情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2010-SLDM-144 No.4
Vol.2010-EMB-16 No.4
Vol.2010-MBL-53 No.4
Vol.2010-UBI-25 No.4
2010/3/26

ECU ソフトウェアのシミュレーション実行時における状態方程式の遷移解析手法

高橋　ひとみ[†1]　張　綱[†1]　小松　秀昭[†1]

ECU ソフトウェアのモデル開発中において、設計したプラントのモデルと実機との差分を補正するため、ECU コントローラに実機のプラントと適合するためのマップやルックアップテーブルのような非線形要素が追加される。このマップにおけるエントリの設定は実機によるテストをもとに手動で行う。そのため、設定したエントリの組み合わせにより、設計者が意図せずシステム全体が制御不可能な状態に陥る可能性がある。しかしマップを含んだ制御システムは、マップ中のエントリ組み合わせにより状態方程式が爆発的に増加し、静的な解析が難しい。そこで本論文はマップのような非線形要素が含む制御システムによるシミュレーションを実行し、実際に制御システムが使用する制御方程式およびその遷移の解析を行う。これにより既存の手法では組み合わせ爆発が発生し解析が困難であった非線形要素の含んだモデルの解析が可能となる。

An analytical method for transitions of state equation in the ECU software simulation

Hitomi Takahashi ,[†1] Gang Zhang [†1]
and Hideaki Komatsu[†1]

This paper shows a novel method to analyze a control model including calibration maps. In the process of development for ECU software, we need to add maps into controller models to calibrate difference between a model and an actual plant. These entries of calibration maps are configured manually based on a test of the actual plant. Thus, there is a possibility that some combinations of map entries make the whole system out of control. It is difficult to analyze such the model including maps statically because the number of state equations for the model increases rapidly by combinations of the entry in calibration maps. We propose and design a method to analyze such the system. This method simulates the model and detects transitions of state of equations. We can analyze the model including non-linear components incremental and dynamically.

1. はじめに

自動車の技術の発展に伴い、高機能かつ高性能な自動車の開発が可能となった。自動車の基本性能の向上はもちろんのこと、環境への考慮、品質の向上を行うため自動車の電子化が進んでおり、自動車の大幅な電子化の結果、ECU ソフトウェアが原因であるリコールの割合が増加している。このようなソフトウェアが起因によるリコールの増加を受け、自動車に特化した機能安全規格である ISO26262 の施行に向けた動きが加速しており、安全な自動車のソフトウェア開発は急務である。しかし、先に述べたように自動車の大幅な電子化、機能向上の結果、ECU ソフトウェアは大規模かつ複雑になっており、既存の開発手法では開発時にソフトウェアの問題を全て発見することが困難となっている。

ECU ソフトウェアの開発ではコントローラ、プラント、アクチュエータを Simulink を始めとするブロック線図などのモデルによって記述し、シミュレーションを行うモデルベース開発手法が普及している。通常 ECU ソフトウェアの開発では、モデルでの開発段階前に各プラント、アクチュエータ、コントローラに対し制御方程式を定め、それを元に Simulink などでモデルを形成し、製品に搭載させるため各モデルの詳細化となるプロセスが多い。その際、モデルと実機の差分を適合させるため、マップやルックアップテーブルといった非線形要素がモデル中に追加される。例えばプラントモデルと実機のプラントでは挙動が異なる場合があり、コントローラに適合用のマップが追加される。

実機へコントローラを適合をするためのマップ (マップ) といった非線形要素は新たな制御方程式をモデル中に発生させ、さらに複数の非線形要素の組み合わせにより状態方程式が爆発的に発生し、設計者が意図しない不安定な状態を制御システムにもたらす可能性がある。しかし、非線形要素の追加により発生した全制御方程式を静的に全て解析することは、組み合わせ爆発を起こしているため不可能である。そこで本論文では、シミュレーションを実行し実際に非線形要素が含まれるモデルの解析を動的かつ増分的に行う。これにより、非線形要素の組み合わせから発生する全制御方程式を解析せず、必要な状態方程式やその遷移を効率的に解析できる。これらの解析結果をモデル間の動作差分と比較することで、もとの線形のみで構成された制御方程式からどの程度動作が異なるか判明でき、またこれらの解析結果を実機の製品にて使用することで、製品の安全性向上が可能となる。

本論文は以下の構成を取る。まず第 1 節で本論文の背景である ECU ソフトウェア開発の現状について述べ、第 2 節で ECU ソフトウェアの非線形要素により発生する問題について

†1 日本アイ・ビー・エム株式会社　東京基礎研究所
IBM Research - Tokyo

Vol.2010-SLDM-144 No.4
Vol.2010-EMB-16 No.4
Vol.2010-MBL-53 No.4
Vol.2010-UBI-25 No.4
2010/3/26

