1E3-3

ハイブリッド制約言語 HydLa 処理系における 数式処理と区間計算を組み合わせたシミュレーション実行

Simulation with formula manipilation and interval arithmetic by HydLa implementation

和田努 *1 松本翔太 *1 上田和紀 *1
Tsutomu WADA Shota MATSUMOTO Kazunori UEDA

*1早稲田大学大学院基幹理工学研究科情報理工・情報通信専攻
Gradute School of Fundamental Science and Engnnering, Waseda University

Hybrid Systems are dynamical systems with both continuous and discrete dynamic behavior. HydLa is a programing language for modeling Hybrid Systems. HyLaGI, an implementation of HydLa, simulates models without calculation error by using formula manipulation. Hybrid systems with complicated equations are difficult to simulate with formula manipulation because it is hard to calculate the time of discrete changes. To solve this problem, we design and implement an algorithm that uses the interval Newton method, which can compute interval numerical solutions containing the exact solution. This algorithm selects an interval suitable for HydLa simulation from solutions computed by the interval Newton method, and guarantees that there is a unique exact solution in the selected interval.

1. はじめに

ハイブリッドシステム [2] とは，連続変化と離散変化の両方を含んだ動的システムであり，制御工学や生命工学などの幅広い分野で応用可能な概念である．これらの分野において，複雑なシステムの挙動の解析や検証の需要がある．そのため，複雑なシステムを記述することのできるモデリング言語と，モデリングされたシステムの解析および検証を高精度に行えるシミュレータは非常に有用である．

ハイブリッドシステムモデリング言語 HydLa[7] は，システムの挙動を時相論理演算子と微分方程式を含む制約で記述する．HydLa では制約間の優先関係を記述することで，KeYmaera[4] に代表される手続き型の記述やハイブリッドオートマトン [8] に比べて，システムの全状態を列挙する必要がなく，簡潔なモデリングを可能としている．HydLa の処理系 HyLaGI[3] は数式処理を用いることで，浮動小数点演算によって生じる誤差のないシミュレーションを実現している．

しかし HyLaGI の数式処理による枠組みでは，シミュレーション中に離散変化条件を表す方程式が複雑化すると，制約を解くことができなくなる場合がある．この問題は物体が単振動と自由落下を繰り返す単純なモデルでも起こりうる．このため本研究では，数式処理で扱うことができないモデルの厳密なシミュレーションを目的とし，数式処理シミュレーションに区間ニュートン法 [5] を組み込む手法を考案，実装した．本手法では，単に方程式の解を区間ニュートン法によって求めるだけでなく，HyLaGI のシミュレーションに必要な解のみを効率的に計算する．更に，計算機上での区間ニュートン法では求められた数値区間中に真の解が複数存在する可能性があるが，本研究では真の解が唯一存在することを保証するための手法を実装することで，シミュレーションの厳密性を高めた．

連絡先: 和田努，早稲田大学大学院基幹理工学研究科情報理工・情報通信専攻，〒 169-8555 新宿区大久保 3-4-1 63 号館 5 階 02 号，03-5286-3340，tsutomu(at)ueda.info.waseda.ac.jp

2. HydLa の処理系 HyLaGI

HyLaGI は HydLa によって記述されたハイブリッドシステムのシミュレータであり，入力された HydLa プログラムの各変数値の軌道を出力する．HyLaGI は数式処理システム Mathematica をバックエンドとして採用しており，浮動小数点演算によって生じる誤差のないシミュレーションが可能となっている．

2.1 HyLaGI の実行アルゴリズム

本節では，HyLaGI の実行アルゴリズムについて説明する．詳しいアルゴリズムの内容は文献 [3] によって与えられているため，ここでは本研究において特に重要な部分についてのみ述べる．本実行アルゴリズムでは離散変化を計算する Point Phase(PP) と，連続変化を計算する Interval Phase(IP) を繰り返すことで，モデル中の変数の軌道を計算していく．

HydLa では，条件が成り立った時のみ有効になる条件つき制約を記述することができ，条件つき制約の条件部分のことをガード条件と呼ぶ．HydLa モデルの離散変化はガード条件の成否の変化によって引き起こされるため，各 IP では次の離散変化時刻を求めるために各ガード条件の成否が変化する時刻を計算し，その中で最小のものを選ぶ必要がある．

