パラメタを含むハイブリッドシステムに対するアフィン演算を用いた記号シミュレーション

松本 翔太　上田 和紀

本研究の目的は，離散変化と連続変化の両方をともなう動的システムであるハイブリッドシステムの高信頼シミュレータの構築である．特に，パラメタを含むハイブリッドシステムの厳密なシミュレーションを行うことで，単なるモデルの到達可能範囲の解析だけではなく，望ましい性質を満たすためのパラメータ設計に役立てることを目標とする．区間演算を用いたシミュレータは到達可能範囲の包囲を求めることができるが，モデル中の不確定量がモデルの振る舞いに与える影響を計算することは依然として挑戦的課題である．

本論文では，記号計算を区間ニュートン法およびアフィン演算と連携させたシミュレーションアルゴリズムを提案する．本手法では，不確定量間の 1 次の依存関係を保存したまま計算を進めることが可能である．さらに，著者らが開発してきた記号シミュレータに本アルゴリズムを実装し，その性能を例題を用いて評価する．

The purpose of this research is to develop a highly reliable simulator of hybrid systems, i.e., systems involving both discrete change and continuous evolution. In particular, we aim at rigorous simulation of parametric hybrid systems, which enables not only the analysis of model's possible behavior but also the design of parameters that realize desired properties. Simulators with interval arithmetic can reliably compute a reachable set of states, but preserving the dependency of uncertain quantities in models is still challenging.

In this paper, we discuss a simulation method that is based on symbolic computation and cooperates with the interval Newton method and affine arithmetic, which is able to preserve first-order dependency of uncertain quantities. We implemented the algorithm on the symbolic simulator we have been developing and evaluated the performance of the method with example models.

1　はじめに

ハイブリッドシステムは離散変化と連続変化の両方をともなう動的システムである．ハイブリッドシステムのシミュレーションにおいては，計算機上の誤差が定性的に誤った結果を生むことにつながるため，モデルの設計の正しさを保証するためには正しさが厳密に保証された計算技術を用いることが重要になってくる．また，モデル化誤差や経年劣化など，モデルに含まれる不確定性を扱えることも重要である．ハイブリッドシステムの厳密な解析を行うツールはこれまでに多く開発されてきている [4] [6] [8] [17] [20]．その中のほとんどのツールは区間演算 [15] に基づいており，モデルの到達可能状態の過大近似（over-approximation）を求めることができる．

感度解析やコントローラの設計などにおいて，モデルに含まれる不確定パラメータがそのふるまいに与える影響の計算はさまざまな応用が考えられる．しかし，既存のツールではそれらの依存関係は無視されるか，計算の途中で暗黙的に利用されるだけで，最終結果には含められていなかった．本研究の達成目標は，パラメタを記号的に扱い，その依存関係を明示的に扱うことができるシミュレータを開発することである．この目標を達成するための課題として，(i) パラメタを含む微分方程式を解くことと (ii) パラメタを含むハイブリッドシステムの離散変化を厳密に計算することの 2 点があげられるが，本稿では後者に主眼を置き，アフィン演算 [2] を用いた記号シミュレーションアルゴリズムについて論じる．前者に関しては，微分方程

Symbolic Simulation of Parametric Hybrid Systems with Affine Arithmetic

Shota Matsumoto, 早稲田大学基幹理工学部情報理工学専攻, Waseda University.

Kazunori Ueda, 早稲田大学理工学術院情報理工学科, Waseda University.

式が線形であるか，もしくはパラメタを含む線形な微分方程式によって近似可能であることを仮定している（近似誤差の包含に，新たなパラメタを用いることで対応する）．アフィン演算は区間演算を拡張した演算であり，記号パラメータを用いることで不確定値間の一次の依存関係を保存可能な手法である．

ハイブリッドシステムのシミュレーションにおいては，各離散変化が発生する時刻を求める必要があるが，その条件は解析解を持たない非線形方程式で記述されている場合がある．このため，離散変化時刻の計算に関しても区間演算の技術を用いる必要がある．著者らの経験では，システムの連続挙動を解析解を持つ線形微分方程式に限定しても，離散変化条件を表す方程式がパラメタを含むために，しばしば記号的に解くことに失敗するケースに遭遇している．これを解決するため，本稿では区間ニュートン法と中間値の定理を利用した離散変化時刻の計算方法を提案する．これらの技術を連携させることで，アフィン演算単体では離散変化の計算に失敗するようなモデルに対してもシミュレーションを行うことが可能になった．本研究では提案手法を，著者らが開発してきたハイブリッド制約言語 HydLa の記号シミュレータ上に実装し，例題を用いた性能評価を行った †1．

