Hybrid cc における区間計算に基づいた離散変化処理方式

An Interval-based Approximation Method for Discrete Changes in Hybrid cc

石井 大輔 † 上田 和紀 †† 細部 博史 †††
Daisuke ISHII Kazunori UEDA Hiroshi HOSOBE

† 早稲田大学理工学研究科
Graduate School of Science and Engineering, Waseda University
†† 早稲田大学理工学術院
Faculty of Science and Engineering, Waseda University
{ishii, ueda}@ueda.info.waseda.ac.jp
††† 国立情報学研究所
National Institute of Informatics
hosobe@nii.ac.jp

Hybrid cc は，連続および離散の振る舞いからなるハイブリッドシステムの簡潔な記述と正確な制御を可能とする制約プログラミング言語である．現状の Hybrid cc 処理系には，ハイブリッドシステム中の離散変化の処理方式に起因し，計算を正確に行うことができなかったり，プログラム中に実装依存の記述が必要になったりするという問題があった．こうした問題を踏まえ，本論文では区間計算に基づき離散変化の解を求めるための処理方式を提案する．提案方式により，解を厳密に包む区間解を高精度の区間幅で得ることができる．Hybrid cc において区間値に基づいた簡潔な記述を行うとともに，ハイブリッドシステムをロバストかつ高精度に制御することが可能となる．

1 はじめに

ハイブリッドシステムは，連続と離散の振る舞いからなる系を表す概念である．例として，エンジンルームの温度変化を測定しながらエンジンの制御を行うデジタル回路を搭載した自動車が挙げられる．ハイブリッドシステムにおいて，連続変化を表す微分方程式と離散変化の状態空間とが複雑になると，効率のよいモデリングや制御が困難になる．

制約プログラミングに基づいた宣言型言語である **Hybrid cc** (Hybrid concurrent constraint programming) [6, 5] は，ハイブリッドシステムの高級な開発ツールとして提案され，ハイブリッドシステムの簡潔な記述および確実な制御を可能とする．以下に Hybrid cc による記述例を示す．

```
y = 10, y' = 0,              // 初期値
hence {
  cont(y),                   // 連続値の宣言
  if y > 0 then y'' = -10,   // 自由落下
  if y = 0 then
    y' = -0.5 * prev(y')     // 衝突
}
```

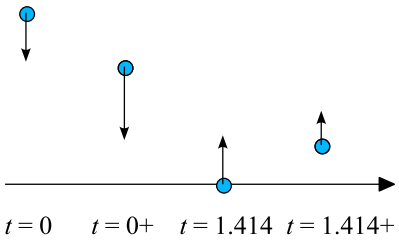

図 1: ハイブリッドシステムの例

この例は，図 1 に示すような質点が重力に従って落下し地面で跳ね返る動作をする簡単なハイブリッドシステムを表している．

Hybrid cc の現在の実装では，計算誤差が厳密に扱われていないため，諸々の問題が生じる．図 2 (a) は上記プログラムの実行結果を示したものである．時間が経過し質点が地面に近づいた際に，質点が想定外の挙動を示す．この原因は，条件 $y = 0$ が処理系内部において $y = 0 \pm \epsilon$ の形で扱われており (計算誤差を吸収するため)，y と地面との距離が微小となった際に，離散変化が微小時間内で連続して発生してしまうためである．プログラム中の 2 番目の制御式

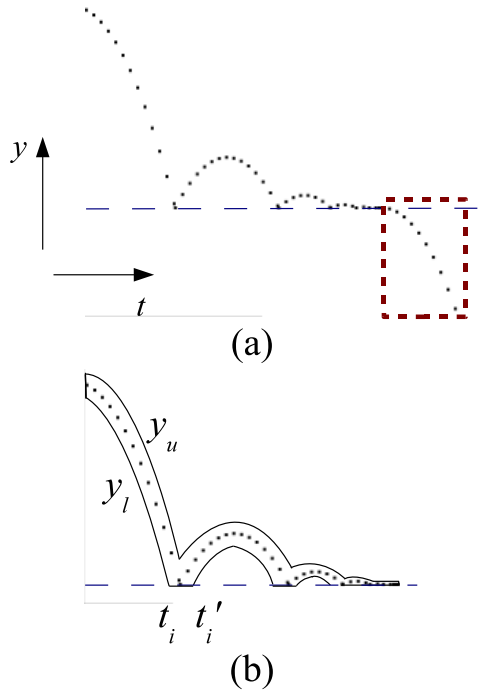

図 2: (a) 実行結果における想定外の挙動，(b) 区間計算に基づく近似解

を以下のような記述に書き換えることにより，この問題を避けることができる．

```
if y = 0 then                    // 衝突
  if (prev(y') > -0.000001)
    then always y' = 0
    else y' = -0.5 * prev(y')
```

