修 士 論 文

対象のダイナミクスを考慮した
最適経路問題に関する研究

北陸先端科学技術大学院大学
情報科学研究科　情報システム学専攻

湯　紅偉

2007 年 3 月

修 士 論 文

対象のダイナミクスを考慮した
最適経路問題に関する研究

指導教官　平石 邦彦 教授

審査委員主査　平石 邦彦 教授
審査委員　金子 峰雄 教授
審査委員　上原 隆平 助教授

北陸先端科学技術大学院大学
情報科学研究科　情報システム学専攻

510069　湯　紅偉

提出年月：　2007 年 2 月

概要

本研究では，障害物が複数存在する環境下において，移動物体のダイナミクスを考慮した最適な衝突回避経路を求める問題を扱う．

従来研究で提案された方法の一つとして，移動領域を凸多面体の部分領域に分割し，各部分領域ごとに線形ダイナミクスをもつ区分的線形システム（PieceWise Linear，PWL）システム上の最適制御問題として定式化する方法がある．この方法では，障害領域の回避は，部分領域間の離散的状態遷移に制限を加えることで実現しているため，必ずしも最適解が求められるわけではない．領域分割を細かくすることで解の精度は向上するが，計算時間が増えるという問題がある．

本研究では，領域分割の方法に着目して，既存研究の状態遷移に新しい種類の遷移を追加することで，既存研究では禁止されていた移動を可能にする方法を提案する．これにより，解の精度を向上させることが可能になる．

目次

第 1 章　はじめに

1.1 目的

本研究では，障害物が複数ある環境下において，移動物体のダイナミクスを考慮した最適な衝突回避経路を求める問題を扱う．

従来研究で提案された方法の一つとして，移動領域を凸多面体の部分領域に分割し，各部分領域ごとに線形ダイナミクスをもつ区分的線形システム（Piecewise Linear，PWL）システム上の最適制御問題として定式化する方法がある．この方法では，障害領域の回避は，部分領域間の離散的状態遷移に制限を加えることで実現しているため，必ずしも最適解が求められるわけではない．領域分割を細かくすることで解の精度は向上するが，計算時間が増えるという問題がある．

本研究では，領域分割の方法に着目して，既存研究の状態遷移に新しい種類の遷移を追加することで，既存研究では禁止されていた移動を可能にする方法を提案する．これにより，解の精度を向上させることが可能になる．

1.2 背景・特徴

ハイブリッドシステム上での最適化問題を扱うための従来研究として，論理混合ダイナミカル（Mixed Logical Dynamical，MLD）システムという表現方法でシステムを 0-1 変数を含む線形不等式に変換し，混合整数2次計画法により最適解を求める方法がある[1]．しかし，混合整数計画法は一般に非常に多くの計算時間を必要とする．

それに対し，離散的状態遷移を CLP（Constraint Logic Programming）を使って探索的に計算する手法が提案されており，ある種の問題に対してはより少ない計算時間で解を求めることができる[2]．障害物回避問題においても，移動領域を有限個の部分領域に分割し，さらに障害物に衝突しないように部分領域間の離散的状態遷移を制限することで，探索的に障害物を回避する経路を計算することができる．解の精度および計算時間は領域分割の方法に強く依存するが，従来研究ではそれらについて十分な検討が行われているとは言えない．本研究では，領域分割の方法，および，実行不能な離散的状態遷移の制限方法について検討する．

第 2 章　問題の定式化

2.1　準備

2.1.1　ハイブリッドシステム

本研究で扱う移動物体の障害物回避問題はハイブリッドシステム（Hybrid System）という表現方式で研究されている.

ハイブリッドシステムとは，アナログ動作とデジタル動作が混在するシステムであり[3]，自動車や航空機，プラント制御，ニューラルワーク，遺伝子ネットワークなど様々な対象をハイブリッドシステムとして扱うことができる[4].

2.1.2　状態方程式

本研究は現実世界に存する移動物体を対象とし，ニュートンの運動方程式などの物理法則を反映したダイナミクスをもつ対象の動作計画を考える．例として，以下の状態方程式に従って動作する対象を考える．これは状態変数 $x = [x_1,\ x_2]'$の値によりダイナミクスが切り替わるPWLシステムである.

$$
\begin{aligned}
&x(t+1) = 0.8\begin{bmatrix}\cos\alpha(t) & -\sin\alpha(t)\\ \sin\alpha(t) & \cos\alpha(t)\end{bmatrix}x(t)+\begin{bmatrix}0\\1\end{bmatrix}u(t)\\
&y(t) = \begin{bmatrix}1 & 0\end{bmatrix}x(t)\\
&\alpha(t) = \begin{cases}\pi/3 & if\ \begin{bmatrix}1 & 0\end{bmatrix}x(t)\ge 0\\ -\pi/3 & if\ \begin{bmatrix}1 & 0\end{bmatrix}x(t)<0\end{cases}\\
&x(t)\in\begin{bmatrix}-1 & 1\end{bmatrix}\times\begin{bmatrix}0 & 1.5\end{bmatrix},\quad u(t)\in\begin{bmatrix}-1 & 1\end{bmatrix}
\end{aligned}
\qquad (2.1)
$$

本研究で扱う物体の移動は，ある位置から任意の目的位置まで移動できるのではなく，可能な次のステップにおける位置は必ずある規則（式(2.1)）により算出した範囲内の一点である．また，多次元の場合は，状態空間内の禁止領域を回避する最適トラジェクトリを計算する問題になる.

2.2 最適経路問題の概要

本研究で扱う最適経路問題では以下を仮定する.

1. 対象のダイナミクスは離散時間の差分方程式で与えられる.
2. 障害物はそれを囲む最小の矩形(障害領域)で近似する.
3. 移動物体は指定された状態方程式に従って動く.
4. 1ステップの移動に対して, 障害物と衝突しないという判断は, 単に始点と終点が障害領域内にないだけではなく, 始点と終点を結ぶ線分上の全ての点も障害領域内にないという基準で行う.
5. 最適性は与えられた目的関数の最小化として与える(2.3.3 参照). 目的関数はステップ数に対して単調非減少であると仮定する.

2.3 従来研究の処理法

2.3.1 領域分割

領域分割とは，初期データ（物体の移動範囲，障害物の数・範囲と物体の運動方程式）を定めた後の作業である．つまり，移動領域，障害物，運動方程式のいずれかを変更すると，領域分割を再度行う必要がある．

分割方法について

全体を「状態方程式による分割」と「障害物による分割」の二つからなる．

✧ 状態方程式による分割

式(2.1)の場合，状態方程式はx_1座標が 0 以上と 0 未満で状態方程式が異なることから，まず移動領域を $x_1 = 0$ の直線により図 2-1 のような二つの領域に分ける．

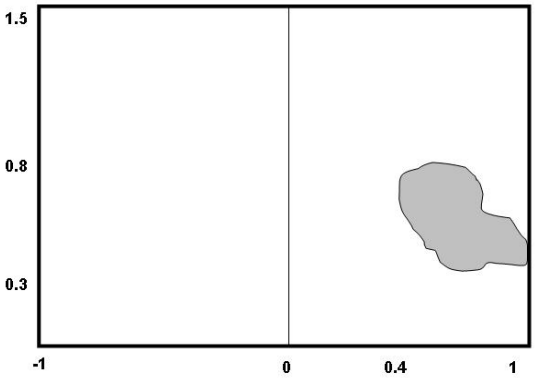


図 2-1 運動方程式の特徴より分割

✧ 障害領域による分割

障害物からそれを含む最小の矩形（x_1，x_2座標の最大・最小値より求められる）を求め，さらに，障害領域の境界線により移動領域全体を部分領域に分ける（図 2-2）．各部分領域は矩形である．

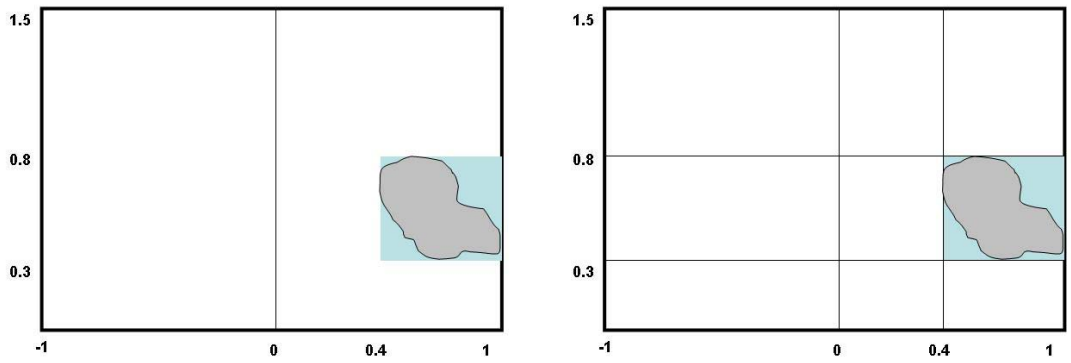


図　2-2　障害領域より分割

2.3.2　領域間状態遷移の制限

障害物回避のために，領域間の状態遷移の制限を行う．ここで，2.2 で述べたように各ステップにおける各位置 $x(t)$ が障害領域に含まれないだけではなく，線分（$x(t)$, $x(t+1)$）も障害領域に含まれてはならない．

以下の基準で領域間の状態遷移を制限する．

- ある経路が障害物と交差しないならば，それと同じ離散的状態遷移をする任意の経路が障害物と交差しない．

つまり，図 2-3 の左図のような全てのパスとも障害物と衝突しない遷移が許可され，右図のような一部分が衝突せず，また，一部分が衝突しているような遷移は禁止される．

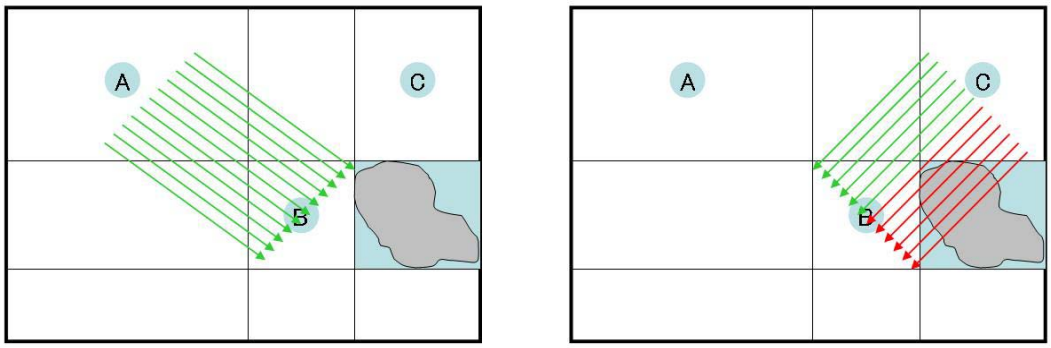


図　2-3　既存研究での許可・不許可遷移

以上の規則に従って部分領域間の状態遷移を追加すると，図 2-4 のようになる．

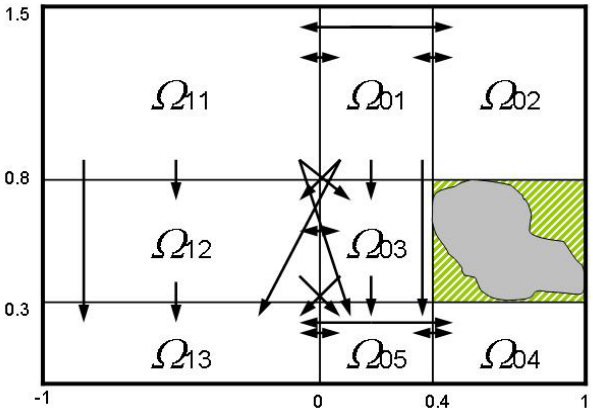


図　2-4　既存研究の遷移

2.3.3　最適経路の定義

本研究の最適経路とは，式(2.2)に与えられるような2次形式目的関数を最小化するような経路である．ここでは，入力の大きさと各ステップにおける終点までの距離の最小化を考慮した目的関数になっている．

$$J_h\left(x_0, x_f, u\right) = \sum_{t=0}^{h-1} \left\| u(t) \right\|_{Q_1}^2 + \left\| x(t) - x_f \right\|_{Q_2}^2 \qquad (2.2)$$

第 3 章　解精度向上のための方法

本章では既存研究における問題点の分析をまず行う. 既存研究では, 二つの部分領域の間で, 全てのパスの中に少なくとも一つのパスが障害領域を通るならば, 他の障害領域を通らないパスも全て禁止され, この二つの部分領域間の遷移が許可されない. 本研究では, これら「禁止されてしまったパス」も考慮することで, 解の精度を向上させる.

3.1　状態遷移の改良について

本研究ではつぎのような新しい種類の遷移を追加する. 図 3-1 のように, R_1が出発領域, R_3が目的領域, そして, 中央のR_2が障害物としたとき, 明らかに領域R_3内Ⓐの部分は, 領域R_1から到達不可能である. それに対し, Ⓑの部分は領域R_1中の任意の点から到達可能である.