2 ハイブリッド制約言語 HydLa

本稿で提案するシミュレーションアルゴリズムへの入力は HydLa [19] プログラムとして与えられる．HydLa は制約に基づいたハイブリッドシステムのモデリングを行う．制約による記述は宣言的でありながら，同期や条件分岐を含む制御構造も表現可能である．さらに，制約を用いることで不確定性を自然に扱うことも可能であり，記号パラメータを用いたプログラムの記号実行に適している．本節では，例題を用いて HydLa の概要を紹介する．HydLa の構文や意味論に関する詳細は [19] を参照されたい．図 1 は例題モデル（[5] より引用）の概要を示しており，図 2 は本モデルに対応する HydLa プログラムを示している．本モデルは 2 つの水槽を含んでおり，1 つ目の水槽の水がパイプを通して 2 つ目の水槽に流れ込んでいる．`x1` と `x2` は各水槽の水位を示しており，`v1` と `v2` は各水槽のバルブの状態を示している．v$i = 1$ (0) はそれぞれバルブが開いている（閉じている）ことを示している．HydLa における変数はすべて（暗黙的に）時刻に関する関数変数である．本プログラムは，時刻 0 において成り立つ性質を記述したいくつかの制約から成り立っている．制約 `INIT` はシステムの初期状態を記述している．本モデルでは，1 つ目の水槽の初期水位は $1.9 \leq$ `x1` ≤ 1.9001 を満たす不定値となっている．制約 `X1` と制約 `X2` は各水位の連続的挙動に関する制約である．時相演算子 `[]` はその制約が有効になった時刻以降常に `[]` のついた制約が成立することを意味する．`x1'` は時間微分 $\partial x1/\partial t$ を表す．含意記号 `=>` の右辺は，その左辺（ガードと呼ぶ）が満たされる時に有効になる．制約 `V1_CONST` と制約 `V2_CONST` は `v1` と `v2` が変化しないことを記述しており，これがこのモデルの通常時のふるまいとなっている．制約 `V1_OFF2ON`，`V1_ON2OFF`，`V1V2_OFF2ON`，`V2_ON2OFF` はバルブの開閉に関する制約であり，`x1-` のように変数のあとにマイナス記号がついたものは左極限値 $\lim_{t_l \uparrow t} x1(t_l)$ を表す．最後の 4 行はこれらの制約間の優先順位を `<<` を用いて宣言しており，水位が閾値をまたぐ時以外はバルブの状態は変化しないことを表現している．なお，より優先度の高い制約を持たない制約（本プログラムでは Vi`_CONST` 以外の制約）は常に有効な制約である．HydLa プログラムの宣言的意味はプログラムに記述された仕様を満たす軌道の集合であり，各時刻における仕様は制約間の優先順位に違反しないもののうち，極大無矛盾な制約の部分集合として与えられる．

3 区間演算とアフィン演算

区間演算 [15] は計算結果の取りうる範囲を保証するための演算である．区間演算における各計算は実数の閉区間に対して定義されており，$[l, u]$ という記法で実数の集合 $\{x \in \mathbb{R} \mid l \leq x \leq u\}$ を表す．本稿では，実数区間のなす全体集合を $\mathbb{I}$ と記述する．区間演算では，計算誤差や不確定量の範囲をともに区間として表現することができる．通常の区間演算ではラッピン

†1 本稿は，[13] で発表予定の論文をベースにしたものである

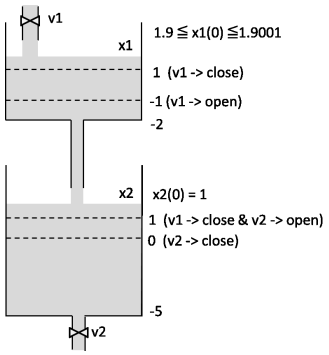

図 1　2 つの水槽のモデル

```
INIT <=>
   1.9 <= x1 <= 1.9001 /\ x2 = 1
   /\ v1 = 0 /\ v2 = 1.
X1 <=>
   [](v1 = 0 => x1' = -x1 - 2)
      /\ (v1 = 1 => x1' = -x1 + 3)).
X2 <=>
   [](v2 = 0 => x2' = x1)
      /\ (v2 = 1 => x2' = x1 - x2 - 5)).
V1_CONST <=> [](v1' = 0).
V2_CONST <=> [](v2' = 0).
V1_OFF2ON <=>
   [](v1- = 0 /\ x1- = -1 => v1 = 1).
V1_ON2OFF <=>
   [](v1- = 1 /\ v2- = 1 /\ x1- = 1 => v1 = 0).
V1V2_OFF2ON <=>
   [](v2- = 0 /\ x2- = 1 => v2 = 1 /\ v1 = 0).
V2_ON2OFF <=>
   [](v2- = 1 /\ x2- = 0 => v2 = 0).

INIT, X1, X2,
(V1_CONST, V2_CONST)
   << (V1_OFF2ON, V1_ON2OFF,
       V1V2_OFF2ON, V2_ON2OFF).
```

図 2　2 つの水槽を記述した HydLa プログラム

グ効果や数値間の依存関係の無視により，計算結果の区間の幅が広がってしまうという欠点がある．この欠点を解決するための手法は数多く研究されており [3] [15]，その 1 つとしてアフィン演算 [2] が存在する．アフィン演算は区間の 1 次の依存関係に着目した区間演算の拡張である．アフィン演算では，それぞれの数量 x はアフィン形式と呼ばれる以下の形式で与えられる．

$$x = x_0 + x_1\epsilon_1 + \cdots + x_n\epsilon_n$$

ここで，ϵ_i はノイズ変数と呼ばれる記号パラメタであり，$[-1, 1]$ の範囲の値を取りうる不確定値である．x_0 は x の中心値を表しており，x_i $(i > 0)$ は各ノイズ変数の x に対する影響の大きさを表している．ϵ_i が 2 つ以上の量の中に現れた場合，それらの量の間には依存関係が存在する．アフィン形式は，$[x_0 - \sum_{i=1}^{n} x_i, x_0 + \sum_{i=1}^{n} x_i]$ を計算することで対応する区間に直すことができる．計算機上でアフィン演算を行う場合，加算や乗算などの各計算を行うたびに，丸め誤差や非線形演算の近似誤差を包含するための新たなノイズ変数が導入される．ノイズ変数は計算結果の精度を向上させるために有用ではあるが，使用されるノイズ変数の数が増えるにつれて計算コストも大きくなってしまう．そうした精度とコストのトレードオフを制御するために，ノイズ変数の数を削減するアルゴリズムが提案されている [9]．この削減アルゴリズムはノイズ変数のうち，削減した場合の精度の低下が小さい順に指定された数までノイズ変数を減らすことが可能である．本稿で用いる手法は，各離散変化の計算の際にこの削減アルゴリズムを利用している（2 節）．

4　シミュレーションアルゴリズム

本節で提案するアルゴリズムは与えられた HydLa プログラムの軌道を記号的に計算するものである．図 3 にアルゴリズムの概要を示す．このアルゴリズムは基本 HydLa [19] のプログラムを入力とし，軌道の集合を出力する．基本 HydLa のプログラムは，元の HydLa プログラム中の個々の制約間の優先度に関する仕様を，その仕様を満たす制約の部分集合を要素とする半順序集合に変換することで得られる（この半順序集合中の順序関係として，部分集合間の包含関係を用いる）．別の言い方をすると，制約集合 S が制約 C を含むならば，S は必ず C より高い優先順位を持つような制約集合のみを，半順序集合中の要素として選ぶ．