このような実装に依存した冗長な記述を行うことにより，Hybrid cc の記述性が損なわれてしまう．

本論文では，区間計算に基づきハイブリッドシステムの近似解を得るための処理方式を提案する．図 2 (b) に示すような軌道を厳密に包む区間解を，高精度の区間幅で得ることを目標とする．提案方式により，区間値に基づき Hybrid cc における記述性が向上するとともに，区間計算に基づきハイブリッドシステムの正確な制御を行うことが可能となる．

以下に本論文の構成を示す．2 節では Hybrid cc の概要を述べる．3 節では Hybrid cc の既存実装における問題点を説明する．4 節では以降の議論で必要となる区間計算の基礎について述べる．5 節では提案方式の概要を述べ，6，7，8 節において提案方式を構成する各アルゴリズムについて説明する．9 節では提案方式に関する実験結果を示す．

2 Hybrid cc の概要

2.1 Hybrid cc 言語

Hybrid cc はハイブリッドシステムを記述するための制約プログラミングの枠組みである．Hybrid cc に関する形式的議論については [6] を参照されたい．以下に Hybrid cc プロセス (以下構文カテゴリ A で示す) の基本的な構文を示す．

c	制約 c を制約ストアに追加する (tell)
if c **then** A	c が成り立つならば (ask)，A を導出する
A, A	並列合成
hence A	現時点より後のすべての瞬間で A を実行する

Hybirid cc が扱う制約の構文定義と意味定義の組を**制約システム**と呼ぶ．現状の Hybrid cc は，以下に挙げる 3 種類の制約を扱う制約システムを備える．

- 連続算術制約
- 非算術制約
- ブール制約

制約 $a(x_1, \ldots, x_k)$ は，変数 $x_i, 1 \leq i \leq k$ について成り立つ関係を表す．たとえば連続算術制約 x^2+y^2-1 は x と y を 1 と -1 の間に束縛する．また，Hybrid cc では変数の時間による微分値を常微分方程式により定義することができる (例: $x' = x$)．Hybrid cc における計算は制約を**制約ストア**に追加していくことでなされる (この追加操作を tell と呼ぶ)．**if** c **then** A という条件式では，制約ストアにおいて **離散条件** (離散変化を引き起こす条件) c が帰結されるかどうかをチェックし，帰結されるならば A を導出する (このチェック操作を ask と呼ぶ)．また並列合成式の各要素は，必要に応じてサスペンドされ同期が取られる．Hybrid cc の重要な構文である **hence** により，時間に関する常微分方程式に基づいた制御機能が提供される．式 **hence** A は，現在時点を端としてそれより後において永久に実行することを意味する．たとえば，**hence** $x' = 1$ は x を現在のシミュレーション時刻に等しくする．

2.2 Hybrid cc 言語解釈系の実装

本論文における提案方式は，Hybrid cc 言語解釈系の既存実装に基づき設計した．既存実装のシステム構成図を図 3 に示す．既存実装の詳細については [1] を参照されたい．ここでは，解釈処理の概要を示

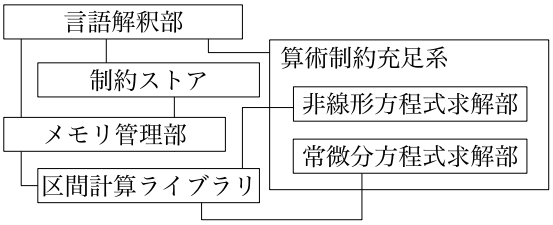

図 3: Hybrid cc 言語解釈系のシステム構成

す．Hybrid cc プログラムの解釈処理は，**点フェーズ**と**区間フェーズ**を繰り返し行うことでなされる．

1. $t = 0$ の点フェーズから処理を開始する．
2. 点フェーズでは，プログラムの簡約処理を行う．その際，制約の tell および ask 処理や，**算術制約充足系** (次節を参照) による算術制約の伝搬処理が行われる．点フェーズの処理は，簡約処理および制約伝搬がすべて完了した安定状態に至ると終了する．
3. その後，処理は区間フェーズに移る．点フェーズにおいて式 **hence** A から簡約された表現 A を処理する．点フェーズ同様に，各式が安定状態に至るまで簡約を行う．これにより，連続的に評価すべき算術制約の集合を決定する．
4. 3. で求められた連続算術制約が離散変化を起こすまで，常微分方程式系の求解処理を行う．
5. 制約ストアにおいて矛盾が発生するか，処理すべき式がなくなったら，計算を終了する．その他の場合は 2. へ移る．