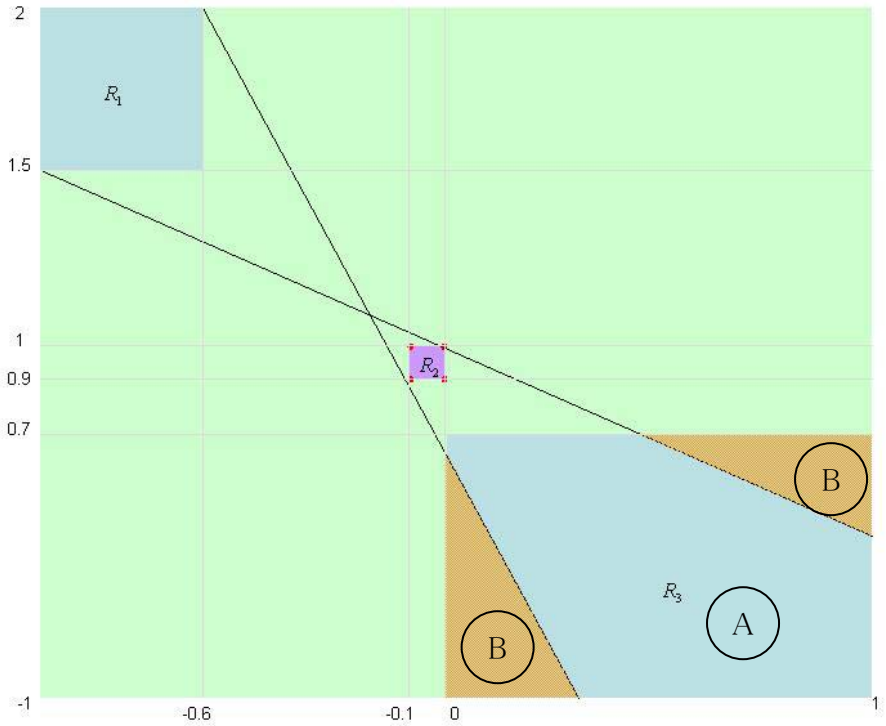


図　3-1　到達可能範囲と到達不可能範囲

ここで，Ⓐの部分の範囲は，R_1内の任意の点とR_2中の任意の点を通る直線上の点で，かつ，R_3に属している点の集合として定義される．このことより，式(3.1)が得られる．

$$f(x_3,y_3)=\exists x_1,y_1,x_2,y_2:\begin{bmatrix}(x_1,y_1)\in R_1\wedge(x_2,y_2)\in R_2\wedge(x_3,y_3)\in R_3\\ \wedge line([x_1,y_1],[x_2,y_2],[x_3,y_3])\end{bmatrix} \tag{3.1}$$

ここで，*line*()は各点が同一直線状にあることを表す述語である．

これを満たす点の集合は，一階述語論理式から限量子を削除した等価な式を求める手法である Quantifier Elimination [5]により計算できる．さらに，本研究で扱う矩形領域に対しては，以下に示すより簡単な方法で到達可能範囲が求められる．

Ⓐの部分を求めるために，まず出発領域R_1内の任意の点から到達可能な範囲を求めてみる．

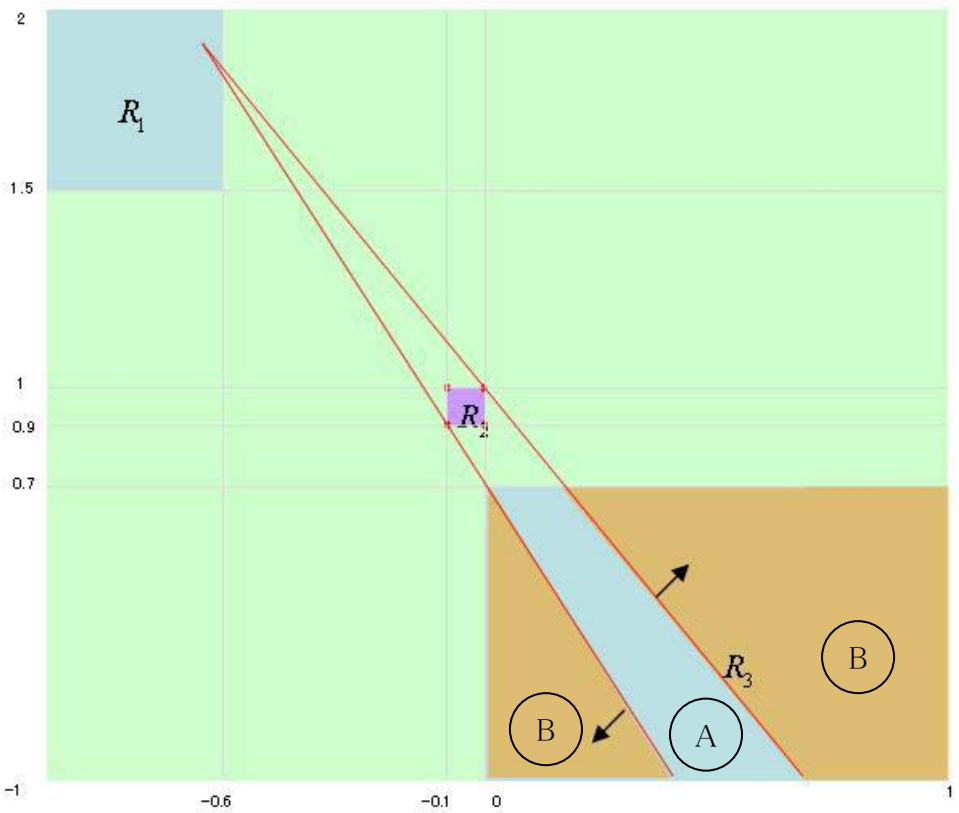


図　3-2　一点の到達可能・不可能範囲

各直線は各領域の点$(x_1, y_1) \in R_1$, $(x_2, y_2) \in R_2$, $(x_3, y_3) \in R_3$ を通過することから，式(3.2)が得られる．

$$\frac{x_1 - x_2}{y_1 - y_2} = \frac{x_2 - x_3}{y_2 - y_3} \tag{3.2}$$

式を変形すると，次の式になる

$$\begin{aligned}
\frac{x_1 - x_2}{y_1 - y_2} &= \frac{x_2 - x_3}{y_2 - y_3} \\
(x_1 - x_2)\cdot(y_2 - y_3) &= (x_2 - x_3)\cdot(y_1 - y_2) \\
(x_1 - x_2)\cdot y_2 - (x_1 - x_2)\cdot y_3 &= x_2\cdot(y_1 - y_2) - x_3\cdot(y_1 - y_2) \\
(x_1 - x_2)\cdot y_3 &= x_3\cdot(y_1 - y_2) - x_2\cdot(y_1 - y_2) + (x_1 - x_2)\cdot y_2 \\
y_3 &= -\frac{y_1 - y_2}{x_2 - x_1}\cdot x_3 + x_2\cdot\frac{y_1 - y_2}{x_2 - x_1} + y_2
\end{aligned} \tag{3.3}$$

つぎに，式(3.3)より，R_1領域内の１つの固定した点(x_1, y_1)に対して，到達可能な範囲の特徴を分析してみる．このとき，x_1，y_1は定数であり，傾きが最大および最小のときの直線は図 3-2 中の三角形の境界線である．つまり，式(3.3)の直線の傾きの最大値と最小値を求めれば，Ⓐの部分の境界が得られる．

$$\text{最小値は}\quad -\frac{y_1 - y_2 \downarrow}{x_2 \downarrow - x_1} \qquad \text{最大値は}\quad -\frac{y_1 - y_2 \uparrow}{x_2 \uparrow - x_1} \tag{3.4}$$

到達可能範囲Ⓑは，図 3-2 のように，傾きが最小の直線の左下と傾きが最大の直線の右上の部分である．

以上の分析により，このような領域の配置の場合，以下が得られる．

出発領域R_1内の任意の点に対して，到達可能な範囲は
その点と 00 頂点からなる接線の左下部分と
その点と 11 頂点からなる接線の右上の二つの部分である．

※　注：　障害物の4つ頂点の命名は図 3-3 の通り．

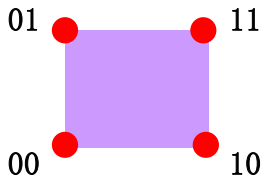


図　3-3　障害物頂点の命名

このことを利用して，R_1領域内の任意の点から到達可能なR_3内の部分領域を，00

頂点と 11 頂点の二つの場合に分けて計算してみる.

00 頂点に対して

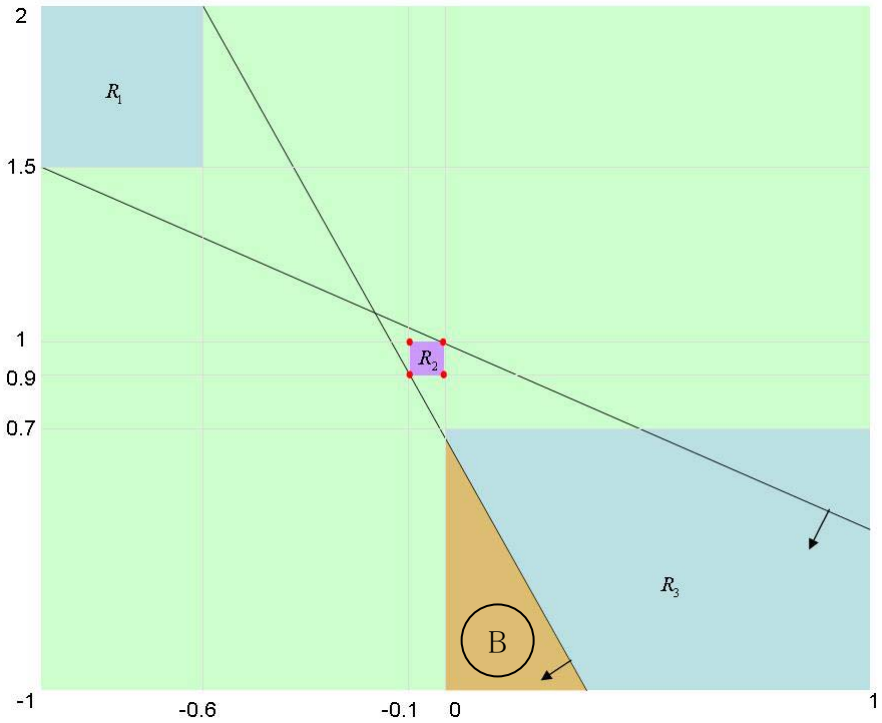


図 3-4 00 頂点の場合

接線は式(3.5)で表される.

$$y_3 = -\frac{y_1 - y_2}{x_2 - x_1} \cdot x_3 + x_2 \cdot \frac{y_1 - y_2}{x_2 - x_1} + y_2 \tag{3.5}$$

直線の傾きをk, y切片をbとおくと, R_1内の任意の点に対して, 式(3.6)が表している部分は到達可能である.

$$y < k \cdot x + b \tag{3.6}$$

そして, 全ての点から到達可能な範囲の共通部分は式(3.7)で与えられる.

$$y < k_{\max} \cdot x + b \tag{3.7}$$

ここで, k_{max}は傾きkの最大値である.

同様に, **11 頂点に対して**

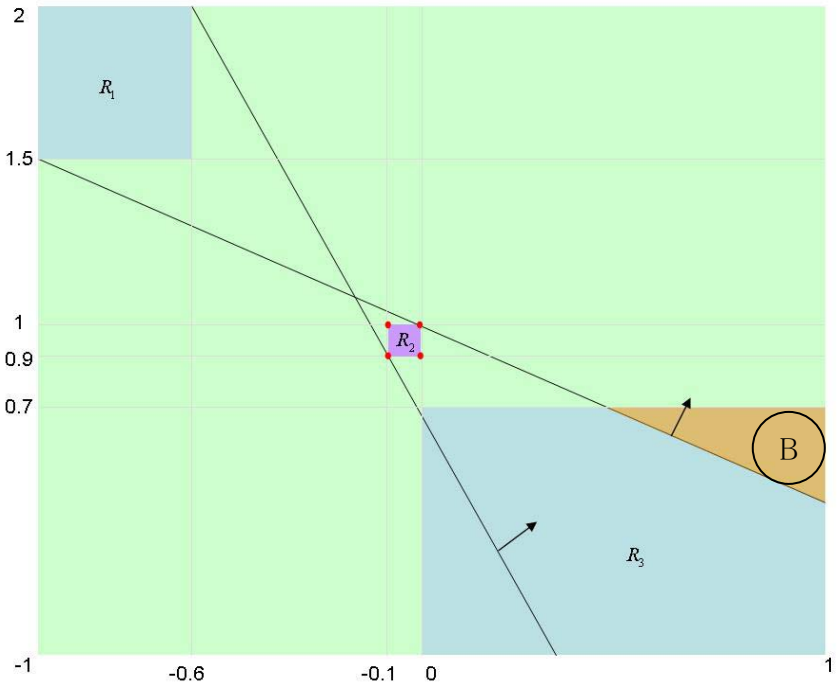


図 3-5 11 頂点の場合

R_1領域内の全ての点から到達可能範囲の共通部分は式(3.8)で与えられる.

$$y > k_{\min} \cdot x + b \tag{3.8}$$

ここで, k_{min}は傾きの最小値である.