本アルゴリズムは 2 つのフェーズを交互に切り替えながら進行する．1 つ目のフェーズは離散変化を扱う

Point Phase であり，2 つ目のフェーズは連続変化を扱う Interval Phase である．各フェーズでは，HydLa の意味論に従い制約の極大無矛盾集合（MCS）を計算して S に代入し，記号パラメータに関する条件を P に代入する (10 行目–16 行目)．MCS は各フェーズに対応した無矛盾性判定を行う関数を引数とする高階関数である．9 行目の $Subst$ は S 中の各数式に現れる時刻変数 t に対し，現在時刻 T を代入する．Point Phase の最後に，その時刻で値が不確定な変数に対して記号パラメータを導入し，その値の包囲を $Enclose$ 関数によって計算する（14 行目）．$Enclose$ 関数はアフィン演算を用いており，最終的に残すノイズ変数の数を指定することができる．6 節において，ノイズ変数の数が性能に与える影響について比較を行う．本アルゴリズムでは，アフィン演算との親和性を高めるため，P 中の全ての記号パラメタ（p_i とする）は $p_i \in [-1, 1]$ を満たすように正規化されている．例えば，下限と上限がそれぞれ $\underline{x}$ と $\overline{x}$ で表される不確定値は，$(\overline{x} + \underline{x})/2 + (\overline{x} - \underline{x})/2 \times p_i$ のようにする．Interval Phase では，再び MCS を求めるとともに，成立したガードの集合 A_+ と成立しなかったガードの集合 A_- を得る（16 行目）．その後，S 中の微分方程式を解き（17 行目）次の離散変化時刻の計算を行う．(21 行目)．$Subst$ は微分方程式の解を S と g と $\neg g$ 中の各数式に対して代入している．一般に，離散変化時刻の候補はパラメタの値の不確定性によりが複数存在しうる．例えば，x の軌道が $x(t) = -(t-1)^2 + x(0)$，$-1.5 \leq x(0) \leq 1.5$ であり，離散変化条件が $x(t) = 1 \ \vee \ x(t) = -10$ である場合には，2 つの候補時刻が存在する．第 1 の候補は $1 \leq x(0) \leq 1.5$ の場合の $t = 1 - \sqrt{x(0) - 1}$ であり，第 2 の候補は $-1.5 \leq x(0) < 1$ の場合の $t = 1 - \sqrt{x(0) - 9}$ である．このように複数の候補が現れるのは，パラメタの条件がある種のコーナーケース（例えばボールが壁にちょうど接するような軌道を描く場合）を含む場合である．$CompareMinTime$ や $FindMinTime$ では，複数の候補が存在するかどうかを判定することが可能である．もし複数の候補が存在する場合，取りうるすべてのふるまいを計算するためにはパラメタ空間を分割し，以降の計算をそれぞれの場合に分岐させる必要がある．著者らがこれまでに開発した記号シミュレータ [14] の記号計算はそうした分岐に対応することができるが，$FindMinTime$ で用いている区間ニュートン法はそのような場合を扱うことができないため，今後の課題として分岐に対応したアルゴリズムに拡張することを考えている．なお，本稿で扱う例題モデルはこうした分岐が発生しないモデルである．本アルゴリズム中の $FindMinTime$ と $Enclose$ 以外の手続きはすべて記号的に計算され，すべての不確定量は P 中のパラメタとして扱われる．

入力: $HydLa$: 基本 HydLa のプログラム,
$MaxT$: 最大シミュレーション時間

1: $MS := TopologicalSort(HydLa)$
// 候補となる制約集合のリスト
2: $V := GetVariables(HydLa)$
3: $T := 0$ // 現在時刻
4: $S := true$ // 現在成立している制約集合
5: $P := true$ // 記号パラメタの条件
6: $G_p := \emptyset$ // 直前の離散変化の原因となったガードの集合
7: **while** $T <$ $MaxT$ **do**
8: // Point Phase (PP)
9: $S := Subst(S, T)$
10: $(S, P, G_p, \_, \_) :=$
$MCS(S, MS, P, T, G_p, CheckConsistencyPP)$
11: **if** $S = false$ **then**
12: break
13: **end if**
14: $(S, P) := Enclose(AddParameters(S, P, V))$
15: // Interval Phase (IP)
16: $(S, P, G_p, A_-, A_+) :=$
$MCS(S, MS, P, T, G_p, CheckConsistencyIP)$
17: $S := SolveDifferentialEquation(S)$
18: **if** $S = false$ **then**
19: break
20: **end if**
21: $(MinT, P, G_p) := CompareMinTime($
$\{FindMinTime(Subst(g, S), P, G_p) \mid (g \Rightarrow c) \in A_-\}$
$\cup \{FindMinTime(Subst(\neg g, S), P, G_p)) \mid (g \Rightarrow c) \in A_+\}$
$\cup \{(MaxT - T, true)\})$
22: $T := MinT + T$
23: **end while**

図 3　シミュレーションアルゴリズムの概要

4.1 FindMinTime

FindMinTime 関数は離散変化時刻を計算する関数であり，離散変化時刻としてパラメタを含むことを許している．HydLa プログラムにおいて，離散変化はプログラム中のガードの成否が変化した時に発生する．ガードは連立不等式および方程式によって記述され，時刻に関する制約条件とみなすことができる．このため，*FindMinTime* の目的は与えられた条件の成否が変化する（成立から不成立へ，もしくはその逆）最も早い時刻を計算することとなる．