2.3 算術制約充足系

非線形方程式求解部と常微分方程式求解部からなる算術制約充足系の実装について述べる．これらの充足系では，不確定な情報をパラメタ (区間値) により吸収するため，区間計算が用いられている．

非線形方程式求解部は，$f(x) = 0$ のような implicit な形の算術制約を扱い，制約中の変数区間の枝刈りを行うことにより区間解を得る．実装では，枝刈りの手法として (1) indexical を使った手法，(2) 区間の区分け，(3) Newton-Raphson 法，(4) シンプレックス法が使われている．

常微分方程式求解部は，可変刻み幅を用いた数値解法である Runge-Kutta-Fehlberg 法に基づき区間解を得る．実装では以下の 2 つの変更を施している．

- 各ステップの積分処理を行うたびに方程式の伝搬を行う．これにより任意の方程式を explicit な形に変形して処理を進めることができる．
- 求解部は制約ストア内の制約の状態が変化したことを示す **breakpoint** に至ったところで処理を終了する．breakpoint を超えてステップ処理がなされた overshoot 状態に至った場合には，バックトラックを行う．現状では，バックトラックは単純な線形補完法を用いて行っている．

3 現状の Hybrid cc における問題点

我々は，物理シミュレーションやアニメーションを扱うインタラクティブシステムの開発に Hybrid cc を応用することを考えている．開発システムの例として，初等物理に基づき玉の動作を制御するビリヤード台のシミュレーションが挙げられる．任意個の玉が移動するとともに，他の玉やテーブル端と衝突して跳ね返る．この例では，玉の移動軌跡や範囲を正確に求めたり，ポケットの位置から軌跡を逆向きに計算したりすることが必要になる．

2.2 節や 2.3 節で述べた処理系に基づき，このようなシステムの実装を試みたが，1 節で述べたような実装に起因する問題が生じた．既存実装には以下の制限がある．

- 区間計算処理が **if** c **then** A 式によって表される離散変化を適切に処理していない．たとえば 1 節のプログラム中の変数 y に区間値を代入しても，うまく動作しない (離散条件が無視され質点が地面に突入してしまう)．
- 1 節で見たように，離散変化における breakpoint の検出処理が，丸め誤差を吸収するために定数 ϵ を使った ad hoc な実装になっている．このため，区間計算を採用しているにも関わらず解の精度が保証されない．

上記を踏まえ，離散変化の真解を区間により厳密に包むことが可能な処理方式を提案する．また，提案方式は既存の算術制約充足系と連携動作が可能な設計にする．

4 区間計算の基礎

4.1 基本概念

本論文で用いる区間計算 [12] の基本概念を準備する．$\mathcal{F}$ により浮動小数点数の集合を表す．$\mathcal{I}$ により境界が $\mathcal{F}$ に含まれる $\mathbb{R}$ 上の区間の集合を表す．I を

$\mathcal{I}$ 中の区間とするとき，$\mathrm{lb}(I)$ で I の下端を，$\mathrm{ub}(I)$ で上端を，$\mathrm{w}(I)$ で直径を表す．$\mathcal{D}$ により境界が $\mathcal{F}$ に含まれる $\mathbb{R}^n$ 上の box の集合を表す．D は $\mathcal{D}$ 中の box を表す．r を実数，A を $\mathbb{R}^n$ の部分集合とするとき，$\bar{r}$ を $\mathcal{I}$ 中で r を含む最小区間，$\square A$ を $\mathcal{D}$ 中で A を含む最小の box とする．g が関数のとき，G により g の区間拡張を表す．