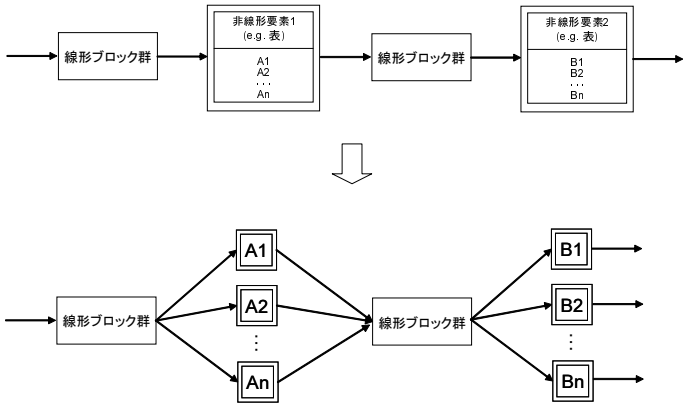

図 1 非線形要素による状態方程式の増加
Fig. 1 Increase of State Equations by Non-linear Components

述べる。次に、第 3 節では前節で述べた問題に対する解決手法を提案し、第 4 節で提案した手法を用い評価を行う。第 5 節では本論文が提案する手法の応用例を説明し、第 6 節では本論文と関連するモデル検証用のツールとの比較を行う。最後に第 7 節で本論文のまとめと今後の課題について述べる。

2. モデル中の非線形要素による状態方程式の増加

ECU ソフトウェア開発ではプラント、コントロール、アクチュエータなどの状態方程式および伝達関数を作成しこれを基に Simulink などによりモデル化を行う。通常、状態方程式から構成された線形システムには非線形要素はなく、系全体を現す状態方程式もしくは伝達関数は一意に算出でき、線形システムを対象にした既存の制御理論による解析が静的に可能である。しかし ECU ソフトウェア開発が進行しモデルを実機へ対応させる場合、既存の線形システムのみのモデルへ多くの非線形要素を追加する必要がある。例えば実機のプラントとモデルのプラントの差分をコントローラに適合させるマップという非線形要素を追加するため、線形システムに用いる制御理論をそのまま適応できない。

そこで設計者は設計した制御システムが正しく動作し、かつ仕様が満たされているかをテストするため、Simulink のようなブロック線図を用い設計したモデルのシミュレーションを行う。このシミュレーション上の動作確認テストで使用されるプラントは、もちろん実機ではなくモデルであるためシミュレーション用モデルと実機のプラントの差分が発生してしまう。そこで ECU ソフトウェアの設計者はモデルと実機のプラント間の差分をコントローラに適合するため、コントローラ内にマップを追加する。この ECU ソフトウェアにおける適合用のマップは非線形要素となり、マップのエントリである値は実機でテストを行った結果から技術者が手動で設定する。その際マップエントリ同士の組み合わせによる制御システムの影響を考慮していないため、テストケース以外の入力やマップエントリの組み合わせにより、システム全体が制御不能となる可能性がある。

このようなシステムに非線形要素が追加されたモデルには、不安定な線形システムからなる切り替え線形システムへ変形可能である。ただし、この切り替えシステムの解析は非線形要素による組み合わせ爆発を起こし状態方程式が無数にできてしまうため、既存の静的解析はできない。例えば図 1 では、線形ブロック群の間に非線形要素のマップである A、B が追加されている。この表は入力の値より出力する値を決定する単純な操作を行う。A には A1 から AN、B1 から BN までの N 個のエントリが定められており、シミュレーションの選択しうる組み合わせは $n \times n = n^2$ となる。非線形要素によって発生する状態方程式はこのマップエントリの組み合わせ数と同数となるため、例えば、A1-B1、A1-B2、A1-B3… AN-BN のそれぞれ各組み合わせ毎に異なる状態方程式が発生する。つまり図 1 では、シミュレーションが選択する組み合わせの数である n^2 の状態方程式が発生する。このように非線形要素が追加されると設計者の意図しない状態方程式が乗算的に発生し、非線形要素が多いモデルでは静的な解析は困難となる。

3. 状態方程式の遷移状態解析手法

第 2 節でも述べたように、線形システムに非線形要素を追加すると膨大な数の状態方程式が発生してしまう。しかしシミュレーションおよび実機上でシステムが動作するとき、これらの全状態方程式へ遷移が発生するわけではない。実際に発生する制御方程式の遷移は、全状態方程式のごく一部分のみであるという性質がある。そのため発現した全状態方程式を対象に解析するのではなく、実際にシミュレーションを動作させ遷移が発生した状態方程式のみを解析することで、膨大な数の状態方程式の中から必要な状態方程式のみを解析できる。これにより非線形要素が追加された制御システムにおいても必要な状態方程式およびその遷移が解析可能になる。

しかしこの手法では発現した全状態方程式を解析しないため、もしシミュレーションや実機によるテストケースが不十分な場合、シミュレーションでは遷移が発生しなかった状態方程式への遷移が実機では発生する可能性がある。もし解析をしていない状態方程式への遷移がユーザの使用中に発生し、その状態方程式が制御システムに危険を及ぼすような方程式である場合、ユーザへの安全性が損なわれる。そこで実機において解析をしていない状態方程

Vol.2010-SLDM-144 No.4
Vol.2010-EMB-16 No.4
Vol.2010-MBL-53 No.4
Vol.2010-UBI-25 No.4
2010/3/26