2.2 例：惑星トンネル

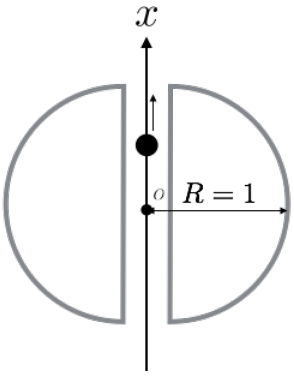

図 1: 惑星トンネルのモデル

例題として，惑星トンネル [6] という物理学のモデルについて述べる．半径 R の球状の惑星があり，惑星には中心 O を通るまっすぐな細い穴 (トンネル) が空いている．穴の体積は惑星の体積に対して充分小さいので無視できる．

物体が惑星の外にある場合，物体が受ける力は物体と惑星間の万有引力のみとする．つまり，物体が受ける力を f，O を中心とした物体の位置を x，物体の質量を m，惑星の密度を ρ，万有引力定数を G とすると，

$$f = G\frac{4m\pi\rho R^3}{3x^2} \tag{1}$$

となる．ただし，式 (1) をもとにした微分方程式は数式処理で解くことが困難なため，本研究では式 (2) に示すように，惑星に働く万有引力を距離によらず一定のものとして近似している．

$$f = \begin{cases} \dfrac{4}{3}\pi mG\rho R^3 & (x < -R) \\ -\dfrac{4}{3}\pi mG\rho R^3 & (x > R) \end{cases} \tag{2}$$

次に物体が穴の中にある場合，物体の位置を $x\,(-R < x < R)$ とすると，物体に受ける力は O を中心とした半径 $|x|$ の惑星から受ける万有引力と等しい．つまり，物体が受ける力を f，物体の質量を m，惑星の密度を ρ，万有引力定数を G とすると，

$$\begin{aligned} f &= G\frac{4m\pi\rho x^3}{3x^2} \\ &= \frac{4}{3}\pi\rho Gx \end{aligned} \tag{3}$$

となる．

本モデルを HydLa プログラムとして記述したものを図 2 に示す．ただし，$m = 1$，$G = 0.667$，$\rho = 0.552$ としている．HydLa プログラム中では，すべての変数は時刻に関する関数変数であり，x' は変数 x の時間微分を表す．INIT は物体の初期状態に対応する制約であり，$x(0) = 0.5$，$x'(0) = 10$ としている．CONST は定数に関する制約であり，FORCE1，FORCE2，FORCE3 はそれぞれ物体が惑星の内部，惑星の正の外側，惑星の負の外部にいる時の運動に対応した制約である．

```
INIT(x0) <=> x=x0 & x'=10.
CONST <=> [](r=0.552 & g=0.667).
FORCE1 <=> [](x''=-4/3*Pi*r*g*x).
FORCE2 <=> [](x- > 1 => x''=-4/3*Pi*r*g).
FORCE3 <=> [](x- < -1 => x''=4/3*Pi*r*g).

INIT(0.5), CONST, FORCE1 << (FORCE2, FORCE3).
```

図 2: 惑星トンネルの HydLa プログラム

2.3 HyLaGI の問題点

図 2 のプログラムを従来の HydLaGI で実行すると，式 (4) の条件を満たす最小時刻 t が導き出せないため，シミュレーションが失敗してしまう．

$$\begin{cases} \dfrac{1}{2}\cos\left(\dfrac{23\sqrt{\dfrac{29\pi}{2}}t}{125}\right) + \dfrac{1250\sqrt{\dfrac{2}{29\pi}}}{23}\sin\left(\dfrac{23\sqrt{\dfrac{29\pi}{2}}t}{125}\right) = 1 \\ t > 0 \end{cases} \tag{4}$$

式 (4) は条件式に三角関数を含んでおり，数式処理による枠組みで扱うのは難しいため，本研究では代わりに区間演算の技術で精度保証解を求めることを考えた．

3. 提案手法

本手法は HydLa のシミュレーションにおける離散変化時刻の計算時に，数式処理の代わりに区間演算を用いて精度保証解を導くことを目的としている．このため，区間演算を利用した代数方程式に対する求解手法の一つである区間ニュートン法によって解の候補を求める．さらに区間ニュートン法により得られる解のうち，HydLa のシミュレーションに必要なものだけを得るための手法と，得られた解の中に真の解が唯一存在することを保証する手法を考案した．本節では区間ニュートン法や上記 2 つの手法の詳細について説明する．