そしてR_1領域の境界値と 00, 11 頂点の座標を代入すると,

$$\begin{aligned} y_3 &= -\frac{2-0.9}{-0.1-(-0.6)} \cdot x_3 + (-0.1) \cdot \frac{2-0.9}{-0.1-(-0.6)} + 0.9 \\ 20y_3 &= -44x_3 + 17 \\ y_3 &= -\frac{1.5-1}{0-(-1)} \cdot x_3 + 0 \cdot \frac{1.5-1}{0-(-1)} + 1 \\ 2y_3 &= -x_3 + 2 \end{aligned} \tag{3.9}$$

以上の二つの直線の式が得られる. これらは図 3-2 に描かれた直線である. そして, Ⓐの部分の数式表現は,

$$\begin{aligned} &0 < x_3 < 1 \wedge 0 < y_3 < 0.7 \wedge \\ &20y_3 \geq -44x_3 + 17 \wedge 2y_3 \leq -x_3 + 2 \end{aligned} \tag{3.10}$$

となる. これにより, Ⓑの部分（本研究で求めたい部分）の数式表現は

$$0 < x_3 < 1 \wedge 0 < y_3 < 0.7 \wedge$$
$$(20y_3 < -44x_3 + 17 \vee 2y_3 > -x_3 + 2) \qquad (3.11)$$

となる.

以上の考えにより, 障害物が干渉している遷移から到達可能範囲を取り出すことができる.

3.2 可能領域の考え

前節で記述されていた「到達可能範囲」の算出方法に加え,本節で紹介する「可能領域」の概念も用いれば更に解の精度を向上することができる.

例により説明する. ある物体に対して, 次のような状態方程式が与えられているとする.

$$
\begin{aligned}
&x(t+1) = 0.8\begin{bmatrix}\cos\alpha(t) & -\sin\alpha(t)\\ \sin\alpha(t) & \cos\alpha(t)\end{bmatrix}x(t)+\begin{bmatrix}0\\1\end{bmatrix}u(t)\\
&y(t) = \begin{bmatrix}1 & 0\end{bmatrix}x(t)\\
&\alpha(t) = \begin{cases}\pi/3 & if\ \begin{bmatrix}1 & 0\end{bmatrix}x(t) \ge 0\\ -\pi/3 & if\ \begin{bmatrix}1 & 0\end{bmatrix}x(t) < 0\end{cases}\\
&x(t) \in \begin{bmatrix}-1 & 1\end{bmatrix}\times\begin{bmatrix}0 & 1.5\end{bmatrix},\quad u(t) \in \begin{bmatrix}-1 & 1\end{bmatrix}
\end{aligned}
\tag{3.12}
$$

この状態方程式により, 現時点の座標範囲から, 次のステップ(t + 2)の点の座標の存在可能範囲を矩形により近似して計算すると, 図 3-5 のような領域Ω_{11}中のⒶの部分の領域になる. これを「可能領域」と呼ぶことにする.

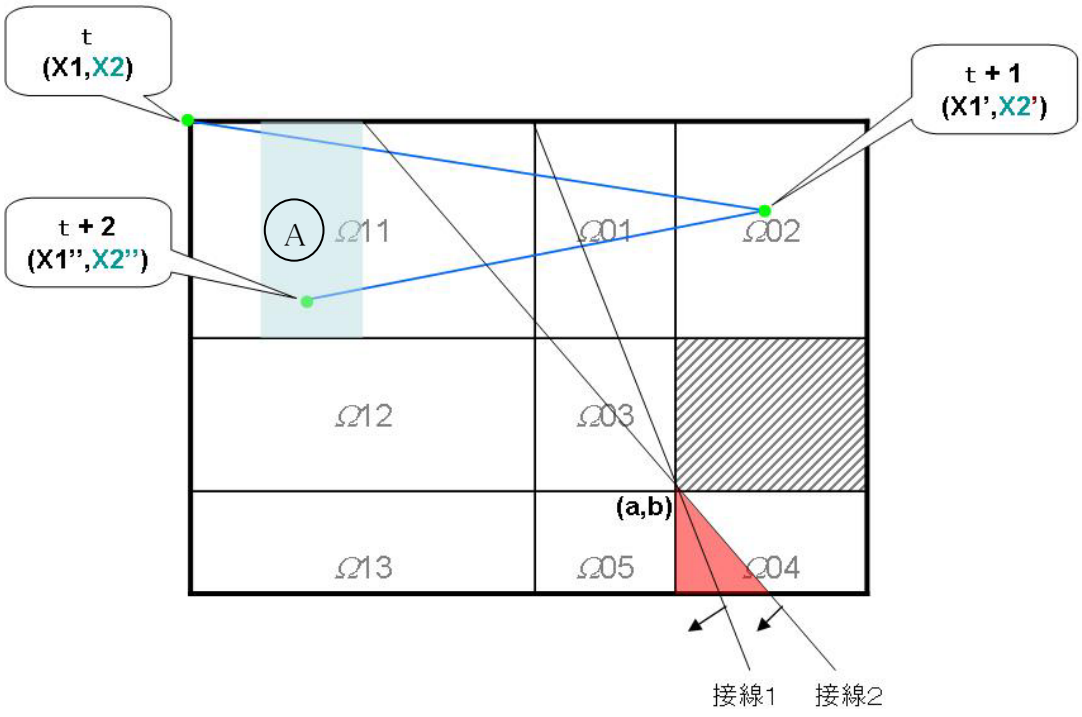


図 3-6 部分領域・部分領域内可能領域より作成した接線

可能領域は一般に出発領域よりも小さい．つぎに，領域Ω_{11}の頂点，および，可能領域の頂点から接線を引くと，それぞれ接線1と接線2となる．明らかに，可能領域の頂点から引いた接線2で計算すれば，物体の到達可能範囲がより広げられ，精度がより高くなる．

本研究では，各ステップごとに可能領域と部分領域の共通部分を計算し，それにより到達可能範囲を計算する．このとき，部分領域内の任意の矩形領域に対する到達可能範囲をあらかじめパラメトリックに計算しておくことができる．すなわち，矩形領域の各頂点をパラメータとして与え，それに対する到達可能範囲を計算できる．

3.3 制限付き遷移の定義

3.1 節および 3.2 節の遷移を，障害物が干渉している遷移（従来研究で禁止していた遷移）に追加すれば，障害物が干渉していても，到達可能範囲まで遷移することが可能になる．これを「制限付き遷移」と呼ぶ．

制限付き遷移の処理は以下の二つの処理に分かれる．

1. 出発領域内の可能領域から目的領域内の到達可能範囲を計算する
2. 目的領域内の可能領域へ到達可能な出発領域内の出発点範囲を計算する．

下図は，この二つの処理過程を表している．

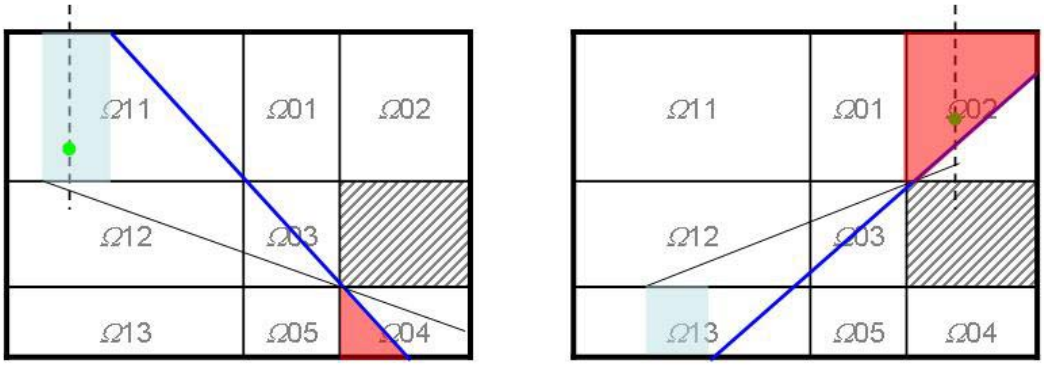


図　3-7　制限付け遷移の対処法

図 3-7 左図は，出発領域内の可能領域から目的領域内の到達可能範囲を計算する過程を示したものである．その手順は，

- 出発領域内の可能領域を算出し．
- 可能領域の頂点と障害物頂点により接線を引く．
- 接線の可能領域側の部分と目的領域の共通部分は，到達可能範囲である．

$$\begin{bmatrix}0\\1\end{bmatrix}x(T+1)=A21\times\begin{bmatrix}1\\0\end{bmatrix}x(t)+A22\times\begin{bmatrix}0\\1\end{bmatrix}x(T)+U$$
$$\begin{bmatrix}0\\1\end{bmatrix}x(T+1)\in[MAX\_LocN,MIN\_LocN] \qquad (3.13)$$
$$\left(A-\begin{bmatrix}1\\0\end{bmatrix}x(T+1)\right)\times\begin{bmatrix}0\\1\end{bmatrix}x(T)\geq\left(\begin{bmatrix}1\\0\end{bmatrix}x(T)-A\right)\times(B-MIN\_LocN)+B$$

図 3-7 右図は，目的領域内の可能領域へ到達可能な出発領域内の出発点範囲を計算する過程を示したものである．その手順は，

- 目的領域内の可能領域を算出し．
- 可能領域の頂点と障害物頂点により接線を引く．
- 接線の可能領域側の部分と出発領域の共通部分は，目的領域内の可能領域へ出発できる部分である．

$$\begin{bmatrix}0\\1\end{bmatrix}x(T+1)=A21\times\begin{bmatrix}1\\0\end{bmatrix}x(t)+A22\times\begin{bmatrix}0\\1\end{bmatrix}x(T)+U$$
$$\begin{bmatrix}0\\1\end{bmatrix}x(T+1)\in[MAX\_LocN,MIN\_LocN] \tag{3.14}$$
$$\left(A-\begin{bmatrix}1\\0\end{bmatrix}x(T+1)\right)\times\begin{bmatrix}0\\1\end{bmatrix}x(T)\geq\left(\begin{bmatrix}1\\0\end{bmatrix}x(T)-A\right)\times(B-MIN\_LocN)+B$$

なぜ二つ処理が必要と言うと，もし図 3-7 の右図のような領域Ω_{02}から領域Ω_{13}への遷移が左図のように到達可能範囲を求めると，到達可能範囲は図 3-8 のように空であるためである．

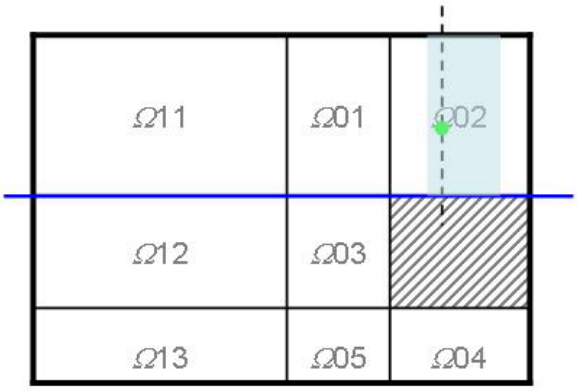


図 3-8 二種目遷移が必要となる理由

本研究は，まず左図のよう可能領域内のすべての点から到達可能な範囲を求め．それが空ならば，さらに右図のように，出発領域内の一部分の点から到達可能範囲を求める．

以上の考えにより，図3-8 に示す新しい遷移を追加することができる．実線・破線はそれぞれ図 3-7 の左図・右図の遷移を表す

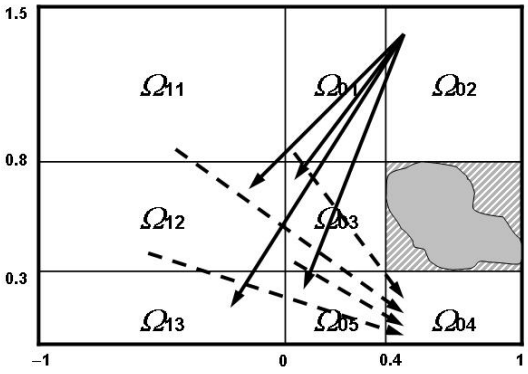


図 3-8 本研究で追加した新しい遷移

第 4 章　実装と評価

4.1　CLP 言語

本研究の経路探索方法は Perl および CLP 言語により実装した. この節では CLP に関して紹介する.

4.1.1　Prolog について

PrologはAI用言語で, PROgramming in LOGic（論理プログラミング）の省略である[6]. C言語やJava言語と違って, 手続き型言語* ではなく, 非手続き言語の論理型言語である.

非手続き言語とは, 前後の命令とは独立で, 一つ一つの命令で完結している言語のことである. 実行するたびに回答が返ってくる. それに対して, 手続き言語は, 関連を持った複数の命令を, 決められた順序で実行しなければならない言語のことである. 実行順序や回数を制御するために「分岐」や「繰り返し」といった命令が用意されている. しかし, 非手続き型言語である CLP は IF や FOR といった制御文はない.

また, 対話的な使用法はもう1つの特徴である. Prolog システムの使用形態は対話的である. すなわち, ユーザーは Prolog システムを立ち上げた後, システムと会話するような形で命令を与え, 述語を定義したり実行したりできる.

もっとも基本的な概念は記号, 変数, リスト, 述語, 事実, 規則, 質問などがある.