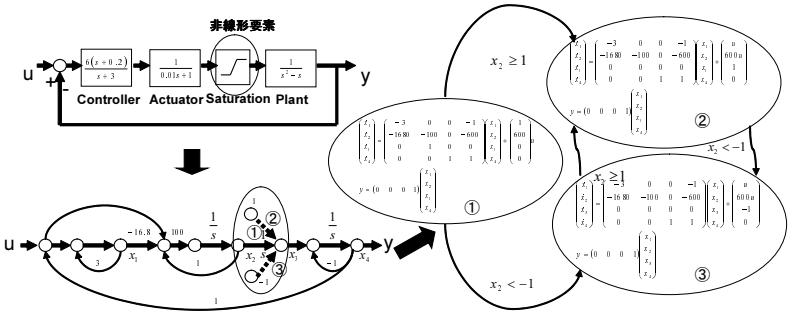

図 2 非線形要素が含まれるモデルの変形例
Fig. 2 Example of A Model Including A Non-linear Component

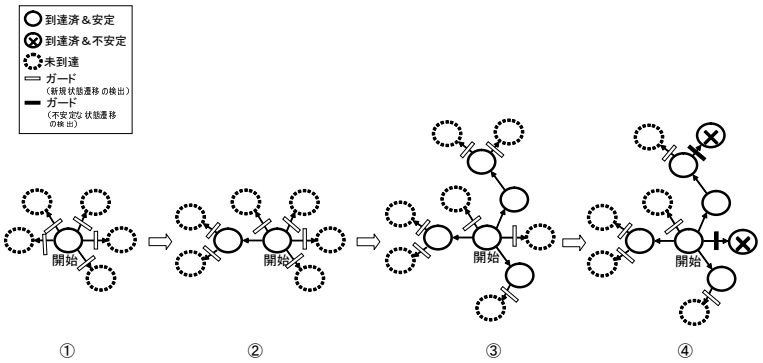

図 3 動的かつインクリメンタルな検証手法
Fig. 3 Method of Incremental and Dynamical Verification

式へ遷移が発生した場合、実際に遷移が発生する前に制御システムのリセットを行い遷移を初期状態に戻すことで、解析を行っていない状態方程式への遷移が発生した場合でもユーザの安全性を確保できる。

後節では非線形要素が追加されたモデルの線形変換と、シミュレーションおよび製品動作中の動的な状態方程式の遷移検証について詳細を述べる。

3.1 線形システムへの変形

線形要素から制御システムへ非線形要素が追加された場合、非線形要素に入力する状態変数の値により出力値が決定される。その場合、非線形要素へ入力する状態変数の領域により線形近似が可能であり、線形システムとして取り扱いできる。このように非線形要素への入力により、非線形要素以降の制御システムが切り替わることから、安定、不安定な線形システムからなる切り替え制御システムとして制御システム全体を変形することで、状態方程式の遷移状況を把握し、解析可能となる。

図 2 に非線形要素が入った線形システムへの変形例を挙げる。図 2 ではコントローラ、アクチュエータ、プラントに対応する伝達方程式が与えられた線形システムが存在し、プラントの前に非線形要素である飽和要素が追加されたとする。非線形要素が追加される前は、各伝達方程式から系全体を現す伝達方式が算出でき、その特性方程式の根の実部は全て負となる安全な系である。しかし飽和要素の追加により、飽和要素へ入力する状態変数である x_2 の値より、上限値、中間値、下限値での 3 状態の制御システムが発現し、それぞれに状態方程式が存在する。このとき飽和要素からの出力を $x_2 < -1$ ならば $x_2 = -1$、$-1 \leq x_2 < 1$ ならば $x_2 = x_2$、$x_2 \geq 1$ ならば $x_2 = 1$ とすると、それぞれの飽和要素の領域で算出される状態方程式は図 2 の円の式となり、遷移条件は矢印と x_2 の値となる。特にこの遷移図では $x_2 < -1$ もしくは $x_2 \geq 1$ の条件で遷移する 2 つの状態方程式は固有値から不安定と判断でき、不安定な状態方程式へ遷移した場合、制御システム全体が不安定となる可能性があり、原因はこの状態方程式であると判別できる。本システムは非線形要素が含まれた制御システムをこのような状態方程式の遷移表に変換し、遷移を発生する内部変数に注目することで、システムの遷移状態を把握できる。これにより、非線形要素の追加とそれらの組み合わせから設計者が意図せず発現した状態方程式を把握でき、制御システム全体の動作理解を助ける。

ただし第 2 節でも述べたように、これらの状態方程式、遷移情報は組み合わせ爆発により膨大な数となるため、事前に全状態方程式を算出するわけではない。シミュレーションによって実際に状態方程式の遷移が発生する場合にのみ、シミュレーションを一度停止させ、状態方程式を算出する。これらの詳しい動作手順は次の小節で説明する。

3.2 動的かつインクリメンタルな状態遷移検証

本システムは全ての状態方程式および遷移条件を事前に算出するのではなく、シミュレーションを動作させ、実際に遷移した状態方程式のみを算出、解析する。また、本システムは一サイクル毎にシステムがどの状態方程式へ遷移するかを判断しなければならない。もしこの判断に大きな負荷がかかってしまう場合、ユーザへの利便性が損なわれる。そのため本システムではこれらの状態方程式の遷移の判断を低い負荷で行う必要がある。次に本システムの動作手順と状態方程式の遷移の検証方法について詳しく述べる。

動的かつインクリメンタルな検証

本システムはシミュレーションを動作させながら、必要な状態方程式の算出および解析

Vol.2010-SLDM-144 No.4
Vol.2010-EMB-16 No.4
Vol.2010-MBL-53 No.4
Vol.2010-UBI-25 No.4
2010/3/26

を行う。図 3 に示すように、シミュレーションを行うと状態方程式は以下の 3 つに区分できる。1、該当する状態方程式へ遷移が発生し、さらにその状態方程式ではシステムが安定となる（到達済&安定）2、該当する状態方程式へ遷移が発生し、さらにその状態方程式ではシステムが不安定となる（到達済&不安定）3、遷移が発生していない状態方程式である（未到達）。遷移先の状態方程式がこれらのどの状態方程式に区分されるかにより本システムの挙動が変化する。