4.2 区間計算に基づく常微分方程式の求解

これまでに，区間計算および制約伝搬を用いた常微分方程式の求解手法 [2, 4, 11, 3] が提案されてきた．ここでは [4] から常微分方程式に関する諸概念を導入する．初期値 $u(t_0) = u_0$ を持つ常微分方程式系 $\mathcal{O}$ の解を，関数 $s^*(t) : \mathbb{R} \to \mathbb{R}^n$ により与える．ここで，$s^*(t)$ は $\mathcal{O}$ を満たすとともに，初期条件 $s^*(t_0) = u_0$ が成り立つ．また，$\mathcal{O}$ の解を関数 $s(t, u, t') : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ によって表すこともできる ($s(t_0, u_0, t) = s^*(t)$ とする)．s を $\mathcal{O}$ の解とするとき，すべての $t \in [t_0, t_1]$ について $s(t_0, D, t) \subseteq B$ が成り立つような B を，s の $[t_0, t_1]$ での bounding box と呼ぶ．

補題 1 (常微分方程式の解の連続性，[4]) f が開集合 $E \subseteq \mathbb{R} \times \mathbb{R}^n$ において連続とし，各 $(t_0, u_0) \in E$ について初期値問題 $\{u' = f(t, u), u(t_0) = u_0\}$ は，唯一の解 $u(t) = s(t_0, u_0, t)$ を持つとする．$T(t_0, u_0)$ を $u(t) = s(t_0, u_0, t)$ が存在する E 中の最大区間とする．このとき，s は $\{(t_0, u_0, t) | (t_0, u_0) \in E, t \in T(t_0, u_0)\}$ において連続である．

5 提案方式の概要

提案方式の概要を図 4 の RunHcc アルゴリズムとして示す．提案方式では既存の Hybrid cc 処理系の基本的な処理方式を採用するが，連続変化を計算する常微分方程式系の求解処理を，Trace アルゴリズムおよび SolveDiscreteCond アルゴリズムにより置き換える．

図 5 に提案方式の処理を示す．ハイブリッドシステムの連続変化の区間解を，Trace アルゴリズムが逐次的に求める．Trace はステップ時間ごとに離散条件 c による状態変化を判定し，状態変化が発生した場合，状態変化が発生した時間区間 $T_d : [t_d, t'_d]$ と，T_d での変数値 (およびその微分値) を包囲する box B_d とを求める．次に，SolveDiscreteCond アルゴリズムが，T_d と B_d を組にした box $T_d \times B_d$ を，

1: **repeat**
2: 点フェーズを処理
3: 区間フェーズを処理
4: 常微分方程式の求解を行う
5: Trace
6: SolveDiscreteCond
7: **until** 制約ストアで矛盾が発生した ∨ 実行キューが空になった

図 4: RunHcc アルゴリズム

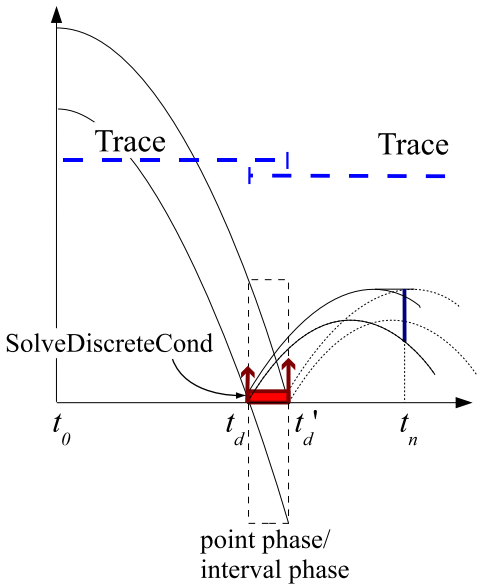

図 5: ハイブリッドシステムの区間解の求解

離散条件 c に基づき刈り取る処理を行う．その後提案方式は，SolveDiscreteCond の結果をもとに点フェーズと区間フェーズの処理を行い，新たな常微分方程式系を構築する．離散変化が時間軸上の区間で起きるため，その後の区間フェーズは初期時間を区間値とした常微分方程式系になる．Trace ではこのような常微分方程式系の区間解を求めることができる．さらに，離散変化の際に上記 $T_d \times B_d$ を複数の box に分割し，それぞれの box について以降の処理を行う区分け処理により，さらに精度のよい区間解を得ることができる．以下の節では，これらの処理について説明していく．