4.1.2　CLP について

CLP は Constraint Logic Programming（制約論理プログラミング）の省略で, Prolog の長所を継承した上で, 数値制約などさまざまな種類の制約を解く機能（ソルバー）追加した処理系である[7]. 本研究で使用するCLP言語である KCLP-HS は, 線形の不等式制約に対するソルバー, 線形および2次形式目的関数の最適化, および,

---

* 記述された命令を逐次的に実行し、処理の結果に応じて変数の内容を変化させていくプログラミング言語のこと。

凸多面体操作を行う機能を持つ言語である[7].

4.2 アルゴリズムと前処理について

実装したCLPプログラムは「前処理部分」と「探索部分」の二つからなる.

4.2.1 前処理部分

前処理部分とは, 実際の計算を行うCLPソースコードを作成するものである. 具体的には, 初期データ(物体の移動範囲, 障害物の数・範囲と物体の運動法方程式)が与えられた後に, その環境を反映した初期データからソースコードを作成する. 領域分割, 状態遷移の定義の二つの部分に分けて行っている. (詳しくは, 4.3 を参照)

このソースコードの作成する処理は, 既存研究では手動で行われていた. これに対して, 本研究では後述の章で説明する「作成ツール」を利用して, 自動的に作成する形で行っている.

4.2.2 探索部分

探索部分は, 全体を二つの部分に分かれる.

ステップ1:各領域からの目的関数値の下界

部分領域内の任意の点から終点までに移動する経路に対する目的関数の最適値を計算し, それをこの部分領域に対する目的関数値の下界とする.

ステップ2:探索

探索アルゴリズムは, 始点が属する部分領域を出発領域とし, 状態遷移で定義した遷移に従って, 到達可能な部分領域を一つずつ移動して行く。そして, 各ステップで現在位置が属する部分領域までの目的関数値の下限とステップ1の処理で求めた下界値の合計が現在までに求められている(一時的な)最適目的関数値と比較してより小さいならば, 探索をさらに継続する. そうでなければ, それまでに探索した経路は最適経路とは成り得ないので探索を中止し, バックトラックの機能により別のパスを探索する.

以上の探索が終ったら, 最適値と最適パスの各ステップの座標が出力される.

4.3 ソースコードの自動的な作成

既存方法では，CLP のソースコードを全て手で入力する形で作成した．このため，初期データ（物体の移動範囲，障害物の数・範囲と物体の運動方程式）を変更して計算したい時には，新たにコーディングしなおさなければならないという問題があった．それに対して，本研究では，初期データを入力するだけで，CLPのコーディングは全て Perl 言語で作ったスクリプト（make_clp.pl）により自動的に作成するという形になっている．

4.3.1 作成の流れ

- ✧ 初期データより領域分割して，各領域のデータをハッシュに格納しておく．
- ✧ 状態遷移を定義
  - ➢ 禁止する遷移を定義
  - ➢ 1ステップで終了した処理範囲*と障害領域の相対位置により，制限付き遷移を分類（4.3.3 を参照）して，そして，各種類に対してもその種の遷移の障害リスト（4.3.2 を参照）の作成方法を定義しておく．
- ✧ CLPソースコードの初期データと関係ない部分をそのまま!出力する

4.3.2 障害リストとは

図 4-1 のような制限付き遷移において，1ステップの経路を干渉している障害物の名前と到達可能領域を求めるための接線を引くための頂点を列挙したリストである．

[…, …, b00, 00, b00, 01, b01, 10, b01, 11]

図 4-1 障害リスト

4.3.3 制限付き遷移の分類

本研究で遷移の分類は，1ステップで終了した処理範囲（図 4-2 の左図）と障害領域（図 4-2 の右図）の相対関係により以下の手順で判別している．

---

* 出発・目的領域の座標の最大値・最小値から作った範囲である

! 全部そのままではなく、領域の数に応じて、事実の数も増える

- 障害領域を4つの頂点を持つ矩形とみなす.
- 処理範囲に含まれる障害物の頂点数は0から4の5つの可能性がある.
- 5つの可能性に基づいて，制限遷移を5種類に分けて，それぞれに対応した処理を行う

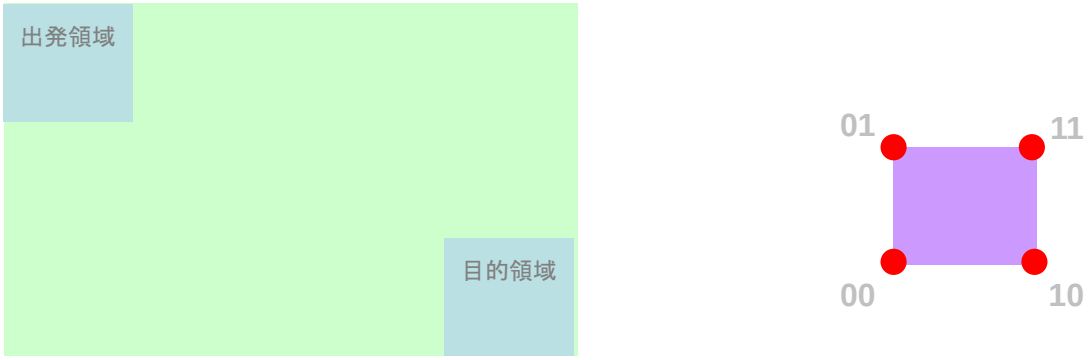


図 4-2 1 ステップで終了した処理範囲と障害領域

0から4の頂点を含まれる5種類の制限遷移へ対応した対処法をそれぞれ紹介する.

0個含まれる場合

以下の2つの可能性がある

- 図 4-3 の左図:直上から直下（又は，直左と直右間の移動）.
- 図 4-3 の右図:障害物なし.

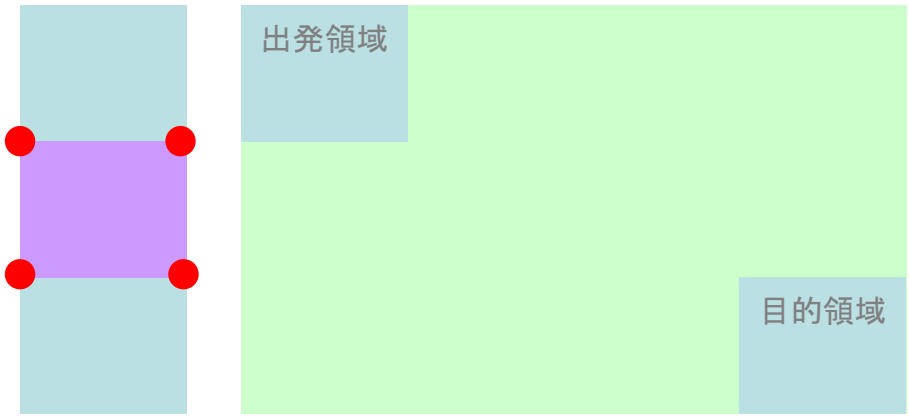


図 4-3 0個含まれる場合

- 左図の状況であれば，すべての遷移は禁止になる．
- 右図の状況であれば，既存研究の制限なし遷移になる．（この場合は，障害リストは空である）．

1個含まれる場合

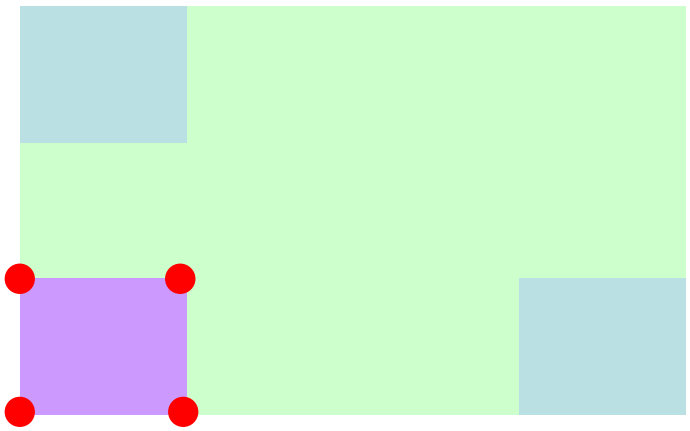


図 4-4 1個含まれる場合

含まれた頂点数が1個しかないので，判断を必要とせず，その障害物の名前とただ 1個の含まれた頂点を障害リストの後ろに追加する．

[…, …, b01, 11]

2個含まれる場合

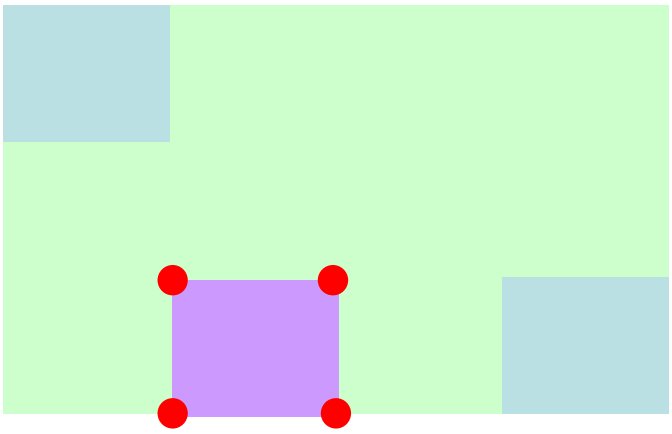


図 4-5 2個含まれる場合

含まれた頂点が2個あるとき，どの点を障害リストに乗せるのかを判断する必要がある．

[…, …, b01, 01]

or

[…, …, b01, 11]

ここでどちらの頂点を選択するかは以下の方法を用いる．

- 含まれていない二個の頂点が属する処理範囲の境界線を求める．
- その境界線が，出発・目的領域のどの境界線と同じ直線に属しているのかを判断する．
- その領域により近い頂点が接線を作るために用いる点である．

3.1 で提案した新しい種類の遷移を使って，出発領域から障害物を回避して，目的領域への到達可能範囲は，図 4-6 のような点1，3からなる直線の左下と点2，4からなる直線の右上の二部分と目的領域の共通部分である．つまり，この二つの直線は到達可能範囲の境界線である．

そこで，この二つの頂点が含まれていた場合には，点3が境界にあるため，明らかに1，3からなる直線は目的領域と交差していない．したがって，点2，4からなる直線により到達可能範囲を求めればよい．

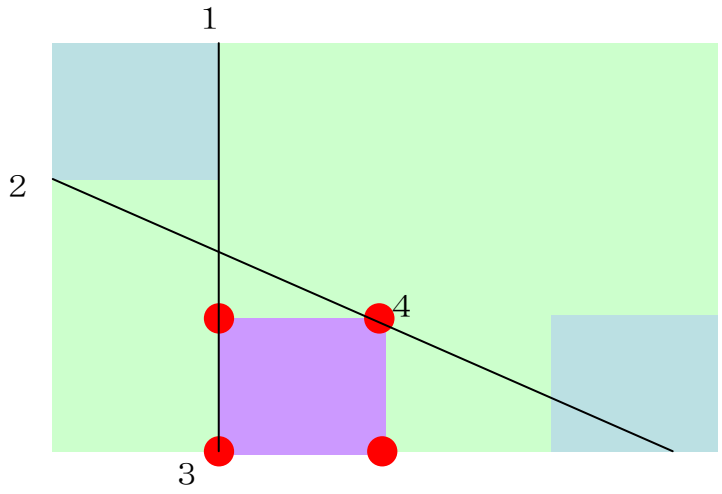


図 4-6 2つの境界線

3個含まれる場合

各部分領域および障害領域が矩形であることから，明らかにこのような場合は存在しない．

4個含まれる場合

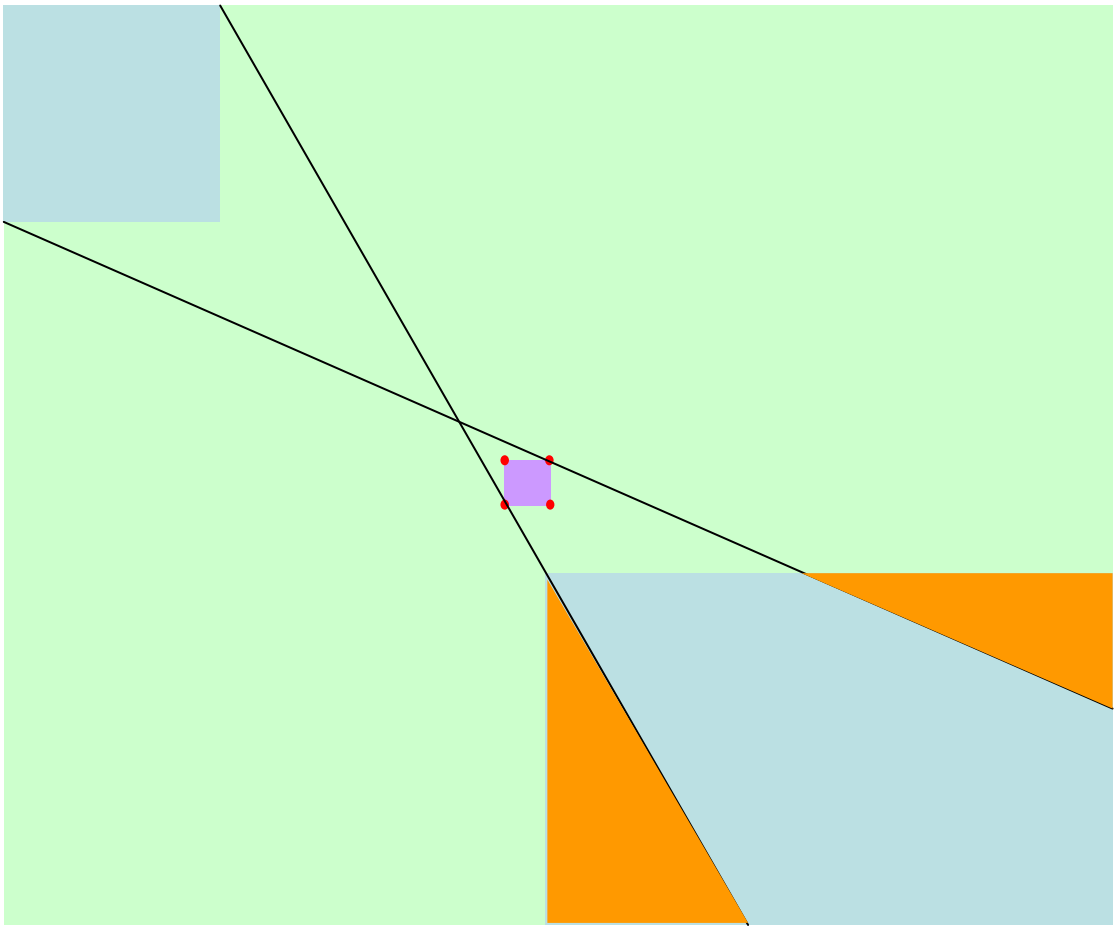


図 4-7 4個含まれる場合

この状況では，1つの障害物に対して，二本の接線を作る必要がある．しかし，このような障害物が中央で干渉している場合，以下の二つの場合にまとめることができる．

1. 左上から右下
2. 右下から左上

このとき，どの頂点を障害リストに載せるのかを判断する方法が必要となる．本研究

で用いた方法は以下のとおりである.