図 4 に *FindMinTime* のアルゴリズムを示す．まず，与えられた条件 G から原子境界条件のリストを得る．原子境界条件は，G 中の各原子条件（方程式もしくは不等式）の関係演算子を等号に置き換えることで得られる．例えば，$GetAtomicBoundaryConditions(t > 0 \wedge t^2 = 1 \wedge t \leq 1)$ の結果は $\{t = 0, t^2 = 1, t = 1\}$ となる．次に，*SolveTimeEquation* によって各方程式の解区間を計算する．*SolveTimeEquation* は与えられた方程式を解析的に解くことができるなら，その解析解を返す．そうでなければ，方程式を 4.2 節に示す方法で解き，解のリスト *sols* と，解区間に対応する新たなパラメータの条件を含んだ P を返す．それぞれの解は，g と組にして t_{list} に入れておく．これにともない，各ガードが現在の Interva Phase の初期時刻で成り立っているかどうかの対応表として Map_g を作成しておく（8 行目）．9 行目から 16 行目では，t_{list} 中の各時刻を始点とした時刻区間について，ガード全体の成否を調べていく．*PopMinimumTime* は t_{list} から最も早い時刻 t_{min} を取り出し，t_{min} において成否が変化する原子境界条件 g_{list} とともに返す．*CheckAndUpdateGuards* は t_{min} から始まる時刻区間における，ガード全体の成否と，各ガードについて成否を更新した対応表 Map_g を返す．

図 5 に *CheckAndUpdateGuards* のアルゴリズムを示す．3 行目から 9 行目では， t_{min} ちょうどにおける各原子条件の成否を計算している．関係演算子（$relop(g)$）が {‘=’, ‘≤’, ‘≥’} のいずれかの場合は成立し，そうでなければ成立しない．10 行目ではガード全体の成否を調べている．13 行目から 24 行目では，t_{min} から始まる時刻の開区間において同様の判定を行う．g が境界を通りすぎているため，更新時の判定が 3 行目から 9 行目のものと異なることに注意されたい．

入力: G: 微分方程式の解をガードに代入してえられた，時刻 t に関する条件，
P: パラメタの条件
G_p: 直前の離散変化の原因となったガードのリスト
出力: t_{min}: ガードの成否が変化する最も早い時刻，
g_{list}: t_{min} において成否が変化する原子ガードのリスト

```
1:  g_list := GetAtomicBoundaryConditions(G)
2:  t_list := ∅; Map_g := ∅
3:  for g_b ∈ g_list do
4:      (sols, P) := SolveTimeEquation(g_b, P, G_p)
5:      for sol ∈ sols do
6:          t_list.add(sol, g_b)
7:      end for
8:      Map_g.insert(g_b, ConsistentAtInitialTime(g_b, G_p))
9:  end for
10: while t_list ≠ ∅ do
11:     (t_min, g_list) := PopMinimumTime(t_list, P)
12:     (Map_g, satisfied) :=
            CheckAndUpdateGuards(Map_g, g_list, G)
13:     if satisfied = true then
14:         break
15:     end if
16: end while
```

図 4　*FindMinTime* のアルゴリズム

4.2 SolveTimeEquation

SolveTimeEquation は解析解を計算できない場合，精度保証数値計算による計算を行う．この手続きは 2 つの段階に分けられる．

まず，与えられた方程式 g_b を区間ニュートン法 [15] によって解く．区間ニュートン法では，初期区間 $X^{(0)}$ から始めて，

$$X^{(k+1)} = X^{(k)} \cap N(X^{(k)})$$

という漸化式によって区間を狭めていく．なお，$N(X)$ は

$$N(X) = m(X) - f(m(X))/f'(X)$$

であり，$m(X)$ は X の中点である．$X^{(0)}$ が真の解を含んでいれば $X^{(k)}$ の区間幅は 2 次収束することが証明されている．この計算は $X^{(0)}$ が g_b の複数の解を含んでいる場合には分岐し，それぞれが真の解を含む

入力: Map_g: 直前の時刻区間における，ガードとその成否の対応表，
g_{list}: 現在時刻で成否が変化する原子ガードのリスト，
G: 全体のガード
出力: Map_g: 現在の時刻区間における，ガードと成否の対応表，
satisfied: 現在の時刻区間における G の成否

```
1: Map_prev := Map_g
2: satisfied := false
3: for g ∈ g_list do
4:     if relop(g) ∈ {'=', '≤', '≥'} then
5:         Map_g.replace(g, true)
6:     else
7:         Map_g.replace(g, false)
8:     end if
9: end for
10: if G.satisfiedBy(Map_g) then
11:     satisfied := true
12: end if
13: for g ∈ g_list do
14:     if relop(g) = '=' then
15:         Map_g.replace.(g, false)
16:     else if relop(g) = '≠' then
17:         Map_g.replace.(g, true)
18:     else
19:         Map_g.replace.(g, ¬Map_prev.getValue(g))
20:     end if
21: end for
22: if G.satisfiedBy(Map_g) then
23:     satisfied := true
24: end if
```

図 5 ***CheckAndUpdateGuards*** **のアルゴリズム**

ような複数の区間を計算することができる．

なお，もし g_b が G_p に含まれている場合，g_b がは現在の時刻区間の始点において成立するため，区間ニュートン法は始点を含む時刻区間に対しても分岐してしまう．そのような分岐は無視する必要があり，この判定を行うために *SolveTimeEquation* は引数として G_p を取っている．

次に，区間ニュートン法で得られた結果を用いてパラメタの 1 次項を保存した解を計算する．区間ニュートン法の結果は，g_b 中のパラメタを区間に置き換えて計算することで得られた単なる区間解であり，パラメタの依存関係は放棄されていることに注意されたい．このステップでは，中間値の定理に基づいてパラメタの 1 次の依存関係を保存した記号解を計算することを目的とする．このステップの仕様は次の通りである：

入力: $f(t,\vec{p}) : \mathbb{R} \times [-1,1]^n \to \mathbb{R}$,
T: 区間ニュートン法によって計算された時刻区間
出力: $T_{result}(\vec{p}) : [-1,1]^n \to \mathbb{I}$ （方程式 $f(t,\vec{p}) = 0$ の解 $t(\vec{p}) : [-1,1]^n \to \mathbb{R}$ を包含する，パラメタ化された時刻の式）