6 離散条件

本論文では，連続関数 $fd : \mathbb{R}^n \to \mathbb{R}$ を考え，$fd = 0$ という非線形等式で表される算術制約 c を

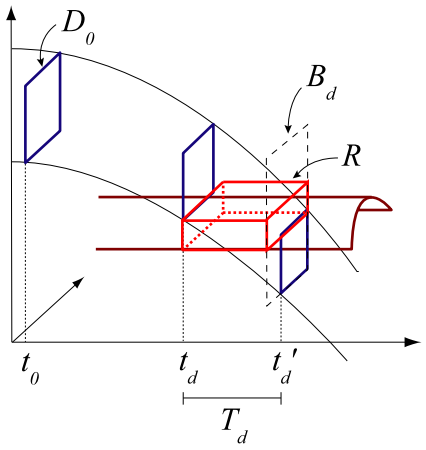

図 6: 離散条件の求解処理

離散条件とする．式の変形を行ったりスラック変数を用いたりすることにより，不等式を含む他の形式の制約も扱うことができる．

提案方式では，Trace がステップごとに求解を行いながら不等式 fd $\leqslant 0$ の符号の反転をチェックすることにより，離散条件 $c : fd = 0$ が成立する時間区間 T_d を求める．T_d と T_d における常微分方程式系の解の bounding box B_d との積 $T_d \times B_d$ を，離散条件の解として求める (Trace は区間値 T_0 を初期時点とした常微分方程式系を扱うが，ここでは $T_0 = 0$ の場合について述べる)．図 6 に提案方式による処理を示す (図中の縦軸および奥行き軸は変数値の box を与える)．

本論文では，3 節で述べた開発システムに応じた前提条件により扱う問題を限定した上で，精度のよい処理方式を提案する．扱う問題としては，たとえば等加速度運動をする box 値が凸曲面に衝突する例を考える (凹曲面の場合も同様に考えることができる)．曲面に向かう box の先端とそれに対する曲面の極値とが最初に衝突してから (時点 t_d)，box の逆端の 1 点が曲面上で衝突するまで (時点 t_d') が，T_d となる．以下に，提案方式のための前提条件と定理を示す．

前提 1 $\mathcal{O}$ を $u' = f(t, u)$ で表される常微分方程式系とする (f は連続な関数とする)．s を $\mathcal{O}$ の解とする (存在と一意性が保証されているとする)．U を $\mathbb{R}^n$ における開集合，D_0 を U 上の box とする．$fd(u)$ を U において連続な関数とする．また，$fd(u)$ は時間について不変であるとする．T を $\mathbb{R}$ における 開集合，T_d を T 上の box とし，T_d において $s(t)$ は

Require: 離散条件 c
Require: c が成立する時間区間 T_d
Require: T_d における bounding box B_d
Ensure: box R
1: $R' :=$ branch-and-prune $(c, T_d \times B_d)$
2: return $\square R'$

図 7: SolveDiscreteCond アルゴリズム

単調増加または単調減少であるとする．

定理 1 前提 1 が成り立つとする．ある $t \in T$ において，すべての $u \in D_0$ について $fd(s(t_0, u, t)) > 0$ が成り立つとする．また，ある $t' \in T$ において，すべての $u \in D_0$ について $fd(s(t_0, u, t')) < 0$ が成り立つとする．上記のもとで，すべての $u \in D_0$ について離散条件 $fd(s(t_0, u, t_d)) = 0$ を満たす $t_d \in T_d$ が，$t' \leqslant t_d \leqslant t$ の範囲でただ 1 つ存在する．

証明. 任意の u について，$fd(s(t_0, u, t)) > 0$ を満たす t と，$fd(s(t_0, u, t')) < 0$ を満たす t' とが存在する．fd は連続なので，$fd(s_d) = 0$ を満たす s_d が $s(t_0, u, t') < s_d < s(t_0, u, t)$ の範囲に存在する．s は補題 1 から連続であり，かつ単調に増加あるいは減少するので，$s(t_0, u, t_d) = s_d$ を満たす t_d が，$t' \leqslant t_d \leqslant t$ の範囲でただ 1 つ存在する． ∎

6.1 SolveDiscreteCond アルゴリズム

Van Hentenryck ら [7] により，区間計算に基づき非線形方程式の近似区間解を複数の box の和集合として求める **branch-and-prune アルゴリズム**が提案された．branch-and-prune アルゴリズムは，入力する box から，真解に包含される box の集合と，真解を包含する box の集合とを，指定した精度内で得ることができる．branch-and-prune アルゴリズムに box $T_d \times B_d$ を入力して離散条件の解を包む box R を求める SolveDiscreteCond アルゴリズムを図 7 に示す．