1の場合, 図 4-8 のように, 00 と 11 の二つの頂点を障害リストに載せる.

[…, …, b01, 00, b01, 11]

2の場合, 01 と 10 の二つの頂点を障害リストに載せる.

[…, …, b01, 01, b01, 10]

4.3.4 スクリプト make_clp.pl の使い方について

次の例で説明する.

```
# 環境変数の定義
$x0 = "-1";
$y0 = "1.5";
@area     = ("-1", "1", "0", "1.5");
%bad = (
            "0",        ["0.4", "1", "0.6", "0.8"],
            "1",        ["-1", "-0.6", "0.6", "0.8"],
            "2",        ["-0.4", "0", "0.2","0.3"]
);
@ver = ("00", "01", "10", "11");
```

上のソースコードは, 初期データを編集用の Perl のヘッドファイル(init.pm)である, 主な変数の説明は下のとおりである.

変数$x0:　　始点の x 座標
変数$y0:　　始点の y 座標
配列@area:　移動領域範囲
ハッシュ%bad:　各障害物の範囲

環境(初期データ)が変わるたびに, これらの値を変更すれば, 新しい環境に対応した CLP ソースコードを作成することができる.

- 入力: なし.
- 出力ファイル: CLP 環境で実行できる“clp”という CLP のソースコード.

4.4 実行例

4.4.1 状態方程式の変更

既存研究で用いていた状態方程式では入力 u で調整できるのが一方の座標しかないので，本研究の手法が十分には活用できない．ここでは既存研究の状態方程式を次のように変更した場合を考える．

$$
\begin{aligned}
&x(t+1) = 0.8\begin{bmatrix}\cos\alpha(t) & -\sin\alpha(t)\\ \sin\alpha(t) & \cos\alpha(t)\end{bmatrix}x(t)+\begin{bmatrix}1\\1\end{bmatrix}u(t)\\
&y(t) = \begin{bmatrix}1 & 0\end{bmatrix}x(t)\\
&\alpha(t) = \begin{cases}\pi/3 & if\ \begin{bmatrix}1 & 0\end{bmatrix}x(t) \ge 0\\ -\pi/3 & if\ \begin{bmatrix}1 & 0\end{bmatrix}x(t) < 0\end{cases}\\
&x(t) \in \begin{bmatrix}-1 & 1\end{bmatrix}\times\begin{bmatrix}0 & 1.5\end{bmatrix},\quad u(t) \in \begin{bmatrix}-1 & 1\end{bmatrix}
\end{aligned}
$$

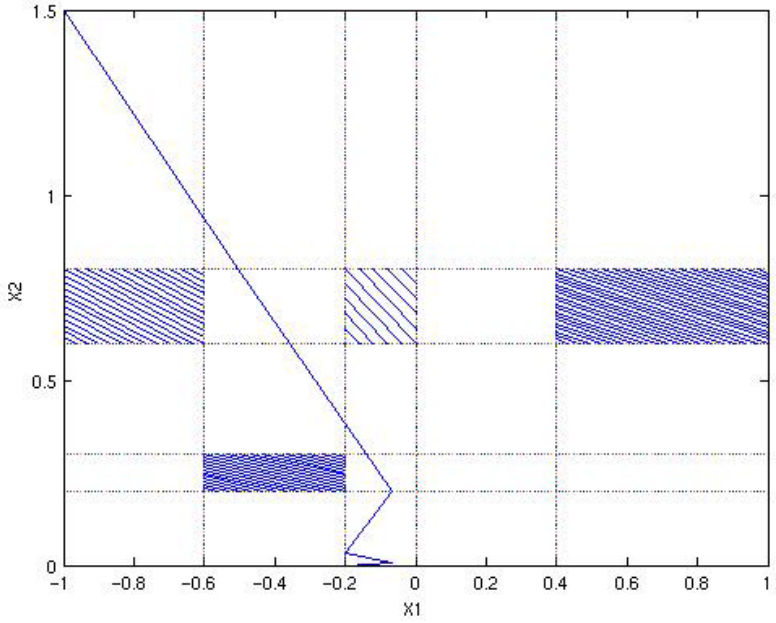
4.4.2 例1

初期データはつぎのとおりである．

```
$x0 = "-1";
$y0 = "1.5";
@area     = ("-1", "1", "0", "1.5");
%bad = (
            "0",        ["0.4", "1", "0.6", "0.8"],
            "1",        ["-1", "-0.6", "0.6", "0.8"],
            "2",        ["-0.6", "-0.2", "0.2","0.3"],
            “3”,        [“-0.2”, “0”, “0.6”, “0.8”]
) ;
```

結果のグラフでの表現を図 4-8 に示す．

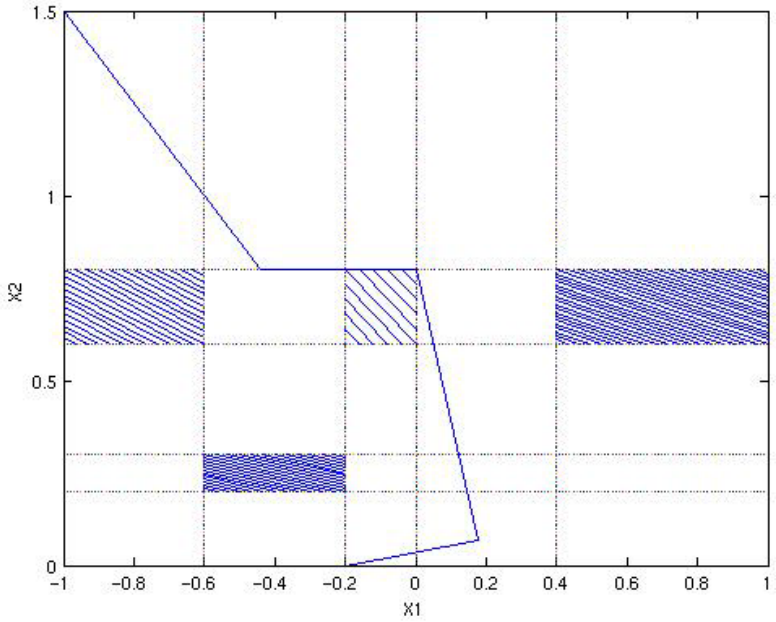
新



最適解　　6.642685

計算時間　　0.3416 sec.

旧



最適解　　8.27493

計算時間　　0.8111 sec.

図 4-8 実行例1のグラフ表現

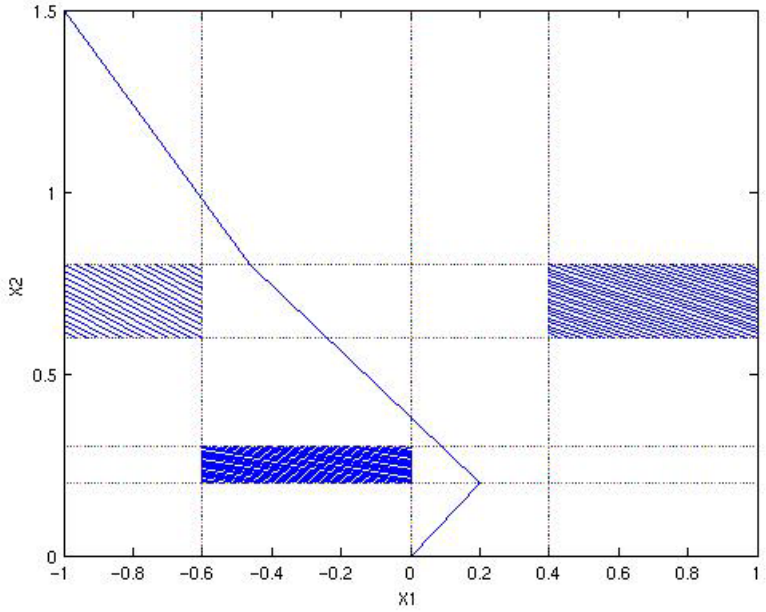
例2

初期データはつぎのとおりである.

```
$x0      = "-1";
$y0      = "1.5";
@area    = ("-1", "1", "0", "1.5");
%bad     = (
              "0",   ["-1", "-0.6", "0.6", "0.8"],
              "1",   ["-0.6", "0", "0.2", "0.3"],
              "2",   ["0.4", "1", "0.6", "0.8"]
 );
```

結果のグラフでの表現を図 4−9 に示す.

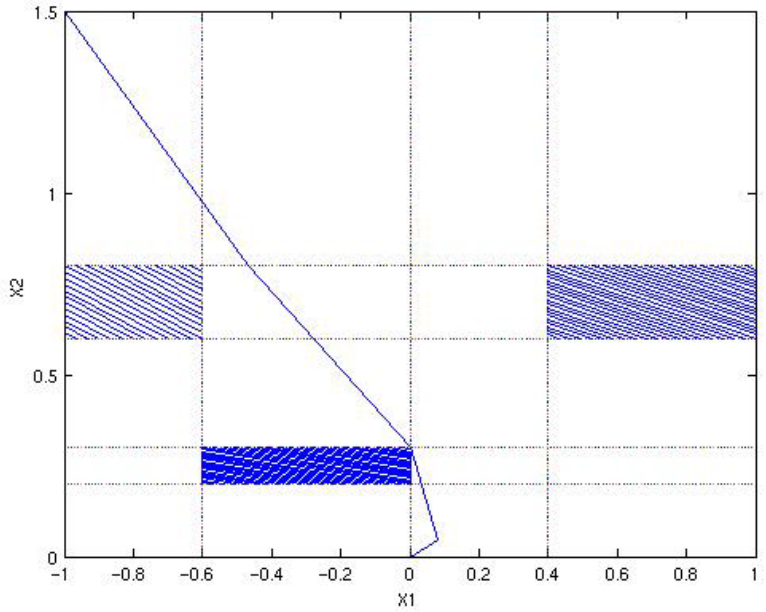
新



最適解　　6.908611

計算時間　　0.2079 sec.

旧



最適解　　6.948735

計算時間　　1.0902 sec.

図 4-9 実行例2のグラフ表現

例3

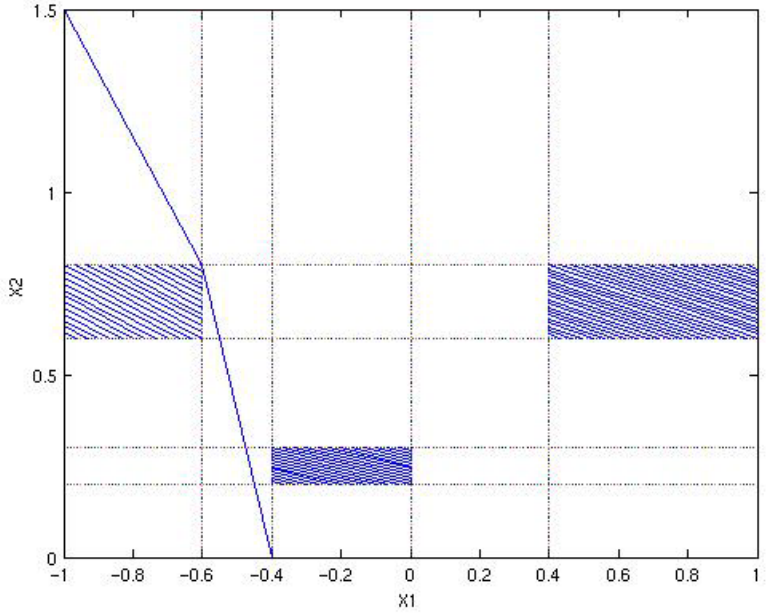
初期データはつぎのとおりである.

```
$x0      = "-1";
$y0      = "1.5";
@area    = ("-1", "1", "0", "1.5");
%bad     = (
              "0",   ["-1", "-0.6", "0.6", "0.8"],
              "1",   ["-0.4", "0", "0.2", "0.3"],
              "2",   ["0.4", "1", "0.6", "0.8"]
 );
```

結果のグラフでの表現を図 4-10 に示す.