また、到達済&不安定および未到達の状態方程式の前にはガードと呼ばれるトリガが設置される。シミュレーション実行中に状態方程式の遷移が発生し、このガード条件と当てはまる場合、一度シミュレーションを停止させ本システムが動作する。遷移先の状態方程式が到達&不安定である場合、制御システムが不安定となるため制御システムおよびシミュレーションを終了し、必要に応じ停止した環境や状況をログする。遷移先の状態方程式が未到達である場合、シミュレーションを停止し次に遷移する状態方程式を算出、安定性解析を行う。この結果、状態方程式が安全であると判断した場合は、この操作を発生させたガード条件を削除し、遷移先の状態方程式から遷移できる隣接の状態方程式へのエッジを算出する。それらのエッジに対し、ガード（新規状態遷移の検出）を設置しシミュレーションを再開させる。シミュレーションを行いながらこのガードを設置していくことで、膨大な数の遷移から実際に遷移が発生しうる周囲のみにガードを設置でき、製品に搭載した場合には遷移状態が必ず既知の状態方程式のみでシステムが動作できる。これにより、制御システムの暴走が起因となる事故を防止できる。

次に一連の解析動作手順を図 3 を用い説明する。

(1) シミュレーションの初期状態からどの状態方程式からシミュレーションが開始するかが一意に定まるため、シミュレーション開始前に使用される状態方程式の算出と解析を行う。その状態方程式が安全である場合、隣接の全状態方程式へのエッジにガード（新規状態遷移の検出）を設置し、シミュレーションを開始する。

(2) シミュレーション実行中に状態方程式の遷移が発生し、遷移先の状態方程式が未到達である場合、ガード条件と一致するためシミュレーションが停止する。本システムでは次の遷移先の状態方程式を算出し、さらにその状態方程式が安全であると判明した場合、この操作を発生させたガードを除去する。また遷移先における状態方程式の隣接ノードへのエッジにガード（新規状態遷移の検出）を設置し、シミュレーションを再開させる。

(3) シミュレーションを行い、新たな状態方程式へ遷移する毎に状態方程式の解析、ガードの除去、隣接ノードのエッジへガードを設置し、実際に遷移が発生する状態方程式のみを解析する。

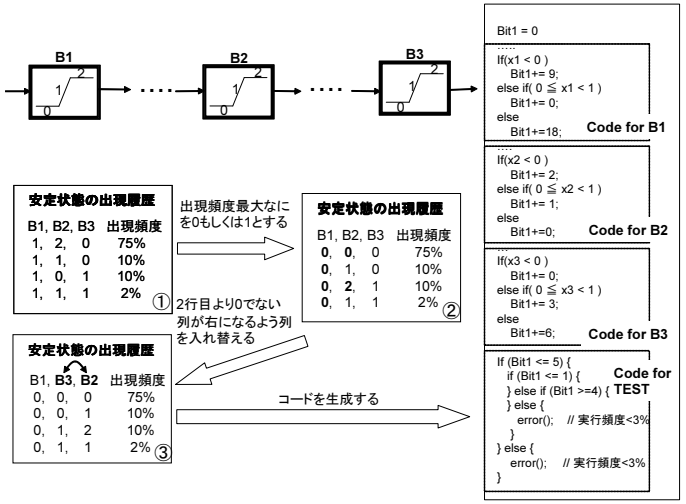

図 4 コーディングの概要
Fig. 4 Outline of Coding Method

(4) シミュレーションを実施し、実際に遷移した状態方程式を解析する中に制御システムが不安定となる状態方程式が存在した場合、該当する状態方程式を到達&不安定な状態方程式に分類しシミュレーションを中断する。また、この動作を発動させたガードを新規状態遷移の検出から不安定な状態遷移の検出に変更する。

状態遷移の検証方法

本システムでは状態方程式の遷移状況を把握するために、シミュレーションの 1 サイクル毎に制御システムがガードの条件に一致するかを判断しなければならない。このガードはシミュレーションの 1 サイクル毎に比較されるため、判断処理の負荷が低い必要がある。ここでは制御システムにおける状態方程式の遷移状況の把握、ガードの生成手法について説明する。

非線形要素によって発生する状態方程式は、非線形要素の区間により定められる。つまり飽和要素である場合、飽和要素への入力値によって上限、下限および中間部の 3 区間に分割でき、それぞれの飽和要素への入力値が取りうる区間により制御システムの状態方程式が異なる。もし、非線形要素がマップである場合、マップに含まれるエントリ毎に制御方程式が変わる。つまり非線形要素の入力値がとりうる区間を監視することで、制御システムの状態方程式が把握できる。

Vol.2010-SLDM-144 No.4
Vol.2010-EMB-16 No.4
Vol.2010-MBL-53 No.4
Vol.2010-UBI-25 No.4
2010/3/26