3.1 区間ニュートン法

区間ニュートン法は代数方程式に対する求解手法の一つで，区間演算により厳密解を含む数値区間を導出することができる．以下，求解対象の方程式を $f(x) = 0$ とする（f は実変数 x に関する実関数で，連続かつ微分可能であるとする）．区間ニュートン法は初期区間 $X^{(0)}$ から開始し，式 (5) で示した $X^{(k+1)}$ を求める計算（これを 1 ステップとする）を繰り返すことで区間を狭めていく．

$$X^{(k+1)} = X^{(k)} \cap N\left(X^{(k)}\right) \quad (k = 0, 1, \dots) \tag{5}$$

ただし，

$$N(X) = m(X) - F(m(X))/F'(X) \tag{6}$$

ここで F，F' はそれぞれ f，f' の区間拡張であり，$m(X)$ は X の中点を表す．区間ニュートン法は初期区間 $X^{(0)}$ を適切に設定することで，$X^{(0)}$ 内に存在する真の解を含む区間を導出することができる．

しかし 式 (6) において $0 \subset F'\left(X^{(0)}\right)$ を満たす場合には，0 を含む区間による除算が必要になってしまうため，通常の区間ニュートン法は適用できない．したがって，本研究では，$0 \subset F'\left(X^{(0)}\right)$ を満たす場合でも計算可能な拡張区間ニュートン法を採用する．

3.2 拡張区間ニュートン法

拡張区間ニュートン法は，区間ニュートン法内で用いる区間演算の除算を式 (7) のように拡張したものである．

$$1/[c,d] = \begin{cases} [1/d, +\infty) & (c = 0 < d) \\ (-\infty, 1/c] \cup [1/d, +\infty) & (c < 0 < d) \\ (-\infty, 1/c] & (c < d = 0) \end{cases} \tag{7}$$

この拡張により，$0 \subset F'\left(X^{(0)}\right)$ を満たす場合に，拡張区間ニュートン法の計算結果は式 (8) に示すように二つの区間に分割される．

$$X^{(k+1)} = X_L \cup X_U \quad (X_L < X_U) \tag{8}$$

分割した区間それぞれに対して区間ニュートン法の計算を行うことで，初期区間内に含まれる複数の解を計算することができる．

3.3 最小解の計算

拡張区間ニュートン法では，区間の分割が生じるため，初期区間 $X^{(0)}$ 中に存在する複数の解を求めることが可能である．しかし HyLaGI において，離散変化時刻として採用されるべき値は 0 より大きい最小の値である．しかし，分割が生じた際に次のステップで採用する区間を単純に式 (8) で示している X_L としてしまうと，解の存在しない区間を採用してしまう可能性がある．したがって，初期区間中から得られる解の中で，HyLaGI で採用すべき解を選択する手法が必要となる．

最小解の計算アルゴリズムを図 3 に示す．本アルゴリズムは入力として，計算対象の方程式 $Exp = 0$ の左辺 Exp と，Exp を時刻に関して微分した $DExp$ と，拡張区間ニュートン法の初期区間 $init$ とを用いる．ただし，HyLaGI は次の離散変化時刻までの時間が 0 以下であるということを許していないため，$init$ は負の値を含まない区間に限定する．出力は，初期区間内で真の解を含む数値区間 ret である．本アルゴリズムでは，スタックを用いて解候補である区間の管理を行っている．

まず，スタックに初期区間を入れ，返り値である ret には空集合を代入する（図 3:1〜2 行目）．返り値が空集合の場合，$init$ 中に解が存在しないことを示す．

次に，スタックが空になるまで拡張区間ニュートン法の計算の反復を行っていく（図 3:3〜29 行目）．始めにスタックから解候補の区間を取り出し，区間の分割が生じるかを判定する．分割が生じる場合，分割した 2 つの区間それぞれを求める．分割した区間のうち値の大きい方が空集合でないかを判定し（図 3:12 行目），空集合でない場合，スタックに入れる．分割が生じない場合は，通常の区間ニュートン法の計算を行う．

毎回の反復の終わりに，計算結果のチェックを行う（図 3:19 行目）．ここで計算した結果が空集合であった場合，現在の候補区間を破棄し，スタックから次の候補区間を取り出し，拡張区間ニュートン法の反復を行う．計算された区間が前回の区間とまったく同じものだった場合，正常に区間ニュートン法の計算が収束したと判定し，計算結果を返り値として採用する．

以上により，初期区間中に解が存在する場合，必ず解の存在する区間を求めることができる．解候補区間についてより値の小さいものを優先した探索を行っているため，最初に収束した解が最小解となる．