前の二つの例題と異なって, この例で新旧手法の計算結果は所要時間が違うであるが, 計算した最適経路の中に新しい種類の遷移を含まれていないため最適解が同じである.

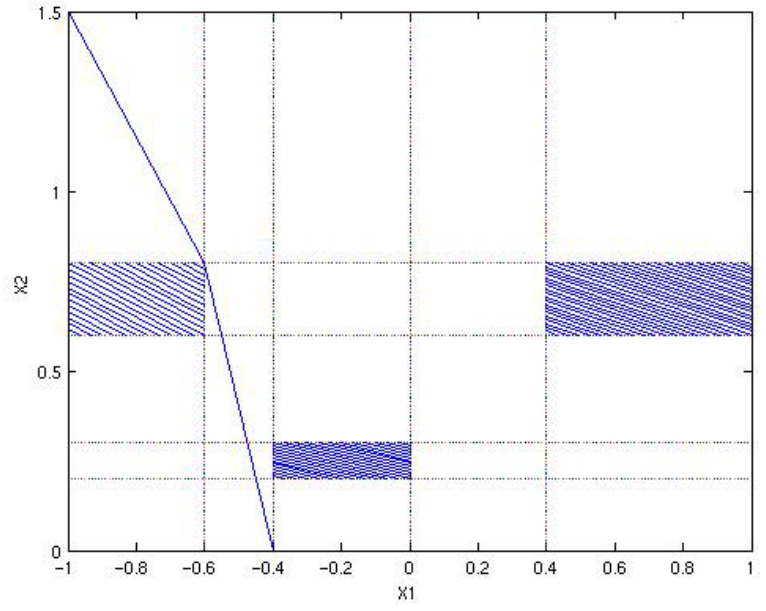
新



最適解　　6.168734

計算時間　　0.2938 sec.

旧



最適解　　6.168734

計算時間　　0.6071 sec.

図 4-10 実行例3のグラフ表現

4.4.3 CLP ソースコードの実行手順について

本研究の CLP ソースコードの実行に関しては，以下の2回の操作を行う．

一回目（最小バウンド計算）：

- CLP 環境で make_clp.pl で作成したファイル clp を読み込んで，コマンド“all_lb(ステップ数)”を実行して，実行結果をテキストファイルに保存する．
- 実行結果ファイルを入力とし，スクリプト make_bound.pl を使って CLP で扱える形式に直し，その結果を bound というテキストファイルに保存しておく．

二回目（経路探索）：

- CLP 環境で，clp と bound の2つのファイルを読み込んで，コマンド“go(ステップ数)”を実行すると，最適経路が出力される．
- 実行結果を保存して，後述の可視化ツールの入力ファイルにする．

以上の手続きで，CLP プログラムを使用した最適経路の探索が終る．

4.5 評価

4.5.1 比較方法

同じ計算機環境，同じ問題（初期データ）を用い，既存研究による計算結果（最適解と所要時間）と本研究で提案した新しいタイプの遷移を使って計算した結果（最適解と所要時間）を比較する．

4.5.2 評価の結果

前節の実行例から，本研究の手法で求めた最適解が明らかに既存研究より精度の良い結果が出た．更に所要時間が少なくなった．

本研究は新しい遷移を提案したが，従来研究の遷移も全部残しているから，従来研究と同じの解を求められる上で，より良いパスを計算することが可能である．

それに，4.4.2 で与えた例1の新旧手法を用いて，探索の過程を示す探索ツリー図 4-11 で表している．新手法のグラフに丸を付いた所は旧グラフに丸を付いた所をバックトラックされた．探索のサイズは従来法より小さくなったから，所要時間も短縮された．

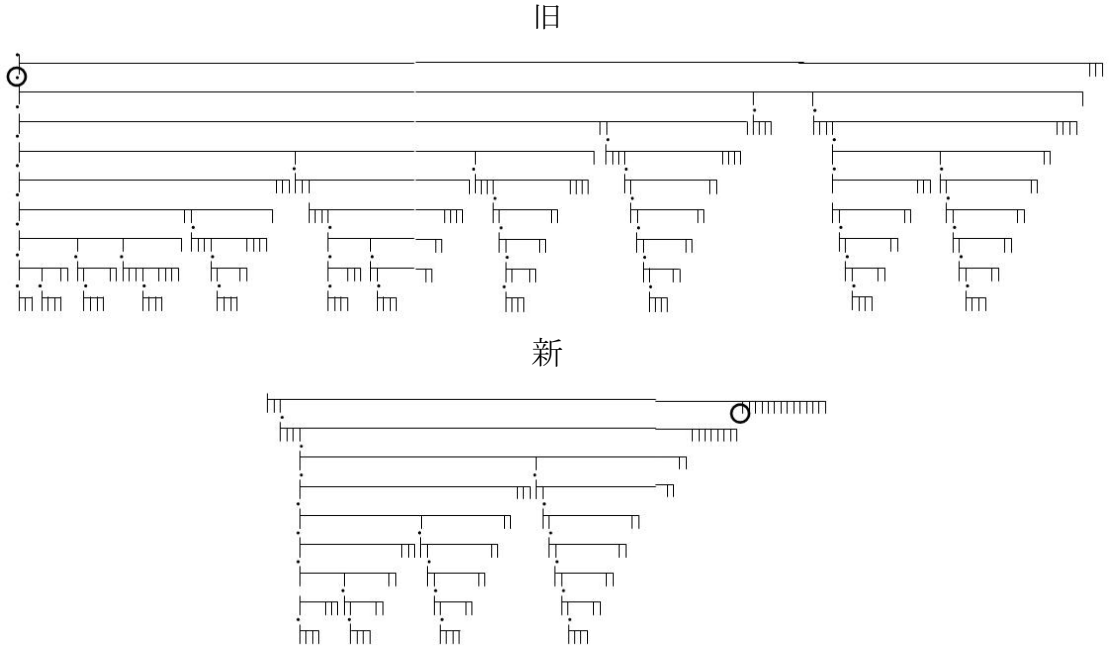


図 4-11 新旧探索ツリーのグラフ

また，CLPは非手続き型言語であるため，実際にパスを探索する時にCLPソースコードに書いてある遷移の順番に従って探索を行う．なので，従来法でも本研究の手法でも所要時間はその順番に強く依存する．順番を変更すると探索の順番も変更するから所要時間も変わる．例えば，4.4 の三つの例の遷移の順番を逆にすると，同じの問題に対して順番を逆にした後，新旧手法とも所要時間が変わった．変更前は本研究の手法のほうが早いが，変更後はより遅いである．

表 4-1 逆順にした前後の所要時間

		例1		例2		例3	
		新	旧	新	旧	新	旧
総遷移数		209	88	132	58	234	110
逆順前	前処理	25.6151	11.4865	8.2218	5.0469	19.6976	9.0729
	パス探索	0.3416	0.8111	0.2079	1.0902	0.2938	0.6071
逆順後	前処理	52.0306	9.4448	22.2665	6.2523	42.4745	11.4377
	パス探索	4.2407	0.9897	1.7285	0.896	2.3463	1.1751

以上で，本研究で提案した手法は精度の面において，既存研究より本研究で提案した新手法がより良いことが分かる．

4.6 評価補助ツール

本研究の評価補助ツールは，CLP プログラムで求めた最適パス座標の数字データをグラフ表現する可視化ツールのことである．

CLPプログラムによって算出した最適パスは，図 4-12 のように各ステップの横・縦座標の数字の羅列で，どのようなパスを表しているのか分かりづらく，評価もしにくい．

```
KCLP Interpreter for Hybrid Systems
Version 0.92(9/12/2005)
(C) Kunihiko Hiraishi, JAIST
with POLKA: Convex Polyhedra Library

:- seteps(0.000001).

*** yes ***

0.0000 sec.

Consulted : try.

0.0010 sec.

| ?- go(10).
JOPT = 6.587261
Time =    0: Loc = l1_1, X1 = -1.000000, X2 = 1.500000, U = -0.158494
Time =    1: Loc = l0_2, X1 = 0.480737, X2 = 1.134327, U = 0.013205
Time =    2: Loc = l1_1, X1 = -0.580385, X2 = 0.800000, U = -0.722102
Time =    3: Loc = l1_3, X1 = -0.400000, X2 = -0.000000, U = -0.000520
Time =    4: Loc = l1_5, X1 = -0.160520, X2 = 0.276608, U = -0.127432
Time =    5: Loc = l0_5, X1 = -0.000000, X2 = 0.094423, U = 0.025434
Time =    6: Loc = l1_5, X1 = -0.039984, X2 = 0.063203, U = -0.027795
Time =    7: Loc = l0_5, X1 = 0.000000, X2 = 0.025188, U = 0.011634
Time =    8: Loc = l1_5, X1 = -0.005817, X2 = 0.021709, U = -0.012714
Time =    9: Loc = l0_5, X1 = 0.000000, X2 = 0.000000, U = -0.000000

*** yes ***

0.2167 sec.
```

図 4-12 CLP プログラムで算出した最適パスの座標

そこで，本研究では，簡単にそれらの数字を Matlab の M-ファイルに変換する make_matlab.pl という Perl のスクリプトを作成した．これにより，4.4 の実行例で示したグラフのように，直観に結果を把握することができる．

スクリプト make_matlat.pl の使い方について

```
# 環境変数の定義
$cell     = 200;
$sharp    = 0.03;
@area     = ("-1", "1", "0", "1.5");
%bad = (
            "0",        ["0.4", "1", "0.6", "0.8"],
            "1",        ["-1", "-0.6", "0.6", "0.8"],
            "2",        ["-0.6", "-0.2", "0.2", "0.3"],
            "3",        ["-0.2", "0", "0.6", "0.8"]
);
```

スクリプト make_clp.pl と同じく初期データの編集は Perl のヘッドファイル（init.pm）を介して行っている．一番上に用意してあるのはグラフの解像度制御変数である．ここで，配列@area は領域全体の範囲である．ハッシュ%bad は各障害物の範囲である．環境が変わるたびに，この二つの変数を編集すれば，新しい環境に応じたグラフを描くことができる．入出力に関しては，以下のファイルを使用する．

- 入力ファイル： スクリプト CLP の実行結果を保存したテキストファイル．
- 出力ファイル： Matlab で扱える“out_入力ファイル”という M－ファイル．

第５章　おわりに

本研究は，ハイブリッドシステムにおける障害物を回避する最適経路問題に関して，PWL＋CLP の手法を用いた新解法で，既存研究で扱っていた例題を解き，既存研究の結果と比較した．

本研究は，一部分の考慮されていなかった遷移を有効にすることで解の精度を向上した．また，従来研究では手計算で行っていたソースコードの作成作業を自動化した．

今後の課題としては，状態空間が3次元以上の多次元の場合の対応が考えられる．

参考文献

[1]A. Bemporad and M. Morari, “Control of systems integrating logic, dynamics and Constraints” Automatica, Vol.35, No.3, pp.407-427, (1999).

[2]K. Hiraishi, “Computation of a Class of Hybrid Systems by Graph Exploration”, Proc. WODES06 pp.282-287, (2006).

[3]R.Alur, et al.,The algorithmic analysis of hybrid systems, Theoretical Computer Sciences 138, 3/34, (1995).

[4]電子情報通信学会第 3 種研究会「ハイブリッドダイナミカルシステム論理とその応用」 http://ushiolab.sys.es.osaka-u.ac.jp/hds

[5]特集 Quantifier Elimination,数式処理, 第 10 巻, 第 1 号, (2003).

[6]田村直之, Prolog 入門, http://bach.istc.kobe-u.ac.jp/prolog/intro/

[7]平石, 石川, “制御論理プログラミングによるハイブリッドシステムのパラメータ設計” 第 17 回回路とシステム軽井沢ワークショップ, pp.597-606, (2004).

謝辞

本研究を進めるにあたり，終始熱心に御指導を頂いた平石邦彦教授に感謝の意を表し，心より御礼申し上げます．また，本論文をまとめるにあたって様々な面で御協力いただいた平石研究室の皆様に心より感謝します．

付録 A

以下は本研究で用いたプログラムを表す.