そこで、入力値がどの非線形区間に存在するかを把握するため、図 4 に示すように、シミュレーション開始前に各非線形要素の区間に初期 ID を割り当てる。飽和要素の場合 3 区間となるので、初期 ID は 0、1、2 となる。もし各飽和要素の B1、B2、B3 の入力区間がそれぞれ 1、2、0 である場合、この各 ID の組み合わせがそれぞれの状態方程式を識別する ID となる。本システムは 1 サイクル毎にこの ID をログすることで、シミュレーションにおける ID の出現率も算出する。先に述べたように、非線形要素によって発現する状態方程式は全状態方程式への遷移が発生するのではなく、ごく一部の限られた状態方程式のみであり、さらにその発現確率は大きく偏りがあると推測される。つまり全テストケースのシミュレーションにおいて使用される状態方程式は、ある数個の状態方程式が非常に発現確率が高く、まれに他の状態方程式への遷移が発生する。

本システムではこの出現確率の偏りを使用し、ガード条件と一致するか否かを少ない負荷で判別可能にする。具体的には各非線形要素の順列や非線形要素の区間の ID 割り当てを交換することで、ブロックの条件に一致するかどうかを判断する処理を低減させる。この処理を本論文ではコーディングと呼ぶ。コーディングは以下の手順を取る。

(1) 状態方程式の解析結果が安定でありかつその出現率が最大のエントリを 0 になるよう、非線形要素の区間に割り当てた ID を交換する。
(2) 最小値からのハミング距離を基準にしてエントリを昇順でソートする。また、ハミング距離が等しい場合、値（ラティス距離）を基準にし、同じハミング距離のエントリ同士で昇順にソートする。
(3) 状態方程式の解析結果が安定でありかつその出現率が次に最大のエントリを選択する。
(4) 選択されたエントリのハミング距離に応じ最大値もしくは最小値に集約するかを決定する。その後、もしハミング距離が大きく安定なエントリが多い場合、出現率が最小値からハミング距離の最も遠い安定なエントリを選択し、そのエントリが最大値になるよう ID を交換する。
(5) 他のエントリの値の依存関係が壊れないよう、選択エントリが最大もしくは最小となるよう列を交換する。
(6) 未操作のエントリに安定となるエントリがなくなるまで 3 を繰り返す。

以下のコーディングの例を図 4 を用い説明する。モデルには飽和要素である B1、B2、B3 があり、それぞれの区間に初期 ID として 0、1、2 が割り当てられている。これらの非線形要素が含まれるシミュレーションを実行し、それぞれの非線形要素に対する入力値の区間とその組み合わせに対する出現率を記録すると図 4 で示される表 1 のような情報が取得できる。表のエントリである 1、2、0 は B1 では 1、B2 は 2、B3 は 0 の区間を入力値が取ったことを示す。今、表 1 に記載されいているエントリが全て安定であった場合、出現頻度が一番高いエントリである 120 が基準となるエントリとして選択される。このエントリが全て 0 となるよう各ブロックの区間に割り当てられている ID が交換される。B1 では 1 と 0、B2 では 2 と 0 を交換することで出現頻度が 75%のエントリは全て 0 のエントリとして処理できる。この状態が図 4 の表 2 となる。さらに表におけるブロックの順序は、実際にシュミレーションを行うブロック線図と異なっても支障がないため、安定な状態方程式のエントリ群が最小もしくは最大値へ接近させるよう、ブロックの順序である表の列を交換する。図 4 の表 2 であればブロックの B2 と B3 の順序を交換する。これにより安定であるエントリが 0 から順に並べられた状態となる。

この表はシミュレーションが一サイクルでとった遷移状態がどの遷移状態と一致するかを判断するためのコード生成に使用される。例えば B1 への入力である x1 が飽和要素の区間で 0 となる場合、ID は 1 となるよう図 4 の表 2 で決定したため、シミュレーションが実行された状態を示す Bit1 に 9 が加算される。飽和要素である B2、B3 についても同様な処理を行うと、Bit1 はシミュレーションの一サイクルにおいて入力がどの飽和区間となったかを表現できる。その際、コーディングを行うことで、状態を判定する処理がコーディングを行っていない表から生成された処理より低減できる。図 4 では Code for TEST と記述されている部分が、システムの遷移状態を検証する処理となる。

4. 評　　価

この節では本システムのプロトタイプ実装および評価を説明する。まず線形要素のみのブロック線図に疑似的なマップを追加し、本システムとシミュレーションを実行させた。評価ではマップを追加することによって生じる制御システムの変化および状態方程式の遷移を示す。

4.1 プロトタイプ実装および評価環境

実装環境として Linux 2.6.18 上に MATLAB R2008b および Simulink 7.2 を用い本システムのプロトタイプ実装を行った。本システムを用いる場合、図 5 のようにモデルで用いる非線形要素の代わりに作成した S-Function と置換し、さらにモデルへ状態遷移検証用の S-Function ブロックを追加する。この S-Function は通常の非線形要素の働きと共に実行時にどの区間に入力があったかを本システムに通知する。また、モデルへ新たに追加される状態遷移検証用 S-Function は、実行時にシミュレーションを停止させ、現段階のシミュレーションにおける遷移状態を検証するよう本システムへ検証要求を送信する。そして検証結果の返信があるまで、シミュレーションを停止しさせ検証結果を待つ。

本システムのシステム構成は図 7 となる。Simulink 上で既存ブロックと置換した S-Function 以外の機能は、Simulink とは別のプロセスで動作し IPC により情報の交換を行う。まず、

Vol.2010-SLDM-144 No.4
Vol.2010-EMB-16 No.4
Vol.2010-MBL-53 No.4
Vol.2010-UBI-25 No.4
2010/3/26