以下，$T_{result}(\vec{p})$ の計算手続きについて述べる．まず $\vec{P} \in \mathbb{I}^n$，$t \in T$，$\vec{p} \in \vec{P}$ に対し，中間値の定理より

$$\begin{aligned} f(t,\vec{p}) \in\ & f(T_m, \vec{P}_m) \\ & + (\frac{\partial f(T,\vec{P})}{\partial t}, \frac{\partial f(T,\vec{P})}{\partial p_1}, \ldots, \frac{\partial f(T,\vec{P})}{\partial p_n}) \\ & \cdot (t - T_m, p_1 - \vec{P}_{m1}, \ldots, p_n - \vec{P}_{mn}) \end{aligned} \tag{1}$$

が成り立つ．T_m は T の中点であり，$\vec{P}_m$ は各要素が $\vec{P}$ の対応する要素の中点であるようなベクトルである．以下は式（1）の証明である．

Proof. 中間値の定理より，連続かつ微分可能な関数 $h : \mathbb{R}^n \to \mathbb{R}$ と閉区間 $[a,b] \in \mathbb{I}^n$ に対し，以下を満たす $c \in [a,b]$ が存在する．

$$h(b) = h(a) + \nabla h(c) \cdot (b - a)$$

ここで，$[a,b] \subseteq I$ を満たすような区間 I を考えると，I は $h(c) \in h([a,b]) \subseteq h(I)$ を満たす．これにより，以下が言える．

$$h(b) \in h(a) + \nabla h(I) \cdot (b - a)$$

上式において，h，b，a，I をそれぞれ f，$(t,\vec{p})$，$(T_m,\vec{P}_m)$，$(T,\vec{P})$ に置き換えることで，式（1）を得る． □

ここで各パラメタの範囲は $[-1,1]$ に正規化してあるため，$\vec{P}_m$ は $\vec{0}$ と見なすことができ式（1）は次のように簡約化できる．

$$\begin{aligned} f(t,\vec{p}) \in\ & f(T_m, \vec{0}) \\ & + (\frac{\partial f(T,\vec{P})}{\partial t}, \frac{\partial f(T,\vec{P})}{\partial p_1}, \ldots, \frac{\partial f(T,\vec{P})}{\partial p_n}) \\ & \cdot (t - T_m, p_1, \ldots, p_n) \end{aligned} \tag{2}$$

式（2）の右辺を f_{mean} とし，方程式 $f_{mean} = 0$ を t について解くことで次式を得る．

$$t = -\frac{(f_{\partial p_1}, \ldots, f_{\partial p_n})}{f_{\partial t}} \cdot \vec{p} + T_m - \frac{f(T_m, \vec{0})}{f_{\partial t}} \tag{3}$$

ここで，$f_{\partial t}$ と $f_{\partial p_i}$ はそれぞれ $\partial f(T,\vec{P})/\partial t$ と $\partial f(T,\vec{P})/\partial p_i$ を表す区間値である．これらの区間を記号パラメタとして扱うと，計算コストは急速に増大する可能性がある．これを避けるため，各離散変化時刻の計算において新たに導入する記号パラメタを 1 つに限定することを考える．$\vec{p} \in \vec{P}$ を満たすすべての $\vec{p}$ について，次の性質が成り立つ．

$$\frac{(f_{\partial p_1},\ldots,f_{\partial p_n})}{f_{\partial t}} \cdot \vec{p} \in mid(\frac{f_{\partial p_1},\ldots,f_{\partial p_n}}{f_{\partial t}}) \cdot \vec{p} + [-1,1] \times \sum_{i=1}^{n} rad(\frac{f_{\partial p_i}}{f_{\partial t}}) \tag{4}$$

式（3）と式（4）より，元の方程式 $f(t,\vec{p})=0$ を包含する記号解として次の $T_{result}(\vec{p})$ が得られる．

$$T_{result}(\vec{p}) = mid(\frac{f_{\partial p_1},\ldots,f_{\partial p_n}}{f_{\partial t}}) \cdot \vec{p} + T_m + [-1,1] \times \sum_{i=1}^{n} rad(\frac{f_{\partial p_i}}{f_{\partial t}}) - \frac{f(T_m,\vec{0})}{f_{\partial t}} \tag{5}$$

ここで，$[-1,1]\times\sum_{i=1}^{n} rad(f_{\partial p_i}/f_{\partial t}) - f(T_m,\vec{0})/f_{\partial t}$ は 1 つの区間に還元することができ，そうすると式（5）の右辺はアフィン形式と見なせる．この還元後の $T_{result}(\vec{p})$ がこの手続きの最終結果となる．

Example 1. *SolveTimeEquation* に対し，以下の入力を与える．

$g_b := 2 - exp(t + p_y/1000) + p_x/1000 = 0,$

$P := -1 \le p_x \le 1 \wedge -1 \le p_y \le 1,$

$G_p := \emptyset$

まず，区間ニュートン法で実数区間 $[0.69163, 0.69466]$ を得る．次に，$T_{result}(\vec{p})$ を以下の計算によって得る．

$$\begin{aligned} T_{result}((p_x,p_y)) &= mid(\frac{f_{\partial p_x}}{f_{\partial t}})p_x + mid(\frac{f_{\partial p_y}}{f_{\partial t}})p_y + T_m \\ &+ [-1,1] \times (rad(\frac{f_{\partial p_x}}{f_{\partial t}}) + rad(\frac{f_{\partial p_y}}{f_{\partial t}})) \\ &- \frac{f(T_m,\vec{0})}{f_{\partial t}} \\ &= 5.0000 \times 10^{-4} \times p_x \\ &- 9.9999 \times 10^{-4} \times p_y \\ &+ 0.69314 \\ &+ 1.0321 \times 10^{-5} \times [-1,1]. \end{aligned}$$

なお，この例では話を簡単にするために仮数部を 10 進 5 桁に制限しているが，後述の実装は *IEEE754* の倍精度浮動小数演算に基づいたものになっている．