A.1 make_clp.pl

```
#!/usr/local/bin/perl
require "init.pm";
$outfile = "clp";
open(OUT,">$outfile");
print      "output: $outfile¥n";
print OUT "/* by tokoyi */¥n";
make_regions(%bad);
make_souce();
close(OUT);

sub make_souce {
        head();
        inv_minmax_vertex();
        edge();
        activity();
        update();
        next_l_and_next2_l2();
        global();
        type();
        get_ab_limit();
        aux();
        go();
        bound_head();
}

sub make_regions {
        %_ = @_;
        my ($i,$j);
        @v = ($area[0],$area[1]);
        @h = ($area[2],$area[3]);
        foreach $i (sort keys %_){
                for($j=0;$j<@{$_{$i}};$j+=2){
                        @v = (@v,$_{$i}[$j],$_{$i}[$j+1])    if $j eq "0";
                        @h = (@h,$_{$i}[$j],$_{$i}[$j+1])    if $j eq "2";
                }
        }
        @v = (@v,"0");
        @v = sort_and_distinct(@v);
        @h = sort_and_distinct(@h);
        regions();
}
sub sort_and_distinct {
        @_ = sort by_number @_;
```

```
	my ($i,$n,@temp);
	$n = 0;
	for($i=0;$i<@_;$i++){
		$temp[$n++] = $_[$i]	if($_[$i-1] ne $_[$i]);
	}
	return @temp;
}
sub regions {
	$n = 0;
	my ($i,$j,$key,$m);
	for($i=0;$i<@v-1;$i++){
		for($j=0;$j<@h-1;$j++){
			@temp = ($v[$i],$v[$i+1],$h[$j],$h[$j+1]);
			@{$cell{"loc".$n++}} = @temp		if(noequal(@temp));
		}
	}
	foreach $key (sort keys %cell){
		print $key." : ";
		for($m=0;$m<@{$cell{$key}};$m++){
			print $cell{$key}[$m].",";
		}
		print "¥n";
	}
}
sub noequal {
	my ($i,$j,$flag);
	foreach $i (sort keys %bad){
		$flag = 0;
		for($j=0;$j<@_;$j++){
			$flag++		if($_[$j] eq $bad{$i}[$j]);
		}
		if($flag eq "4"){
			return 0;
			exit;
		}
	}
	return 1;
}
sub by_number {
	$a <=> $b;
}
sub head {
	my @lines = ("
/*
	Region to be avoided: 0.4 < x1 $ 0.3 < x2 <0.8 in location 0

	We separate locations as follows:
*/
");
	print OUT @lines;
}
sub inv_minmax_vertex {
	my ($key);
	foreach $key (sort keys %cell){
		print OUT "inv(".$key.", X1, X2):- ".$cell{$key}[0]." <= X1, X1 < ".$cell{$key}[1].",
".$cell{$key}[2]." <= X2, X2 <= ".$cell{$key}[3].".¥n";
	}
	print OUT "¥n";
	foreach $key (sort keys %cell){
```

```
                print OUT "minmax(".$key." : ".$cell{$key}[0].", ".$cell{$key}[1].", ".$cell{$key}[2].",
".$cell{$key}[3].").¥n";
        }
        print OUT "¥n";
        foreach $key (sort keys %bad){
                print OUT "vertex(b".$key." : [00, ".$bad{$key}[0].", ".$bad{$key}[2].", 01, ".$bad{$key}[0].",
".$bad{$key}[3].", 10, ".$bad{$key}[1].", ".$bad{$key}[2].", 11, ".$bad{$key}[1].", ".$bad{$key}[3].]).¥n";
        }
}
sub edge {
        my ($i,$j,$k,$n,$loc_i,$loc_j,$num,@lines,@list);
        @lines = ("
/*
        Transition relation:
*/

");
        print OUT @lines;
        $n = 0;
        for($i=0;$i<(keys%cell);$i++){
                $loc_i = "loc".$i;
                for($j=0;$j<(keys %cell);$j++){
                        $loc_j = "loc".$j;
                        next                    if ($loc_j eq $loc_i || $cell{$loc_i}[3]<$cell{$loc_j}[3]);
                        ($num,@list) = num_and_list($loc_i,$loc_j);
                        $temp = "[".@list[0];
                        for($k=1;$k<@list;$k++){ $temp = $temp.",".$list[$k]; }
                        $temp = $temp."]";
                        next
                if $num eq "5";
                        print OUT "edge(".$loc_i.", ".$loc_j." ,0, []).¥n"
        if $num eq "";
                        print OUT "edge(".$loc_i.", ".$loc_j." ,1, ".$temp.").¥n"       if $num ge "1";
                        $n++;
                }
        }
        print OUT "/* Total : $n edges.*/¥n";
}
sub num_and_list {
        my ($i,$j,$n,$k,$flag,@a,@b,@new,@xy,@list,@back,%num);
        @a = @{$cell{$_[0]}};
        @b = @{$cell{$_[1]}};
        @new = (min($a[0],$b[0]),max($a[1],$b[1]),min($a[2],$b[2]),max($a[3],$b[3]));
        make_bad_vers();
        foreach $i (sort keys %bad){
                if( ($a[0] eq $bad{$i}[0] && $b[0] eq $bad{$i}[0] ) && ( $a[1] eq $bad{$i}[1] && $b[1] eq
$bad{$i}[1] ) ){
                        $num{$i} = "5";
                        @list = ();
                        next;
                }
                if( ($a[2] eq $bad{$i}[2] && $b[2] eq $bad{$i}[2] ) && ( $a[3] eq $bad{$i}[3] && $b[3] eq
$bad{$i}[3] ) ){
                        $num{$i} = "5";
                        @list = ();
                        next;
                }
                for($j=0;$j<@ver;$j++){
                        @xy = @{${"b".$i}{$ver[$j]}};
```

```
                                        if($new[0]<$xy[0] && $xy[0]<$new[1] && $new[2]<$xy[1] && $xy[1]<$new[3]){
                                                $num{$i}++;
                                                @list = (@list,"b".$i,$ver[$j]);
                                        }
                                }
                                if($num{$i} eq "2"){
                                        $back[3]=pop(@list);                                            $back[2]=pop(@list);
$back[1]=pop(@list);$back[0]=pop(@list);
                                        $flag = better($_[0],$_[1],$back[0],$back[1],$back[3]);
                                        @list = (@list,$back[0],$back[1])   if $flag eq "true";
                                        @list = (@list,$back[2],$back[3])   if $flag eq "false";
                                }
                                if($num{$i} eq "4"){
                                        for($k=0;$k<8;$k++){ $back[7-$k] = pop(@list); }
                                        $flag = four_2_two($_[0],$_[1]);
                                        @list = (@list,$back[0],$back[1],$back[6],$back[7])          if $flag eq "1";
                                        @list = (@list,$back[2],$back[3],$back[4],$back[5])          if $flag eq "2";
                                }
                        }
                        $n = max(values %num);
                        return ($n,@list);
}
sub min {
                        my $ans = $_[0] < $_[1] ? $_[0] : $_[1];
                        return $ans;
}
sub max {
                        my $ans = $_[0] > $_[1] ? $_[0] : $_[1];
                        return $ans;
}
sub abs {
                        my $ans = $_[0] >= 0 ? $_[0] : $_[0] * (-1);
                        return $ans;
}
sub better {
                        my ($loc,@a,@b,@c);
                        @a = @{@{$_[2]}{$_[3]}};
                        @b = @{@{$_[2]}{$_[4]}};
                        $loc = nearloc(@_);
                        @c = ($cell{$loc}[0],$cell{$loc}[2]);
                        if(abs($a[0]-$c[0])<abs($b[0]-$c[0]) || abs($a[1]-$c[1])<abs($b[1]-$c[1])){
                                return true;
                        }
                        else{           return false; }
}
sub nearloc {
                        my ($i,$b,$n,$keyword,$loc);
                        $n = $_[3] + $_[4];
                        $i = "1"        if $n eq "1";
                        $i = "0"        if $n eq "21";
                        $i = "3"        if $n eq "10";
                        $i = "2"        if $n eq "12";
                        $b = $_[2];
                        $b =~ s/b//g;
                        $keyword = $bad{$b}[$i];
                        $loc = $_[0]                if($cell{$_[0]}[$i] eq $keyword);
                        $loc = $_[1]                if($cell{$_[1]}[$i] eq $keyword);
                        return $loc;
}
```

```
sub make_bad_vers {
		my ($key,@a);
		foreach $key (sort keys %bad){
			@a = @{$bad{$key}};
			%{"b".$key} = ("00",[$a[0],$a[2]],"01",[$a[0],$a[3]],"10",[$a[1],$a[2]],"11",[$a[1],$a[3]]);
		}
}
sub four_2_two {
		my $ans = $cell{$_[0]}[0] < $cell{$_[1]}[0] ? "1" : "2";
		return $ans;
}
sub activity {
		my ($key,@lines);
		@lines = ("
/*
		continuous dynamics
*/

");
		print OUT @lines;
		foreach $key (sort keys %cell){
			print OUT "activity(".$key." ,0).¥n"		if $cell{$key}[1] <= 0;
			print OUT "activity(".$key." ,1).¥n"		if $cell{$key}[1] >    0;
		}
		print OUT "¥n";
}
sub update {
		my ($key);
		foreach $key (sort keys %cell){
			print OUT "update(".$key.", T, L, [[1, X1], [1, X2], [1, U1], [1, U2] | L]):-¥n";
			print OUT "¥tx1(T : X1), x2(T : X2), u1(T : U1), u2(T : U2), -1 <= U1, U1 <= 1, -1 <= U2, U2
<= 1,¥n";
			print OUT "¥ta(0 : A11, A12, A21, A22),¥n"	if $cell{$key}[1] <= 0;
			print OUT "¥ta(1 : A11, A12, A21, A22),¥n"	if $cell{$key}[1] >    0;
			print OUT "¥tx1(T + 1 : A11 * X1 + A12 * X2 + U1),¥n";
			print OUT "¥tx2(T + 1 : A21 * X1 + A22 * X2 + U2).¥n";
		}
}
sub next_l_and_next2_l2 {
		my @lines = ("
/*
		main
*/

next(_, T, L, J):- final_time(: T), !, qmin_list(L, [], J), store_solution(J).
next(Loc, T, L, J):-
		final_time(: H), lb(Loc, H - T, LB), number(LB),
		(optval, !,
			qmin_list(L, [], JTEMP, _),
			JTEMP + LB <= ¥@optval * 0.95;
		true),
		x1(T : X1), x2(T : X2), !,
		edge(Loc, LocN, Type, BlocList),
		maxX1(0 : -1),
		minX1(0 : -1),
		maxX1(T : MAXX1),
		minX1(T : MINX1),
		maxX1(T - 1 : MAXX1pre),
		minX1(T - 1 : MINX1pre),
```

```
		type(Type, T, Loc, LocN, BlocList, MAXX1pre, MINX1pre, MAXX1,MINX1),

		inv(LocN, X1, X2),
		l(LocN, T, L, J).

l(Loc, T, L, J):-
		activity(Loc, A),
		a(A : A11, A12, A21, A22),
		x1(T : X1), x2(T : X2),
		max(M, M = A11 * X1 + A12 * X2, MAXX1),
		min(M, M = A11 * X1 + A12 * X2, MINX1),
		maxX1(T + 1 : MAXX1),
		minX1(T + 1 : MINX1),

		update(Loc, T, L, L1),
		loc(T : Loc),
		next(Loc, T + 1, L1, J).
");

		my @lines2 = ("
/* for all_lb(), without LB */
next2(_, T, L, J):- final_time(: T), !, qmin_list(L, [], J), store_solution(J).
next2(Loc, T, L, J):-
		final_time(: H),
		(optval, !,
			qmin_list(L, [], JTEMP, _),
			JTEMP <= ¥@optval * 0.95;
		true),
		x1(T : X1), x2(T : X2), !,
		edge(Loc, LocN, Type, BlocList),
		maxX1(0 : -1),
		minX1(0 : -1),
		maxX1(T : MAXX1),
		minX1(T : MINX1),
		maxX1(T - 1 : MAXX1pre),
		minX1(T - 1 : MINX1pre),
		type(Type, T, Loc, LocN, BlocList, MAXX1pre, MINX1pre, MAXX1,MINX1),

		inv(LocN, X1, X2),
		l2(LocN, T, L, J).

l2(Loc, T, L, J):-
		activity(Loc, A),
		a(A : A11, A12, A21, A22),
		x1(T : X1), x2(T : X2),
		max(M, M = A11 * X1 + A12 * X2, MAXX1),
		min(M, M = A11 * X1 + A12 * X2, MINX1),
		maxX1(T + 1 : MAXX1),
		minX1(T + 1 : MINX1),

		update(Loc, T, L, L1),
		loc(T : Loc),
		next2(Loc, T + 1, L1, J).
");


		print OUT @lines;
		print OUT @lines2;
}
```

```
sub global {
		my @lines = ('
/*
		global vars
*/

final_time(: _).
x1(_ : X1).
x2(_ : X2).
maxX1(_ : _).
minX1(_ : _).
maxX1(0 : -1).
minX1(0 : -1).
u1(_ : U1).
u2(_ : U2).
loc(_ : _).
a(0 : 0.8 * ¥@cos(¥@pi / 3), 0.8 * (- ¥@sin(¥@pi /3)), 0.8 * ¥@sin(¥@pi / 3), 0.8 * ¥@cos(¥@pi /3)).
a(1 : 0.8 * ¥@cos(- ¥@pi / 3), 0.8 * (- ¥@sin(- ¥@pi / 3)), 0.8 * ¥@sin(- ¥@pi / 3), 0.8 * ¥@cos(- ¥@pi / 3)).
');
		print OUT @lines;

}
sub type {
		my @lines = ('
/*
		type
*/
');
		print OUT @lines;
		print OUT "type(0, _, _, _, _, _, _, _).¥n¥n";
		print OUT "type(1, T, Loc, _, [], MAXX1pre, MINX1pre, _, _).¥n";
		for(my $i=0;$i<@ver;$i++){
			print OUT "type(1, T, Loc, LocN, [Bloc, ".$ver[$i]." | Next], MAXX1pre, MINX1pre, MAXX1,
MINX1):-¥n";
			print OUT "¥tx1(T : X1), x2(T : X2),¥n";
			print OUT "¥tx1(T - 1 : X1pre), x2(T - 1 : X2pre),¥n";
			print OUT "¥tvertex(Bloc : L),¥n";
			print OUT "¥tget_ab(".$ver[$i].", L, A, B),¥n";
			if($ver[$i] eq "00" || $ver[$i] eq "10"){
				print OUT "¥tminmax(Loc : E, F, _, MAX),¥n";
				#print OUT "¥tlimit(MAXX1pre, MAX, A, B, X1, X2max),¥n"
		if $ver[$i] eq "00";
				print OUT "¥tchoice_max(MAXX1pre, F, Q),¥n"                    if   $ver[$i]
eq "00";
				print OUT "¥tlimit(Q, MAX, A, B, X1, X2max),¥n"
		if $ver[$i] eq "00";
				#print OUT "¥tlimit(MINX1pre, MAX, A, B, X1, X2max),¥n"
		if $ver[$i] eq "10";
				print OUT "¥tchoice_min(E, MINX1pre, Q),¥n"          if $ver[$i] eq "10";
				print OUT "¥tlimit(Q, MAX, A, B, X1, X2max),¥n"
		if $ver[$i] eq "10";

				print OUT "¥t(¥n";
				print OUT "¥t¥tX2 <= X2max,¥n";
			}
			if($ver[$i] eq "01" || $ver[$i] eq "11"){
				print OUT "¥tminmax(Loc : E, F, MIN, _),¥n";
				#print OUT "¥tlimit(MAXX1pre, MIN, A, B, X1, X2max),¥n"
		if $ver[$i] eq "01";
```

```
                                        print OUT "¥tchoice_max(MAXX1pre, F, Q),¥n"                             if   $ver[$i]
eq "01";
                                        print OUT "¥tlimit(Q, MIN, A, B, X1, X2max),¥n"
        if $ver[$i] eq "01";
                                        #print OUT "¥tlimit(MINX1pre, MIN, A, B, X1, X2max),¥n"
        if $ver[$i] eq "11";
                                        print OUT "¥tchoice_min(E, MINX1pre, Q),¥n"             if $ver[$i] eq "11";
                                        print OUT "¥tlimit(Q, MIN, A, B, X1, X2max),¥n"
        if $ver[$i] eq "11";

                                        print OUT "¥t(¥n";
                                        print OUT "¥t¥tX2 >= X2max,¥n";
                }
                print OUT "¥t¥ttype(1, T, Loc, LocN, Next, MAXX1pre, MINX1pre, MAXX1, MINX1);¥n";
                print OUT "¥t¥ttype(2, T, Loc, LocN, [Bloc, ".$ver[$i]." | Next], MAXX1pre, MINX1pre,
MAXX1, MINX1)¥n";
                print OUT "¥t).¥n";
        }
        print OUT "¥n";
        print OUT "type(2, T, _, LocN, [], MAXX1pre, MINX1pre, MAXX1, MINX1).¥n";
        for( my $i=0;$i<@ver;$i++){
                print OUT "type(2, T, Loc, LocN, [Bloc, ".$ver[$i]." | Next], MAXX1pre, MINX1pre, MAXX1,
MINX1):-¥n";
                print OUT "¥tx1(T : X1), x2(T : X2),¥n";
                print OUT "¥tx1(T - 1 : X1pre), x2(T - 1 : X2pre),¥n";
                print OUT "¥tvertex(Bloc : L),¥n";
                print OUT "¥tget_ab(".$ver[$i].", L, A, B),¥n";
                if($ver[$i] eq "00" || $ver[$i] eq "10"){
                        print OUT "¥tminmax(LocN : E, F, _, MAX),¥n";
                        #print OUT "¥tlimit(MAXX1, MAX, A, B, MAXX1pre, X2premin),¥n"    if   $ver[$i]
eq "00";
                        print OUT "¥tchoice_max(MAXX1, F, Q),¥n"
        if $ver[$i] eq "00";
                        print OUT "¥tlimit(Q, MAX, A, B, MAXX1pre, X2premin),¥n"                if   $ver[$i]
eq "00";
                        #print OUT "¥tlimit(MINX1, MAX, A, B, MAXX1pre, X2premin),¥n"    if   $ver[$i]
eq "10";
                        print OUT "¥tchoice_min(E, MINX1, Q),¥n"
        if $ver[$i] eq "10";
                        print OUT "¥tlimit(E, MAX, A, B, MAXX1pre, X2premin),¥n"                if   $ver[$i]
eq "10";

                        print OUT "¥tX2pre <= X2premin,¥n";
                }
                if($ver[$i] eq "01" || $ver[$i] eq "11"){
                        print OUT "¥tminmax(LocN : E, F, MIN, _),¥n";
                        #print OUT "¥tlimit(MAXX1, MIN, A, B, MAXX1pre, X2premin),¥n"    if   $ver[$i]
eq "01";
                        print OUT "¥tchoice_max(MAXX1, F, Q),¥n"
        if $ver[$i] eq "01";
                        print OUT "¥tlimit(Q, MIN, A, B, MAXX1pre, X2premin),¥n"                if   $ver[$i]
eq "01";
                        #print OUT "¥tlimit(MINX1, MIN, A, B, MAXX1pre, X2premin),¥n"    if   $ver[$i]
eq "11";
                        print OUT "¥tchoice_min(E, MINX1, Q),¥n"
        if $ver[$i] eq "11";
                        print OUT "¥tlimit(Q, MIN, A, B, MAXX1pre, X2premin),¥n"                if   $ver[$i]
eq "11";
                        print OUT "¥tX2pre >= X2premin,¥n";
```

```
                        }
                        print OUT "¥ttype(1, T, Loc, LocN, Next, MAXX1pre, MINX1pre, MAXX1, MINX1).¥n";
                }
}
sub get_ab_limit {
                my (@lines);
                @lines = ('
get_ab(Ver, []).
get_ab(Ver, [V, X1, X2 | R], A, B):-
                Ver = V,
                A = X1, B = X2;
                get_ab(Ver, R, A, B).

limit(X1pre, X2pre, A, B, X1, X2):-
                (X1pre - A) * (B - X2) = (A - X1) * (X2pre - B).

choice_min(E,MIN,ANS):- E < MIN, ANS = MIN; ANS = E.
choice_max(MAX,F,ANS):- MAX < F, ANS = MAX; ANS = F.
');
                print OUT @lines;
}
sub aux {
                my (@lines);
                @lines = ('
/*
                aux
*/

store_solution(J):- getval(0, RES), record(0, [J, RES]).

getval(T, []):- final_time(: F), T >= F, !.
getval(T, [[T, L, X1, X2, U1, U2] | R]):-
                x1(T : X1), x2(T : X2), u1(T : U1), u2(T : U2), loc(T : L),
                getval(T + 1, R).

ans([]).
ans([[T, Loc, X1, X2, U1, U2] | R]):-
                print(¥"Time = %2.0Lf: ¥", T),
                write(¥"Loc = ¥"), write(Loc), write(¥", ¥"),
                print(¥"X1 = %Lf, ¥", X1),
                print(¥"X2 = %Lf, ¥", X2),

                print(¥"U1 = %Lf, ¥", U1),
                print(¥"U2 = %Lf¥", U2), nl,
                ans(R).
');
                print OUT @lines;
}
sub go {
                my @lines = ('
/*
                go
*/

go(H):-
                clear_records,
                final_time(: H),
                x1(0 : $x0), x2(0 : $y0),
                x1(H : 0), x2(H : 0),
```

```
		min(J, l(".find_loc($x0,$y0)."', 0, [], J), JOPT),
		write(¥"JOPT = ¥"), write(JOPT), nl,
		recorded(0, [JOPT, RES]), ans(RES).

go_lb(Loc, H):-
		clear_records,
		final_time(: H),
		x1(0 : X1), x2(0 : X2), inv(Loc, X1, X2),
		x2(H : 0), x2(H : 0),
		min(J, l2(Loc, 0, [], J), JOPT),
		write(¥"lb(¥"), write(Loc), write(¥", ¥"),
		write(H), write(¥", ¥"), write(JOPT), write(¥"),¥"), nl, !.

all_lb(0).
all_lb(H):-
		location(Loc), go_lb(Loc, H), fail.
all_lb(H):- all_lb(H - 1).

");

		my $i;
		for $i (sort keys %cell) {
			print OUT "location($i).¥n";
		}

		print OUT @lines;
}
sub find_loc {
		my ($i,$x,$y);
		$x = $_[0];
		$y = $_[1];
		for $i (sort keys %cell){
			if($cell{$i}[0]<=$x && $x<$cell{$i}[1] && $cell{$i}[2]<=$y && $y<=$cell{$i}[3]){
				return $i;
				last;
			}
		}
}
sub bound_head {
		my @lines = ("
:- seteps(0.000001).

/*
		lowre bound
*/

lb(Loc, H, LB):- H > 6, lb(Loc, 6, LB).

");
		print OUT @lines;
}
```