7 離散変化とその後の常微分方程式系

離散変化の解を得た後，点フェーズと区間フェーズの処理を行う．その結果得られた常微分方程式系を，提案する Trace アルゴリズムにより処理する．

7.1 点フェーズおよび区間フェーズ

点フェーズと区間フェーズの処理において，離散変化後の常微分方程式系 $\mathcal{O}$ を構築する．まず処理に入る前に，R 中の時間成分 T_d を変数 T_0 として保存しておく．現状の Hybrid cc 処理系では，変数値 (とその微分値) を区間値として点フェーズの処理を行うことが可能である．提案方式では，まず R 中の時間成分 T_d を下端 $\mathrm{lb}(T-d)$ に変換した box R' を作成し，R' に関して点および区間フェーズの処理を行う．以下，この結果得られた box を D_0 により表す．

7.2 Trace アルゴリズム

上記の区間フェーズの結果得られた，初期時点 T_d と初期値 D_0 を持つ常微分方程式系 $\mathcal{O}$ の求解を行う．初期時点として区間値を扱うにあたり，まず離散変化後 t 秒後の $\mathcal{O}$ の解を考えてみると，T_0 中のどの時点で離散変化が起こったかに関わらず，D_0 を初期値とした t 秒分の計算により解が求まる．次に，離散変化前のある時点 (たとえば以前の常微分方程式系の初期時点) から t' 秒後の解を考える．この場合，区間 $[t'-\mathrm{ub}(T_0), t'-\mathrm{lb}(T_0)]$ に含まれる t での上記計算結果の集合が解となる．後述する既存手法により，$\mathcal{O}$ の解曲線を包む bounding box の列を得ることができる．Trace アルゴリズムの処理は，この解曲線の包囲を求めるとともに，T_0 の幅 $\mathrm{w}(T_0)$ 分の bounding box を切り出すような sliding window を動かしていく処理として考えることができる．

図 8 に Trace アルゴリズムを示す．7 行目において，$\mathcal{O}$ の求解を $[t,t+h]$ の 1 ステップ分行う．現状の Hybrid cc 処理系は区間計算を用いた常微分方程式系の求解をサポートしている (ただし厳密な求解のためには丸め方向のコントロールなどの改良が必要)．また既存手法 [4, 11, 3] を用いれば，より高い精度の求解を行える．前述したように，Trace では幅 $\mathrm{w}(T_0)$ の bounding box を求める．そのために，幅 $\mathrm{w}(T_0)$ 分の複数の bounding box を格納するバッファ Q を用意する (4 行目)．図 9 に，複数の bounding box を Q に格納し，その和として解 D_1 を得る処理を示す．$\mathcal{O}$ の解の bounding box は [4] の手法により得ることができる．8 行目で区間 $[t,t+h]$ の bounding box を求め，9 行目で Q に格納し，11 行目で和を求めている．13 行目では，各ステップにおいて D_1 について離散条件 $c: fd \leqslant 0$ の判定を行い，c が成立する box $T_d \times B_d$ が求まったならば処理を終了する．

Require: 常微分方程式系 $\mathcal{O}$
Require: 初期時間 T_0
Require: 初期値 D_0
Require: 離散条件 c
Require: ステップ幅 h
Ensure: box $T_d \times B_d$

1: $T_d := \emptyset$; $B_d := \emptyset$
2: $t := \mathrm{lb}(T_0)$
3: $D := D_0$
4: $Q :=$ サイズ $\mathrm{w}(T_0)/h$ のバッファを初期化
5: $b := false$
6: **loop**
7: $D := t$, D, h より，$\mathcal{O}$ の解を 1 ステップ分計算する
8: $B :=$ bounding box ($\mathcal{O}$, D, $[t,t+h]$)
9: B を Q の末尾に追加
10: Q の先頭から余分になった要素を除去
11: $D_1 := \square\{\text{elements in } Q\}$
12: propagate
13: **if** D_1 に関し c の状態が変化した **then**
14: **if** b **then**
15: $T_d := \square\{T_d, \overline{t+h}\}$
16: $B_d := D_1$
17: return $T_d \times B_d$
18: **else**
19: $T_d := \bar{t}$
20: $B_d := D_1$
21: $b := true$
22: **end if**
23: **end if**
24: $t := t+h$
25: **end loop**