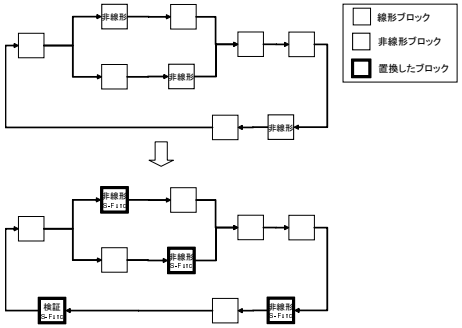

図 5 ブロックの置換
Fig. 5 Replacement of Blocks

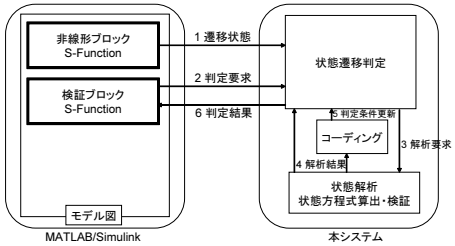

図 6 システム構成図
図 7 System Configuration

Simulink モデル中に存在する非線形ブロックである S-Function は入力値がどの区間の値であったかを本システムへ送信し、シミュレーション一サイクル毎に一回実行される検証ブロックの S-Function は、現状の状態方程式の遷移状況がどのようになっているか本システムの状態遷移判定へ判定要求のメッセージを送信する。検証ブロックは本システムからの結果を受信するまで、Simulink の動作を停止させる。

Simulink の検証ブロックより判定要求メッセージを受信した状態遷移判定モジュールはこれまでの ID を合わせ、コーディング表より現在の状態方程式を特定する。その後、現状の状態方程式がどのような方程式であるか判定するため、状態解析、状態方程式算出・検出モジュールに解析要求を行う。このモジュールはもし新規の状態方程式であるならば、その方程式を算出し、必要ならば安定性解析を行う。状態方程式の遷移状況が判明した際、その状態方程式の出現率および性質が変化する場合がある。それによりコーディングテーブルが変化するため、コーディングモジュールに現状の状態方程式およびその性質を状態解析、状態方程式算出・検証モジュールからコーディングモジュールへ送信し、それを受信したコーディングモジュールは必要に応じ、コーディング表を再計算する。コーディング表に更新があった場合、状態遷移判定モジュールに更新済みのコーディング表が渡される。また、状態解析、状態方程式算出・検出モジュールは受信した状態方程式から、シミュレーションの停止、ログの出力、シミュレーションの続行など必要な処理の指示を状態遷移判定モジュールに送信する。状態遷移判定モジュールはそれを受信後、直ちにシミュレーションを停止させている検証ブロックへ判定結果を返信する。検証ブロックはその指示に従い、シミュレーションの再開、停止、状態のログなどを行う。

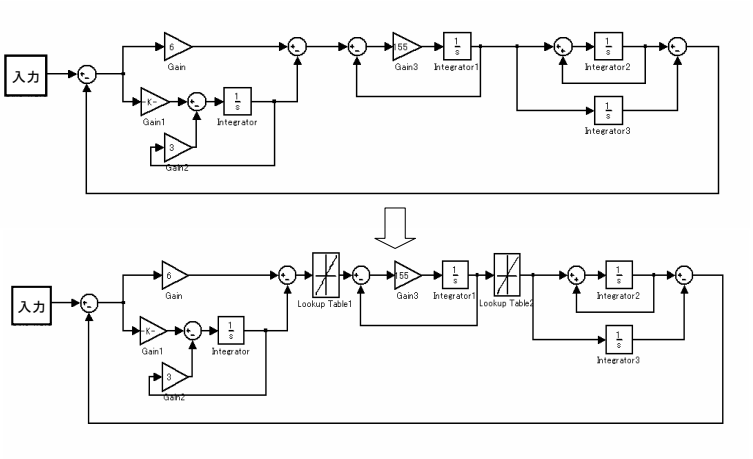

図 8 評価モデル
Fig. 8 Evaluation Model

4.2 評　　価

ここでは第 4.1 節で述べたプロトタイプ実装を用い評価を行った。評価項目として 1：線形要素のみの制御システムへ非線形要素が追加された際の動作の違いを示す。2：非線形要素が含まれるモデルのシミュレーションを行い、本システムで取得した状態方程式の遷移を示す。以上の 2 点を評価する。

評価に使用したモデルを図 8 に示す。上部に示す線形モデルへ適合用のマップとして Lookup Table（LUT）を 2 箇所に追加した。LUT を追加する以前の伝達関数の根は $-1.5387, -0.0071 \pm 0.0131i, -0.0054$ となり安定である。またこの線形モデルに追加した LUT は階段関数と同様なエントリとなっており、LUT への入力を x、LUT からの出力を $x^{'}$ とした場合、Lookup Table1 では $x^{'} = -45\ (x < -40), x^{'} = -40\ (-40 \leq x < -35), x^{'} = -35\ (-35 \leq x < -30), \cdots, x^{'} = 45\ (45 \leq x)$ となっている。また Lookup Table2 も同様に、$x^{'} = -20\ (x < -20), x^{'} = -15\ (-20 \leq x < -15), x^{'} = -10\ (-15 \leq x < -10), \cdots, x^{'} = 20\ (20 \leq x)$ というエントリとなっている。