5 提案手法の実装

2 節に示した手法を，著者らが開発を続けている記号シミュレータ HyLaGI [14] [18] 上に実装した．HyLaGI は C++で実装されており，現在は Mathematica をバックエンドの制約ソルバとして利用している．シミュレーションアルゴリズム全体の流れは C++で実装されており，Mathematica は制約の連言の無矛盾性判定，常微分方程式の求解，離散変化時刻に関する最小化問題，算術式や論理式の変形に用いている．6 節に示す通り，バックエンドの制約ソルバの性能は理想的とは言い難いが，記号的手法と精度保証数値計算技術を組み合わせることが容易な，柔軟なプラットフォームとして利用可能である．現在の実装の健全性はブラックボックスとして使用しているバックエンドソルバの健全性にも依存している部分があるが，今後は計算手続きをさらに細かくモジュール化していくことでブラックボックスとして使用する機能の粒度を小さくすることを課題としている．本実装において，区間演算やアフィン演算およびノイズ変数の削減に関しては KV ライブラリ [10] を利用している．

HyLaGI の出力は与えられた HydLa プログラムの仕様を満たす軌道の集合である．出力される軌道は各時刻の不確定な変数値に対し，記号パラメタを用いることで表現されている．例えば図 2 の例題プログラムに対しては，`x1` の初期値に対応する記号パラメタが導入される．区間演算における区間値や，アフィン演算におけるノイズ変数も，同様に記号パラメタとして扱われる．

6 実験結果

本節では，*Enclose* と *FindMinTime* において，異なる方法で計算を行った場合の結果を比較する．比較のために 2 つの例題を使用する．

6.1 2 つの水槽の例題

1 つ目の例題は 2 節に示したものである．取りうる軌道の 1 つを図 6 に示す．横軸は時刻であり，縦軸がそれぞれの水位を示す．本モデルを，以下に示す 2 つの方法でシミュレーションした．

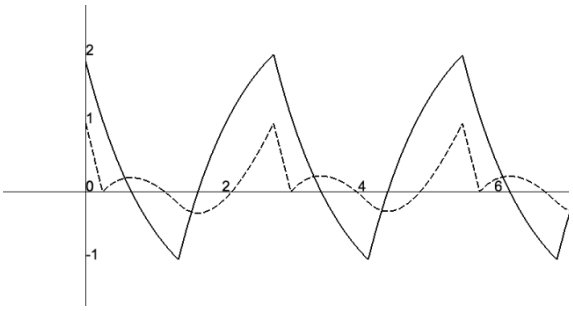

図 6　水位の軌道 （実線：x1，破線：x2）

1. 区間ニュートン法の結果をそのまま *SolveTimeEquation* の結果として用い，*Enclosure* の中でアフィン演算の代わりに区間演算を用いた（`Newton&IA` と略記）．
2. 節に示した通り，*SolveTimeEquation* で区間ニュートン法と中間値の定理を利用し，*Enclose* でアフィン演算を用いた（`Mean&AA` と略記）．なお，ノイズ変数の最大数は 6 個に制限する

図 7 は各フェーズにおける各変数値の区間幅の合計を示しており，図 8 は各ステップの実行時間を示している．横軸はシミュレーションのステップ数を表しており，Point Phase と Interval Phase を合わせて 1 ステップとしている．

図 7 において，最初の区間幅は 0.0001 である．このプログラムでは，水位の理想的なふるまいは初期水位に不確定性があっても一定のふるまいに収束していく．`Newton&IA` では不確定値の依存関係を無視しているため，区間幅が収束せず，61 ステップ後にシミュレーションが破綻する．一方，`Mean&AA` では依存関係を扱うことに成功しており，区間幅が 10^{-7} 未満に収束する．

図 8 に示す通り，`Newton&IA` の計算時間は `Mean&AA` より短い．どちらの方法でも計算時間はステップ数に対して定数オーダーに収まっており，これは *Enclose* によって数式の精度保証近似を行っているためと考えられる．もし *Enclose* を用いない場合，変数値を表す数式のサイズは増大し続け，実行時間はこのプログラムの場合線形に増大していくことになる．

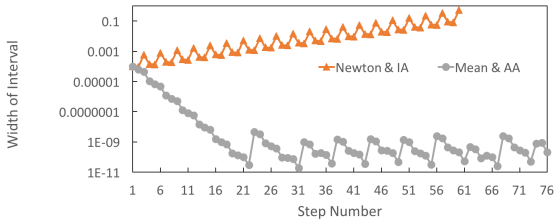

図 7　水位の区間幅

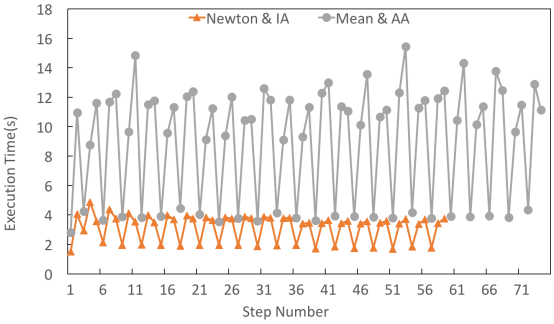

図 8　2 つの水槽の例題の実行時間

6.2　Bouncing Ball on a Sine Curve

2 つ目の例題は，2 次元空間中において正弦波上の床の上で質点が跳ねるモデルである．図 9 にプログラムを示す．本プログラムでは，`x` と `y` が質点の位置を示し，`e` が反発係数を示す．`cont`，`s`，`c` は質点のバウンドを記述するための補助変数である．図 10 に本モデルの軌道の例を示す．

図 11 に各フェーズの区間幅を，図 12 に実行時間を示す．本モデルは初期状態に不確定性を持たず，区間幅の増大は純粋に計算誤差によって発生する．`Newton&IA` と `Mean&AA` の意味は 6.1 節のものと等しい．本実験ではそれに加え，保存するノイズ変数の数を 5，9，13 と変化させる（それぞれ `d5`，`d9`，`d13` と略記する）．本モデルにおいては，`x`, `x'`, `y`, `y'`，`t` の 5 つの変数に対してアフィン形式を用いるため，ノイズ変数の最小数が 5 であることに注意されたい．