図 8: Trace アルゴリズム

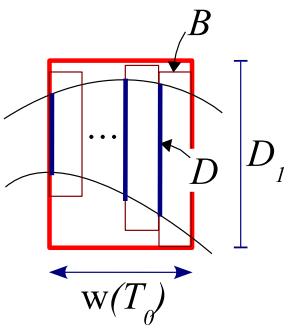

図 9: 時間のパラメタを持つ常微分方程式系の近似解

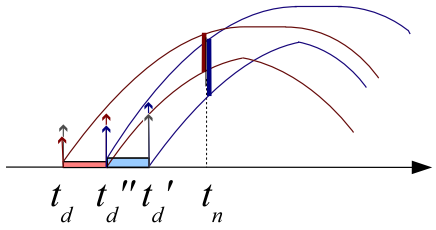

図 10: 1 回の区分けによる求解

8 区分け処理

区分け処理として，離散変化の際に時間区間 T_d を適当な時点 t_d'' (例: 中間値) で区分けし，区分けした区間ごとに以後の処理を行う．図 10 に 1 回の区分けを行う処理を示す．離散条件が判定された T_d 中の時点 t'' を選択し，区間 $[t_d, t_d'']$ および $[t_d'', t_d']$ について bounding box を求め，それぞれについて SolveDiscreteCond の処理を行う．以降も区分け結果に基づき処理を続ける．

区分け処理により，離散変化時点 T_d に含まれる区間とそれに基づく変数値との対応を，高い精度で保持することができる．また，区分け処理結果の和を計算することにより，初期時点からの到達範囲をより高い精度で得ることができる．区分けを複数回行うことにより，さらに精度を改善できる．本論文では離散条件が時間に依存しないことを前提としているが，これを緩和すればさらなる効果が得られると考えられる．

9 実験結果

提案方式を Hybrid cc の既存実装を利用して擬似的に計算する実験を行った．本節では，実験結果を示すとともに，単純な区間計算結果と提案アルゴリズムによる計算結果の比較を行う．実験は 1 節で示した例題を区間に拡張したものを用いて行った．以下に実験で用いた Hybrid cc プログラムを示す．

```
y = [10, 11], y' = [0, 1],
hence {
  cont(y),
  if y > 0 then y'' = -10,
  if y = 0 then
    y' = -0.5 * prev(y')
}
```

実験では 1 回の跳ね返り ($T_d = [1.414214, 1.586607]$ で起こる) について計算を行った．ただし，本実験では丸め方向の制御を行わずに区間計算を行っている．また SolveDiscrete-Cond として，理論値 $[0, 0]$ を設定する擬似的な処理を用いた．表 1 の各列に，(1) 刈り取りを行わずに計算を行った場合，(2) SolveDiscreteCond による刈り取りを行った場合，(3) 離散変化の時間区間を，時点 $t = 1.5$ で 1 回区分けした場合の計算結果，および (4) 理論値を示した．計算結果に理論値が含まれるとともに，刈り取りおよび区分け処理により区間幅が狭められていることが確認できた．

10 関連研究

制約プログラミングに基づきハイブリッドシステムのモデリングや検証を行う手法が提案されている．Hickey ら [10] により，実数と関数との間の解析的関係を記述するための制約論理プログラミング言語である CLP(F) [9] に基づいた手法が提案された．また，CLP(F) では区間計算による処理がサポートされている．上記システムでは，離散変化処理の際に区間幅の増大を抑える処理が行われていない．また Hybrid cc と比較すると，CLP(F) は基本構文として Prolog の構文を採用しており，時間概念を伴った制御やプログラムの部品化に関する表現力が劣る．Urbina [15] は CLP($\mathcal{R}$) 言語を用いた手法を提案した．この手法では区間計算を用いていないため，解が厳密に計算されない．また，扱える数値制約は線形制約に限られる．

ハイブリッドシステムの安全性検証のために，軌道の範囲を区間値により表現する手法が提案されている．Stursberg ら [14] は，連続状態空間をグリッド状の box 表現により近似する手法を開発した．近似に基づき，到達可能な状態集合の安全性などを判定する．Henzinger ら [8] や Ratschan ら [13] は，上記手法を発展させ，状態空間から安全でない軌道や不要な軌道を制約充足に基づき刈り取る手法を提案した．これらの手法では，ハイブリッドシステムをハイブリッドオートマトンに基づきモデリングするため，Hybrid cc に較べると記述性が低い．また，状態空間の刈り取りのために，box の境界における軌道の流れに関する制約を与える必要がある．本論文の提案手法が安全性検証に応用可能かどうかは検討する必要がある．

t	(1) 単純計算	(2) 刈り取り実施	(3) 1 回の区分け	(4) 理論値
0.000000	[10.00000 , 11.00000]	[10.00000 , 11.00000]	[10.00000 , 11.00000]	[10.00000 , 11.00000]
1.586607	[-2.586607 , 3.633218]	[0.000000 , 1.219005]	[0.000000 , 1.144353]	[0.000000 , 1.070408]
2.000000	[-0.724642 , 5.345549]	[1.861965 , 2.931335]	[2.039282 , 2.677670]	[2.218295 , 2.426407]