非線形要素によるモデルへの影響

図 8 に示すモデルを用い、線形要素のみのシミュレーション結果および非線形要素を含んだモデルのシミュレーション結果を示す。

非線形要素として Lookup Table1、2 はそれぞれエントリが 19、9 エントリ存在し、こ

Vol.2010-SLDM-144 No.4
Vol.2010-EMB-16 No.4
Vol.2010-MBL-53 No.4
Vol.2010-UBI-25 No.4
2010/3/26

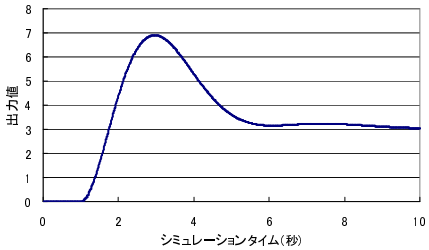

図 9 線形モデルの出力値 (入力:0-3)
Fig. 9 Output of Linear Model (Input:0-3)

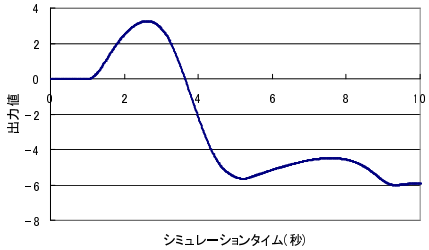

図 11 非線形モデルの出力値 (入力:0-3)
Fig. 11 Output of Non-Linear Model (Input:0-3)

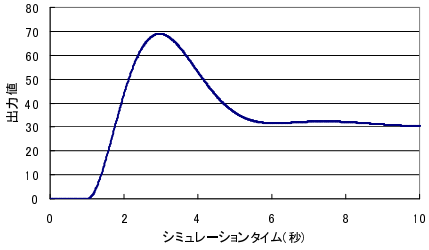

図 10 線形モデルの出力値 (入力:0-30)
Fig. 10 Output of Linear Model (Input:0-30)

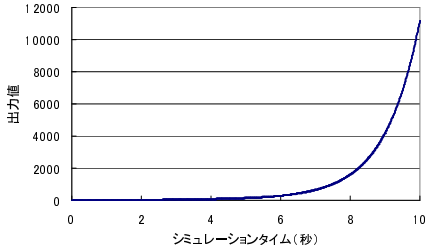

図 12 非線形モデルの出力値 (入力:0-30)
Fig. 12 Output of Non-Linear Model (Input:0-30)

のテーブルによって発現する状態方程式は計 $19 \times 9 = 171$ 個もの方程式がエントリの組み合わせにより発現する。また Lookup Table 1、2 ともに入力値として 0 とならない場合、遷移する状態方程式の固有値解析および伝達方程式の根から判明する安定性は全て不安定な状態方程式となり、状態方程式の組み合わせ爆発に加え静的な安全性解析は困難である。これらの LUT より発現した状態方程式が原因で、テーブルの追加前と追加後では大きく制御システムの挙動が異なる。

図 9、10、11、12 のグラフは x 軸はシミュレーションタイムを y 軸は出力値の y を示しており、シミュレーション時間を 10 秒、ソルバーを ode4（Runge-Kutta）の固定ステップとし、ステップサイズは 0.01 秒のシミュレーション結果である。図 9、10 は LUT を含まない線形要素のみのモデルであり、図 11、12 は LUT を含めたモデルをシミュレーションしたものである。また、図 9、11 はシミュレーションの入力をステップ入力とし初期値は 0、最終値は 3 とし、図 10、12 も同様に入力をステップ入力とし、初期値は 0、最終値は 30 とした出力結果である。

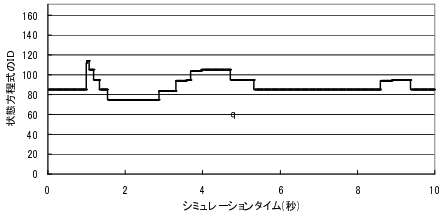

図 13 状態方程式の遷移状態 (入力:0-3)
Fig. 13 Transitions of State Equation (Input:0-3)

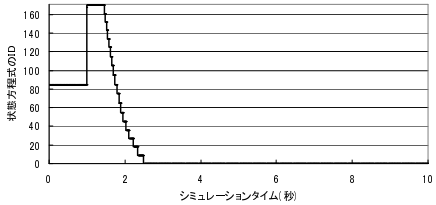

図 14 状態方程式の遷移状態 (入力:0-30)
Fig. 14 Transitions of State Equation (Input:0-30)

これらの出力結果を見ても分かるように単純な LUT ですら、線形要素のみのモデルへ追加すると出力値に大きく影響を与える。例えば同じ入力であるにもかかわらず、図 10 は最終値の 30 に近づき安定するが、LUT を入れたモデルである図 12 は不安定となっている。また LUT を含むモデルの結果である図 11 は発振が発生するが、-6 付近で定常状態となった。このように、非線形要素により発現した状態方程式の固有値解析による判定が全て不安定となったとしても、状態方程式の遷移によっては安定な出力になる可能性もあり、また、急激に不安定となる可能性もある。

非線形要素を含むモデルにおける状態方程式の遷移

次に本システムを用いて、シミュレーション実行時における状態方程式の遷移状況を取得した。結果を図 13 と図 14 に示す。シミュレーションモデルおよび実行環境は先の評価と同一である。図 13 における入力の最終値は 3 とし、図 14 の最終値は 30 であり、横軸はシミュレーションタイムを、縦軸は状態方程式の ID を示している。状態方程式の ID は LUT の区間を昇順で割り当てており、Lookup Table1 では 0 から 18 までを、Lookup Table2 では 0 から 9 までを割り当て、これらの組み合わせにより一意な状態方程式の ID を割り当てた。ただし今回はテストケースが少ないため、コーディングによる ID の入れ替えは行われない。そのため (LUT1, LUT2) = (0, 0) ならば状態方程式の ID は 0、(0, 1) ならば ID は 1、(1, 0) ならば ID は 9 のように割り当てられている。