図 11 を見ると，どの条件でシミュレーションを行っても区間幅はステップ数に対して指数的に増大することがわかる．また，ノイズ変数を増やすことで増大速度が減少することもわかる．区間幅が増大して正弦波

```
INIT <=>
   x = 0 /\ x' = 0 /\ y = 10 /\ y' = 0
   /\ e = 1 /\ [](e' = 0).
FALL <=> [](cont = 1 => y'' = -10).
CONSTX <=> [](cont = 1 => x'' = 0).
SC <=>
   [](s = cos(x-)/(1 + cos(x-)^2)^(1/2)
   /\ c = 1 /(1 + cos(x-)^2)^(1/2)).
BOUNCE <=>
   []( y- = sin(x-) => cont = 0
   /\ x' = ((-e) * s-^2 + c-^2) * x'-
         + ((e+1) * s- * c-) * y'-
   /\ y' = ((e+1) * s- * c-) * x'-
         + (s-^2 + (-e) * c-^2) * y'-).

INIT, SC, FALL, CONSTX,
 [](cont = 1) << BOUNCE.
```

図 9　跳ねる質点に対応する HydLa プログラム

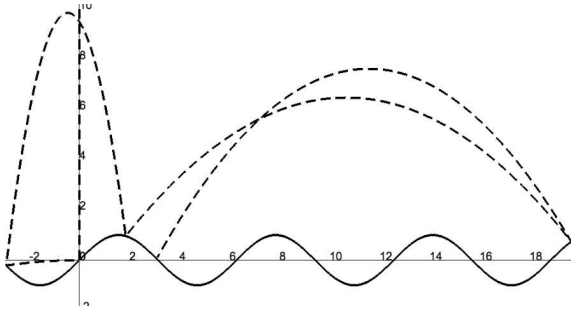

図 10　跳ねる質点の軌道（実線：床，破線：質点）

の振幅を超えると，*FindMinTime* は失敗し，それ以降のシミュレーションを行うことはできなくなる．

図 12 を見ると，ノイズ変数を増やすと実行時間も増えるが，各ステップの実行時間には何らかの上限があることがわかり，一定以上には増大していない．また水槽の例題と同じく，`Newton&IA` の計算時間は `Mean&AA` よりも少ない．

7　Related Work

本節では，ハイブリッドシステムのふるまいを厳密に計算するツールを関連研究として紹介する．Flow* [4] はハイブリッドオートマトン [7] の到達可能範囲を計算するツールである．Flow*では Taylor model に基づき，非線形微分方程式の解包囲を計算することができる．離散変化の計算に関しては，変数の初期値

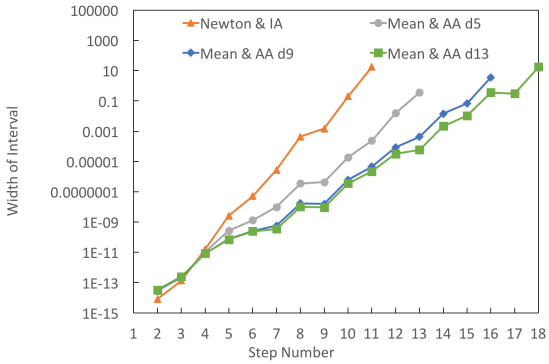

図 11　跳ねる質点の区間幅

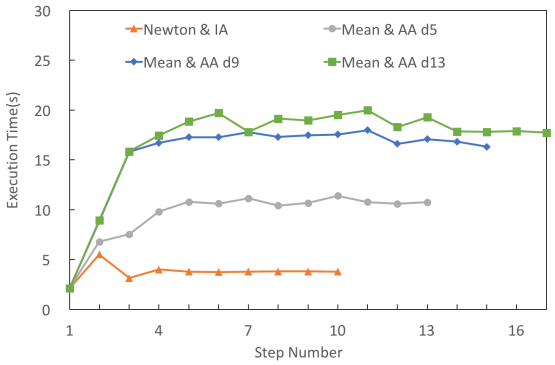

図 12　跳ねる質点の実行時間

の範囲と時刻に関して branch and prune アルゴリズムを適用する，domain contraction を利用している．Flow* は 6 節に示した例題の区間包囲を高速に計算することができ，区間幅も本手法と同様に収束するが，ガード部やインヴァリアント部に非代数的関数を記述できないため，6.2 節の例題は扱うことができない．

Acumen [20] は独自の手続き型言語を入力とするハイブリッドシステムのシミュレータである．3D オブジェクトの描画を含む豊富なグラフィック機能に加え，Zeno 時刻以降の軌道の包囲も計算できる精度保証シミュレーションをサポートしている [1]．著者らが実験したところ，Acumen の現在の実装は 2 つの水槽の例題を扱うことができたが，計算結果の区間幅は収束せず，12 回バルブを開閉したところでシミュレーションが進まなくなった．

dReach[11] は常微分方程式を扱うことができる背景理論付き SAT（Satisfiability Modulo Theories, SMT) ソルバ dReal をベースとした，ハイブリッドシステムの有界モデル検査器である．dReach はハイブリッドオートマトンの有限ステップの実行を展開し，SMT の問題に帰着する．dReach は区間制約伝搬に基づいており，与えられた性質の充足可能性を判定することができる．dReach は充足可能性の証拠として軌道の包囲も出力することができるが，その結果はパラメタを含んでいない．

KeYmaera[17] および KeYmaera X [16] は differential dynamical logic に対する定理証明器であり，記号計算を用いてハイブリッドシステムを扱うツールという点で本研究と関連している．KeYmaera は定理証明器としてのアプローチを取っているため，無限時間に関する性質も証明可能という長所があるが，証明手続きは本質的にユーザ対話型のものとなり，自動的に軌道を求めることができるシミュレータとは対照的な立場にある．KeYmaera はユーザのモデルに対する理解を支援するためにハイブリッドシステムのシミュレーション機能も有してはいるが，あくまで定理証明の補助としての機能であり，モデルに含まれる不確定性を扱ってはいない．また，KeYmaera や KeYmaera X は完全に記号計算に基づいているという点でも HyLaGI と異なっている．