表 1: 実験結果

11 まとめと今後の課題

本論文では，ハイブリッドシステムにおける離散変化の解の区間包囲を求めるための処理方式を提案した．今後，処理の詳細を検討した後，Hybrid cc の既存実装に基づき提案方式の実装を行う予定である．提案方式を実現することにより，Hybrid cc で記述されたハイブリッドシステムにおける軌道全体を区間で包むことが可能となる．

今後の課題として，区間解中の部分のみが離散変化を起こし，複数状態に分岐が発生する場合への対応が挙げられる．たとえば物体が矩形の角で跳ね返るという例では，複数の離散変化が同時に起こることを考慮しなければならない．さらに，上記物体の区間値はちょうど矩形の頂点に衝突する点を含んでおり，予想外の結果をもたらす可能性がある．こうした特異点の対処方法についても考慮する必要がある．

謝辞

提案方式の実験を手伝っていただいた，上田研究室所属の大野善之，西澤亮太，西村光弘の各氏に感謝いたします．

参考文献

[1] B. Carlson and V. Gupta. Hybrid cc with interval constraints. In *HSCC*, 1998.

[2] G. F. Corliss. Guaranteed error bounds for ordinary differential equations. In *Theory of Numerics in Ordinary and Partial Differential Equations*. Oxford University Press, 1994.

[3] J. Cruz and P. Barahona. Constraint satisfaction differential problems. In *CP 2003*, volume 2833 of *LNCS*, pages 259–273. Springer Verlag, 2003.

[4] Y. Deville, M. Janssen, and P. Van Hentenryck. Consistency techniques for ordinary differential equations. In *CONSTRAINT*, volume 7 (3/4), pages 289–316, 2002.

[5] V. Gupta, R. Jagadeesan, and V. A. Saraswat. Hybrid cc, hybrid automata and program verification. In *Hybrid Systems III*, volume 1066 of *LNCS*, pages 52–75. Springer Verlag, 1996.

[6] V. Gupta, R. Jagadeesan, V. A. Saraswat, and D. Bobrow. Programming in hybrid constraint languages. In *Hybrid Systems II*, volume 999 of *LNCS*. Springer Verlag, 1995.

[7] P. Van Hentenryck, D. McAllester, and D. Kapur. Solving polynomial systems using a branch and prune approach. In *SIAM Journal on Numerical Analysis*, volume 34 (2), pages 797–827, 1997.

[8] T. A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi. Beyond HyTech: Hybrid systems analysis using interval numerical methods. In *HSCC 2000*, volume 1790 of *LNCS*. Springer Verlag, 2000.

[9] T. J. Hickey. Analytic constraint solving and interval arithmetic. In *POPL '00 ACM SIGACT-SIGPLAN Symposium*, 2000.

[10] T. J. Hickey and D. K. Wittenberg. Rigorous modeling of hybrid systems using interval arithmetic constraints. In *Hybrid Systems: Computation and Control*, volume 2993 of *LNCS*, pages 402–416, 2004.

[11] M. Janssen, P. Van Hentenryck, and Y. Deville. A constraint satisfaction approach for enclosing solutions to parametric ordinary differential equations. In *IJCAI-01*. Morgan Kaufmann, 2001.

[12] R. E. Moore. *Interval Analysis*. Prentice-Hall, 1966.

[13] S. Ratschan and Z. She. Safety verification of hybrid systems by constraint propagation based abstraction refinement. In *Hybrid Systems: Computation and Control*, volume 3414 of *LNCS*. Springer Verlag, 2005.

[14] O. Stursberg, S. Kowalewski, I. Hoffmann, and J. Preusig. Comparing timed and hybrid automata as approximations of continuous systems. In *Hybrid Systems*, volume 1273 of *LNCS*, pages 361–377. Springer Verlag, 1997.

[15] Luis Urbina. Analysis of hybrid systems in CLP(R). In *CP 1996*, volume 1118 of *LNCS*. Springer Verlag, 1996.