図 13 は出力が安定となる入力である。このシミュレーションモデルで発現する唯一の安定な状態方程式の ID は 85 となり、その状態方程式が多く実行されていることが分かる。また、図 14 は出力が不安定となる入力が与えられており、不安定な状態方程式である ID 0 の状態方程式からシミュレーションタイムが 2.5 秒以降遷移が発生しないため、出力が不安定となっている。このように、安定な状態方程式から離脱した時間や不安定な同一の状態方

Vol.2010-SLDM-144 No.4
Vol.2010-EMB-16 No.4
Vol.2010-MBL-53 No.4
Vol.2010-UBI-25 No.4
2010/3/26

程式の定住時間を測定し、ある一定時間経過した場合、システムのリセットや状態を保存することで、制御システムが異常となった時間やその原因となった状態方程式が特定できる。

5. 応 用 例

本システムを用い、システムの遷移状態が判明することで以下の応用が考えられる。まず、シミュレーションにより発見した危険な状態方程式への遷移が検知でき、実機上でも状態方程式の遷移が把握できるため、危険な状態方程式へシステムが遷移する直前に実機を停止しシステムをリセットできる。例えばシミュレーション中に出力値が大きく逸脱するような状態方程式が発見され、その状態方程式はパラメータ調節によって除去が不可能という状況がありうる。その際、通常の入力ではその状態方程式の遷移が発生しにくい場合でも、本システムを応用することで、万が一その状態方程式に遷移が発生した場合 ECU ソフトウェアを停止しリセット可能である。

また本システムは新規の状態方程式の遷移を検知できる。そこで、製品として販売後、シミュレーションによって実行されなかった状態方程式へ遷移が発生した場合、その遷移を検知できる。新規の状態方程式への遷移が発生した入力や状態のログを行い、ディーラーやネットワークを通じ情報を収集することでシミュレーションにおけるテストケースの見直しやパラメータの調整などが可能となり、販売後の製品の安全性を向上できる。その他の応用として、マップがモデルへ追加された前後での状態遷移の変化や、線形要素のみのシミュレーションモデルとの動作や出力値の比較を行うことで、マップのパラメータ設定が初期モデルとどの程度近似しているかを把握できる。

6. 関 連 研 究

本システムの関連研究として、Simulink といったブロック線図を用いたシミュレーションへの検証ツールを挙げる。本研究と同様にブロック線図を線形化しモデル検証を行うツールとして Mathwork 社の Simulink Control Design[1] および MapleSoft 社の Maplesim BlockImporter[2] が挙げられる。Simulinik Control Design は Simulink のモデル中の動作点を自動的にもしくは手動で設定しモデルの線形化を行うツールである。Simulink のモデルを線形化することで、既存の制御理論が使用できコントローラのチューニングや補償器の設計が容易となる。しかし適合マップのような LUT の非線形要素は未対応であり、また制御方程式の遷移状況は把握できない。

Simulink Control Design と同様に Simulink モデル図の線形化が可能なツールとして Maplesim 社の BlockImporter が挙げられる。このツールは Simulink のモデル図を連立方程式系に変換し、数式により解析、最適化を行い、シンプルな数式モデルへと変形することでシミュレーションの高速化を実現できる。このツールは Simulink Control Design とは異なりモデル中の表を含め線形化可能であるが、どの状態方程式で実行されているかが判明できないため、もし表のパラメータの不具合により出力値が異常となった場合でも、どのマップのエントリによるかという判断が困難である。

7. まとめと今後の課題

本論文では ECU のソフトウェア開発において、シミュレーションモデルの非線形要素に注目をし、特にプラント実機とモデルの差分を適合するために用いられる手法であるマップに注目をした。マップを通常の線形モデルに追加することで、制御システムが遷移しうる状態方程式がマップ同士の組み合わせにより爆発的に増加し、静的な安定性の解析が困難であることを示した。そこで本システムではシミュレーションを実行しながら動的にかつ増分的に状態方程式の算出、遷移の把握を行うことで、マップにより複雑となった制御システムの動作を把握可能とする手法を確立した。

この手法は制御システムがシミュレーションおよび実機において状態方程式の遷移が把握できるため、もし未到達な状態方程式の遷移の把握および、シミュレーションで全体の状態方程式のうちどの程度実行されたかを把握できる。またこれらの結果を利用しモデル間の出力値や状態方程式の遷移状況を比較することで、マップエントリの不具合箇所が検知できるといった応用も考えられる。このように、本システムは状態方程式の数が爆発してしまう制御システムの解析および遷移状態の把握を可能とし製品の安全性が向上できる。

今後の課題としてモデル間の比較を実際に行い、どのマップエントリが原因で制御システムに異常が発生したか、また、内部状態や遷移状態から制御の出力値では検知できなった制御システムの異常を自動的に判明する手法を確立する。

参 考 文 献

1) Simulink Control Design,: http://www.mathworks.com/products/simcontrol.
2) Maplesim BlockImporter,: http://www.maplesoft.com/Products/blockimporter.