A.2 make_bound.pl

```
#!/usr/local/bin/perl
```

```
print "Input file :";
chomp($filename = <STDIN>);
open(INPUT,"$filename") || die "$filename: $!";
unless(-r $filename) {
        print "$filename: can't read!¥n";
        exit;
}
close(INPUT);
$outfile = "bound";
print "==> output:    $outfile¥n¥n";

open(IN,$filename);
open(OUT,">$outfile");
@lines = <IN>;

for($i=0;$i<@lines;$i++){
        if($lines[$i] =~ /^lb¥(loc/){
                $lines[$i] =~ s/¥),/¥)./;
                print OUT $lines[$i];
        }
}
close(IN,OUT);
```

A.3 make_matlab.pl

```
#!/usr/local/bin/perl
require "init.pm";
print "Input file :";
chomp($filename = <STDIN>);
open(INPUT,"$filename") || die "$filename: $!";
unless(-r $filename){
        print "$filename: can't read!¥n";
        exit;
}
close(INPUT);
  ($subname) = split(/¥W/,$filename);
$funcname= "out_".$subname;
$outfile = "out_".$subname.".m";
print "¥n==> input : $filename¥n";
print      "==> output: $outfile¥n¥n";

open(IN,$filename);
open(OUT,">$outfile");
@lines = <IN>;

print OUT "% by tokoyi¥n";
print OUT "axis([".$area[0]." ".$area[1]." ".$area[2]." ".$area[3]."]); box;¥n";
print OUT "hold on;¥n";
print OUT "title(' '),xlabel('X1'),ylabel('X2'),¥n";

draw_bg(%bad);
get_and_draw_xy(@lines);

print OUT "hold off;¥n";
```

```
close(IN,OUT);

system ("/pkg/all/bin/matlab -r $funcname");

sub get_and_draw_xy {
	@lines = @_;
	my $n = 0;
	for($i=0;$i<@lines;$i++){
		if($lines[$i] =~ /^JOPT =/) {
			@temp = split(/=/, $lines[$i]);
			chomp(@temp);
			$jopt = $temp[1];
			$jopt =~ s/ //g;
		}
		if($lines[$i] =~ /^Time =/) {
			@data    = split(/,/, $lines[$i]);
			chomp(@data);
			@temp    = split(/=/, $data[1]);
			$x[$n] = $temp[1];
			$x[$n] =~ s/ //g;
			@temp    = split(/=/, $data[2]);
			$y[$n] = $temp[1];
			$y[$n] =~ s/ //g;
			$n++;
		}
	}
	make_array("x",@x);
	make_array("y",@y);

	print OUT "plot(x,y);¥n";
}
sub draw_bg {
	%temp = @_;
	print OUT "h = linspace(".$area[0].",".$area[1].",".$cell.");¥n";
	print OUT "v = linspace(".$area[2].",".$area[3].",".$cell.");¥n";
	print OUT "plot(0,v);¥n";
	my $i;
	foreach $i (sort keys %temp){
		draw_region($temp{$i});
		draw_sharp($temp{$i},$i);
	}
}
sub draw_region {
	@temp = @{$_[0]};
	for(my $i=0;$i<@temp;$i++){
		print OUT "plot(".$temp[$i].",v);¥n"		if $i eq "0" || $i eq "1";
		print OUT "plot(h,".$temp[$i].");¥n"		if $i eq "2" || $i eq "3";
	}
}
sub draw_sharp {
	@temp = @{$_[0]};
	my $i = $_[1];
	$sharps = ($temp[1] - $temp[0]) / $sharp;
	print OUT "xx".$i." = linspace(".$temp[0].",".$temp[1].",".$sharps.");¥n";
	print OUT "yy".$i." = linspace(".$temp[2].",".$temp[3].",".$sharps.");¥n";
	print OUT "for k = 1 : ".$sharps."¥n";
	print OUT "plot([".$temp[0]." xx".$i."(k)],[yy".$i."(k) ".$temp[2].']);¥n";
	print OUT "plot([xx".$i."(k) ".$temp[1].'],['.$temp[3]." yy".$i."(k)]);¥n";
	print OUT "end¥n";
```

```
}
sub make_array {
        ($name,@temp) = @_;
        print OUT $name." = [";
        for(my $i=0;$i<@x;$i++){ print OUT "$temp[$i] "; }
        print OUT "];¥n";
}
```