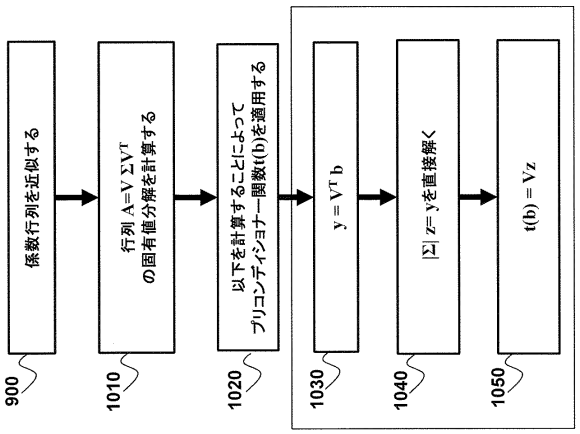
【図８】


係数関数
601
Z_1及びAZ_1を所与として、
Z = [Z_1, Z_2, ..., Z_J]を分解する
801
j=2, 3, . . . Jまで、行列Zにおいて
以下を実行する。
ここで、Jは分割の数である。
810
AZ_j = a(Z_j)
720
AZ = [AZ_1, AZ_2, ..., AZ_J]
730
A = AZ * Z^-1
640
A = (A + A^T) / 2
650

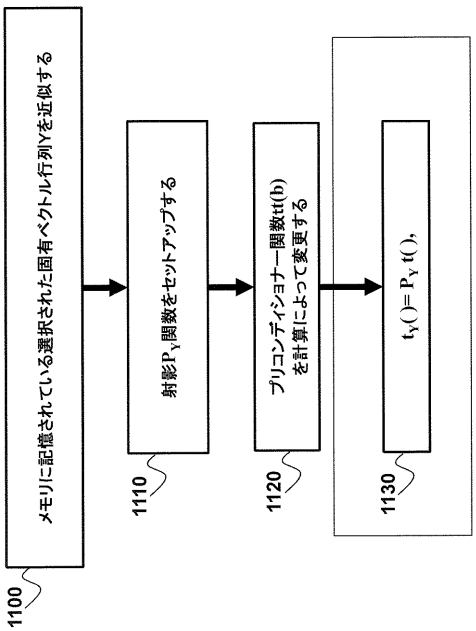
【図９】


係数行列を近似する
900
反三角形因数分解
A= Q K Q^T
を計算する
910
以下を計算することによって
プリコンディショナー関数 t(b) を適用する
920
y = Q^T b
930
後退代入によって
K z= yを直接解く
940
t(b) = Qz
950

【図１０】


係数行列を近似する
900
行列 A=V ΣV^T
の固有値分解を計算する
1010
以下を計算することによって
プリコンディショナー関数t(b)を適用する
1020
y = V^T b
1030
|Σ| z= yを直接解く
1040
t(b) = Vz
1050

【図１１】


メモリに記憶されている選択された固有ベクトル行列Vを近似する
1100
射影P_Y関数をセットアップする
1110
プリコンディショナー関数tt(b)
を計算によって変更する
1120
t_Y( )= P_Y t( ),
1130

フロントページの続き

(72)発明者　アンドレイ・ニアゼフ
アメリカ合衆国、マサチューセッツ州、ケンブリッジ、ブロードウエイ　２０１、ケアオブ・ミツビシ・エレクトリック・リサーチ・ラボラトリーズ・インコーポレイテッド
(72)発明者　藤井　悠太
東京都千代田区丸の内二丁目７番３号　三菱電機株式会社内

審査官　黒田　暁子

(56)参考文献　特開２０１２－１９４９６０（ＪＰ，Ａ）
Takuma TANIDA, Toshiyuki OHTSUKA, Preconditioned C/GMRES Algorithm for Nonlinear Receding Horizon Control of Hovercrafts Connected by a String, Proceedings of the 2004 IEEE International Comference on Control Applications

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０５Ｂ　１３／０４