Input: 数式 Exp, Exp を時間微分した数式 $DExp$, 初期区間 $init$
Output: 近似区間 ret

1: $stack.push(init)$
2: $ret = \emptyset$
3: **while** $stack$ is not empty **do**
4: $current := stack.pop()$
5: **for** $i = 1$ to 100 **do**
6: $prev := current$
7: **if** $Exp(mid(current))/DExp(current)$ is divided into two intervals **then**
8: $ltmp := Larger\left(mid(current) - \frac{Exp(mid(current))}{DExp(current)}\right)$
9: $stemp := Smaller\left(mid(current) - \frac{Exp(mid(current))}{DExp(current)}\right)$
10: $linterval := current \cap ltmp$
11: $sinterval := current \cap stmp$
12: **if** $linterval \neq \emptyset$ **then**
13: $stack.push(linterval)$
14: **end if**
15: $current := sinterval$
16: **else**
17: $current := mid(current) - \frac{Exp(mid(current))}{DExp(current)}$
18: **end if**
19: **if** $current = prev \vee current = \emptyset$ **then**
20: break
21: **end if**
22: **end for**
23: **if** $current = \emptyset$ **then**
24: continue
25: **else**
26: $ret := current$
27: break
28: **end if**
29: **end while**
30: **return** ret

図 3: 最小解の計算アルゴリズム

3.4 解の唯一性保証

最小解の計算アルゴリズムにおいて，反復の終了条件を $current = prev$ としているが，この条件は文献 [1] 中の Theorem 1 を満たしておらず，厳密には得られた数値区間中に真の解が複数存在している可能性がある．したがって，最小解の計算アルゴリズムによって得られた数値区間中に真の解が唯一存在することを示す手法が必要となる．

ある区間 X 中に方程式の真の解が唯一存在することを保証するためには，X に対して区間ニュートン法を 1 ステップ進めた時に，X が計算結果の区間を強く包含していることが確認できれば良い [1]．本研究では，最小解の計算で得られた区間 X を少しだけ広げた区間に対して区間ニュートン法を適用することで，与えられた区間が真の解を 1 つだけ含む区間であることを保証するアルゴリズムを提案する．

解の唯一性を保証するアルゴリズムを図 4 に示す．本アルゴリズムは入力として与えられた数値区間 $candidate$ の中に，方程式 $Exp = 0$ の解があるかどうかを判定する．出力は，$candidate$ 中に真の解が唯一存在したかを示す $assurance$ である．始めに，$candidate$ が縮退区間（上下限が一致している区間）かを判定し（図 4:1 行目），縮退区間だった場合，解が 0 以外の 1 つの値に収束できたことになるので，$assurance$ を

Input: 数式 Exp, Exp を時間微分した数式 $DExp$, 解候補区間 $candidate$
Output: 保証の成否 $assurance$

1: **if** $candidate$ is a degenerate interval **then**
2: $assurance := true$
3: **else**
4: $spread := 10^{floor(\log_{10}(width(candidate)))}$
5: $tmp := [candidate.lower - spread, candidate.upper + spread]$
6: **if** $Exp(mid(tmp))/DExp(tmp)$ is divided into two intervals **then**
7: $assurance := false$
8: **end if**
9: $next := mid(tmp) - \frac{Exp(mid(tmp))}{DExp(tmp)}$
10: **if** $next \subsetneq tmp$ **then**
11: $assurance := true$
12: **else**
13: $assurance := false$
14: **end if**
15: **end if**
16: **return** $assurance$

図 4: 解の唯一性保証アルゴリズム

true にする．次に，縮退区間ではない場合の計算について説明する．*candidate* 中に真の解が存在することを示すために，*candidate* の上下限を *spread* 分だけ広げた区間 *tmp* を用意し，*tmp* を区間ニュートン法で計算した結果である *next* を求める．*tmp* の設定の候補として，上限値を *candidate* の上限値より大きい最小の浮動小数点数にし，下限値を *candidate* の下限値より小さい最大の浮動小数点数にした区間が考えられたが，式 4 で実験した結果，*next* の端点が *tmp* の端点と一致してしまい，保証に失敗した．これは，*tmp* の広げ方が小さすぎるために，区間ニュートン法を 1 ステップ進めても区間が狭まらなかったためである．*spread* は *candidate* の幅を基に設定している値であり，式 4 やこれまでに試したいくつかの例題では保証に成功している．