8 まとめと今後の課題

本稿ではパラメタを含むハイブリッドシステムに対するシミュレーションアルゴリズムを提案した．本アルゴリズムは記号計算に対し，区間ニュートン法，アフィン演算，中間値の定理を用いた方程式求解という 3 つの技術を連携させたものである．本手法を実装したシミュレータによりパラメタ間の 1 次の依存関係を含んだ軌道の式を計算することができ，計算結果精度は通常の区間演算を用いたものより向上が見られた．

今後の課題として，広い範囲を持つパラメタの設計問題を解くために，不確定値が広い範囲を持つようなモデルを扱うことがあげられる．現在の *FindMinTime* は不確定値の範囲が狭く，パラメタの条件に関する場合分けが発生しないことを前提としている．これは，区間ニュートン法における $N(X)$ の計算において，$f(m(X))$ と $f'(X)$ の両方が 0 を含むと計算に失敗するためである．この問題を解決するため，パラメタ空間を分割して計算を進めることを計画している．また，解析解を持たない非線形微分方程式を，パラメタを含む線形微分方程式によって包囲することや，関連ツールとの詳細な性能比較も重要な課題である．

謝辞 精度保証数値計算技術に関して，貴重なご意見をいただいた柏木雅英教授に感謝いたします．本研究の一部は, 科学研究費補助金 (26280024, 15K12010) の補助を得て行った．

参 考 文 献

[1] Duracz, A., Bartha, F.A., Taha, W.: Accurate rigorous simulation should be possible for good designs, International Workshop on Symbolic and Numerical Methods for Reachability Analysis, 2016, pp. 1–10.

[2] de Figueiredo, L. H. and Stolfi, J. : Affine Arithmetic: Concepts and Applications, Numerical Algorithms, Vol. 37, Issue 1-4, 2004, pp. 147–158.

[3] Berz, M. and Makino, K. : Higher order multivariate automatic differentiation and validated computation of remainder bounds, WSEAS Transactions on Mathematics, 1998, Vol. 3, Issue 1, pp. 37–44.

[4] Chen, X., Abraham, E. and Sankaranarayanan, S. : Flow* : An Analyzer for Non-Linear Hybrid Systems. CAV 2013, LNCS 8044, Springer-Verlag, 2013, pp. 258–263.

[5] Chen, X., Schupp, S., Makhlouf, I. B., Abraham, E., Frehse, G., Kowalewski, S. : A Benchmark Suite for Hybrid Systems Reachability Analysis, In 7th NASA Formal Methods Symposium, Springer, 2015 pp. 408–414.

[6] Frehse, G., Guernic, C. L., Donzé, A., Cotton, S., Ray, R., Lebeltel, Olivier., Ripado, R., Girard, A., Dang, T., Maler, O. : SpaceEx : Scalable Verification of Hybrid Systems, CAV 2011, LNCS 6806, Springer-Verlag, 2011, pp. 379–395.

[7] Henzinger, T. : The Theory of Hybrid Automata, LICS'96, IEEE Computer Society Press, 1996, pp. 278–292.

[8] Hickey, T. J. and Wittenberg, D. K. : Rigorous Modeling of Hybrid Systems Using Interval Arithmetic Constraints, HSCC 2004, LNCS 2993, Springer-Verlag, 2004, pp. 402–416.

[9] Kashiwagi, M. : An algorithm to reduce the number of dummy variables in affine arithmetic, 15th GAMM-IMACS International Symposium on Scientific Computing Computer Arithmetic and

Verified Numerical Computations, 2012.
[10] 柏木 雅英 : kv library, http://verifiedby.me/kv/.
[11] Kong, S., Gao, S., Chen, W., Clarke, E.: dReach: δ-reachability analysis for hybrid systems, TACAS 2015, LNCS 9035, 2015, pp. 200–205.
[12] Lunze, J. : Handbook of Hybrid Systems Control : Theory, Tools, Applications, Cambridge University Press, 2009.
[13] Matsumoto, S., Ueda, K.: HyLaGI: Symbolic Simulation of Parametrized Hybrid Systems with Affine Arithmetic, 23rd International Symposium on Temporal Representation and Reasoning (TIME 2016), 2016, accepted.
[14] Matsumoto, S., Kono, F., Kobayashi, T., Ueda, K.: HyLaGI: Symbolic Implementation of a Hybrid Constraint Language HydLa, Electronic Notes in Theoretical Computer Science, Vol. 317, 2015, pp. 109–115.
[15] Moore R. E., Kearfott R.B., Cloud M. J.: Introduction to Interval Analysis, Society for Industrial and Applied Mathematics, 2009.
[16] Fulton, N., Mitsch, S., Quesel, J.D., Völp, M. and Platzer, A.: KeYmaera X: An axiomatic tactical theorem prover for hybrid systems. CADE 2015, LNCS 9195, 2015, pp. 527–538.
[17] Plätzer, A., Quesel, J.D. : KeYmaera : A Hybrid Theorem Prover for Hybrid Systems. IJCAR 2008, LNCS 5195, Springer-Verlag, 2008, pp. 171–178.
[18] HydLa, http://www.ueda.info.waseda.ac.jp/hydla/.
[19] Ueda, K., Matsumoto, S., Takeguchi, A., Hosobe, H. and Ishii, D. : HydLa : A High-Level Language for Hybrid Systems. Second Workshop on Logics for System Analysis(LfSA 2012), 2012, pp. 3–17.
[20] Zeng, Y., Rose, C., Brauner, P., Taha, W., Masood, J., Philippsen, R.., O'Malley, M. and Cartwright, R. : Modeling Basic Aspects of Cyber-Physical Systems, Part Ⅱ. 11th IEEE International Conference on Embedded Software and Systems(ICESS 2014), 2014, pp. 550–557.