4. 評価実験

図 2 で示したプログラムに対して，提案手法を用いて 9 Phase 分シミュレーションを行った．シミュレーションの結果として，IP から PP へ遷移する時間と PP 遷移時の速度を示す．PP1 は初期状態であるので結果から省略する．

提案手法の結果を表 1 と表 2 に示す．

表 1: 提案手法の結果（遷移時間）

Phase	遷移時間
IP2 → PP3	[0.05012923158843772, 0.05012923158843844]
IP4 → PP5	[12.89288555848842, 12.89288555848853]
IP6 → PP7	[0.2001302392146061, 0.2001302392146176]
IP8 → PP9	（式の複雑化により計算不能）

表 2: 提案手法の結果（PP への遷移時の速度）

Phase	PP への遷移時の速度
IP2 → PP3	[9.941997578476402, 9.941997578476407]
IP4 → PP5	[−9.941997578476494, −9.941997578476319]
IP6 → PP7	[−9.941997578476552, −9.941997578476261]
IP8 → PP9	（式の複雑化により計算不能）

提案手法の場合，式 4 で示した条件式の解を区間演算によって求められるため，IP2 以降のシミュレーションが可能となっている．しかし，IP8 以降のシミュレーションは，計算時間が膨大になってしまうため，シミュレーションを止めざるを得なくなっている．

提案手法により，離散変化時刻を求める方程式が複雑になってしまうモデルに対して，HyLaGI でシミュレーション可能となることが確かめられた．計算時間が膨大になる理由は，離散変化時刻が数値区間として表すことになるため，離散変化時刻を求めるための方程式を導く微分方程式が複雑化してしまい，数式処理によって方程式を求めることが困難になるためであると考えられる．さらに，数値区間を用いて真の解を近似しているため，数式処理の枠組みにおいては生じえなかった不必要な場合分けが発生するモデルがある．このようなモデルに対して，ガード条件から PP 遷移時に変数が満たすべき値を導出し，PP においてその値に置換して対処している．なお，区間ニュートン法は 1 つの方程式に対してのみ適用可能な手法なので，提案手法はガード条件が不等式と方程式の論理積で構成されている制約が存在するモデルに対しては適用できない．そこで，ガード条件を構成している方程式や不等式のうち，一部を区間ニュートン法で，それ以外は数式処理を用いて計算し，全てを満たす値を選択することで対処している．

5. まとめと今後の課題

離散変化時刻に関する方程式を数式処理によって解くことができないという問題に対し，区間ニュートン法を用いた求解手法を構築し，従来手法ではシミュレーションできなかったモデルに対して，シミュレーションできるようになったことを確認した．今後の課題としては，シミュレーション可能なモデルを増やすために，現状の問題点を解決する手法を考案することが挙げられる．例えば，微分方程式の複雑化による計算時間の膨大化に対しては，離散変化時刻を求める方程式自体を近似することによって解決されると考えられる．

本研究の一部は科学研究費補助金（基盤研究（B）26280024）の補助を得て行った．

参考文献

[1] Chin-Yun Chen: Extended interval Newton method based on the precise quotient set, Computing, August 2011, Volume 92, issue 4, pp. 297–315.

[2] Lunze, J. : Handbook of Hybrid Systems Control: Theory, Tools, Applications, Cambridge University Press, 2009.

[3] 松本翔太，上田和紀: ハイブリッド制約言語 HydLa の記号実行シミュレータ Hyrose，コンピュータソフトウェア，Vol.30，No.4(2013)，pp.18–35.

[4] Platzer, A., Quesel, J.D. : KeYmaera : A Hybrid Theorem Prover for Hybrid Systems. In IJCAR 2008, LNCS 5195, Springer-Verlag, 2008, pp.171–178.

[5] Ramon E.Moore, R.Baker Kearfott, Michael J.Cloud: Introduction to INTERVAL ANALYSIS, Society for Industrial and Applied Mathematics, Philadelphia, 2009.

[6] 数研出版編集部，2007，2008 物理 I・II 重要問題集，数研出版.

[7] 上田和紀，細部博史，石井大輔： ハイブリッド制約言語 HydLa の宣言的意味論， コンピュータソフトウェア，Vol.28, No.1 (2011), pp.306-311.

[8] Henzinger, T. : The Theory of Hybrid Automata, in Proc. LICS'96, IEEE Computer Society Press, 1996, pp.278-292.