

令和 1 年度 修士 論文

ジャストインタイムモデリングに基づく  
車の自動経路追従制御

2020 年 2 月

指導教員 博士（科学） 甲斐 健也 准教授

東京理科大学大学院 基礎工学研究科  
電子応用工学専攻 甲斐研究室

8118539 信宮 繭

# ジャストインタイムモデリングに基づく 車の自動経路追従制御

[甲斐研究室]

8118539 信宮 繭

## 要旨

近年、様々な企業や研究機関で自動車の自動運転技術についての研究・開発が盛んにおこなわれている。自動運転技術の例として、緊急時自動停止システムや衝突回避、自動で車庫入れをする自動駐車技術、前方の車を一定の車間距離を保ちながら追従する自動追従の技術など様々ある。これらの技術はセンサやカメラなどで周りの状況を把握したり、車体を制御する、ビッグデータを活用するなど多岐にわたった分野の知識が必要となる。近年日本では、高齢化が話題となっており、報道などでも高齢ドライバーによる交通事故や死亡事故が多くなっている。その中、車が移動手段となる郊外に住む人たちは嫌でも車に乗らなければならないという状況となっている。この状況を解決するにも自動車の自動運転化は期待されている。卒業研究では、自動車の自動走行に関する研究を行ったが、道路幅を広くしないと制御できないなどの問題があった。そこで、本研究では、あらかじめ走行したい経路を用意し、その道路の中心線を設定することで目標経路を外れることなく制御することができ、現実に近い経路追従を可能とした。データベースを活用する制御手法であり、車の軌道データを大量に蓄積することで複雑な経路にも対応できる制御が実現した。

# Autonomous Path Tracking of a Automobile Based on Just-In-Time Modeling

[Kai Laboratory]

8118539 Mayu Nobumiya

## abstract

In recent years, various companies and research institutes have been actively researching and developing autonomous driving technology. There are various examples of automatic driving technology, such as an emergency automatic stop system, collision avoidance, automatic parking technology that automatically enters a garage, and automatic tracking technology that follows a preceding vehicle while maintaining a fixed inter-vehicle distance. These technologies require knowledge in a wide variety of fields, such as grasping the surrounding situation with sensors and cameras, controlling the vehicle body, and utilizing big data. In recent years, aging has become a hot topic in Japan, and traffic accidents and fatal accidents by elderly drivers have increased in news reports. Among them, people living in the suburbs where cars are a means of transportation have to ride a car even if they do not want to. Autonomous driving of cars is expected to solve this situation. In my graduation research, I did research on autonomous driving of cars, but there was a problem that control was not possible unless the road width was wide. In this study, we prepared a route that we would like to travel in advance and set the center line of the road so that we could control the vehicle without deviating from the target route, and made it possible to follow a route that was close to reality. It is a control method that utilizes a database, and has realized control that can handle complicated routes by accumulating a large amount of vehicle trajectory data.

# 目次

第1章	序論	1
1.1	研究背景	1
1.2	研究目的	1
1.3	本論文の構成	1
第2章	準備	3
2.1	自動運転について	3
2.2	ジャストインタイムモデリングの概要	3
第3章	JITを用いた自動経路追従制御	6
3.1	問題設定	6
3.2	提案手法	8
3.2.1	データベースの構築	8
3.2.2	JITを用いた制御手法	10
3.3	シミュレーション	13
3.3.1	データベース構築の条件	13
3.3.2	シミュレーション結果	14
第4章	JITを用いた自動経路追従制御の精度向上と計算量低減	20
4.1	提案手法	20
4.1.1	データベースの構築	20
4.1.2	JITを用いた制御手法	21
4.2	本手法での改善点・メリット	22
4.3	シミュレーション	23
4.3.1	データベース構築条件	23
4.3.2	シミュレーション結果1	24
4.3.3	シミュレーション結果2	25
4.3.4	シミュレーション結果3	26
第5章	結論	32
5.1	本論文のまとめ	32
5.2	今後の展望	33
	謝辞	34
	参考文献	35

# 目 次

2.1	1 入力 1 出力の場合のジャストインタイムモデリングの例 . . . . .	5
3.1	自動追従する経路の一例 . . . . .	6
3.2	車両モデル . . . . .	7
3.3	制御入力的设计 . . . . .	8
3.4	目標座標のとり方 . . . . .	9
3.5	座標変換 . . . . .	11
3.6	ソートしたデータベース (1 出力) . . . . .	12
3.7	ソートしたデータベース (2 出力) . . . . .	13
3.8	入力データの範囲 . . . . .	14
3.9	構築したデータベースの一部 . . . . .	15
3.10	入力データの範囲 . . . . .	16
3.11	構築したデータベースの一部 . . . . .	16
3.12	目標軌道 . . . . .	17
3.13	シミュレーション結果 . . . . .	17
3.14	目標軌道と車の軌道の誤差 . . . . .	18
3.15	シミュレーション結果 . . . . .	18
3.16	目標軌道と車の軌道の誤差 . . . . .	19
4.1	抜き出した近傍データ群 . . . . .	22
4.2	入力データの範囲 . . . . .	23
4.3	構築したデータベースの一部 . . . . .	24
4.4	目標経路 (カーブ状) . . . . .	25
4.5	シミュレーション結果 1 . . . . .	26
4.6	シミュレーション結果 1 のゴール付近の様子 . . . . .	27
4.7	目標軌道と車の軌道の誤差 . . . . .	27
4.8	目標経路 (S 字カーブ) . . . . .	28
4.9	シミュレーション結果 2 . . . . .	28
4.10	シミュレーション結果 2 のゴール付近の様子 . . . . .	29
4.11	目標軌道と車の軌道の誤差 . . . . .	29
4.12	目標経路 (複雑な経路) . . . . .	30
4.13	シミュレーション結果 3 . . . . .	30
4.14	シミュレーション結果 3 のゴール付近の様子 . . . . .	31
4.15	目標軌道と車の軌道の誤差 . . . . .	31

# 第1章 序論

## 1.1 研究背景

近年，世界中の企業や研究機関で自動車に関する自動運転技術の開発が著しく進んでいる．さらには自動車業界の企業に限らず，IT系の企業なども自動運転分野に力を入れているなど，自動運転に対する関心が高まっていることがわかる．自動運転技術と一括りに言うが，様々なアプローチがある．自動で車庫に入り駐車をする自動駐車技術や，障害物を検出し，自動で回避経路をとる自動障害物回避技術，緊急時などに自動でブレーキをかけ，衝突などを回避する自動ブレーキ技術，前を走る車をセンサやカメラなどで検出し，一定の間隔を保ちながら走行する自動追従の技術などがある．これらの自動運転技術，運転支援技術によって，交通事故の低減や渋滞の解消・緩和などが見込めるとの見解がある．[1]

また，近年のコンピュータ技術・ITの発展に伴い，ビッグデータの活用が可能・容易になったという背景がある．そこで，車の自動運転技術にデータを活用した手法を開発する．

## 1.2 研究目的

制御工学の分野において，「ジャストインタイムモデリング」という手法がある．これは，制御対象となるシステムの入出力データをあらかじめデータベースに蓄積することで，システム自体の数値モデルの代わりにデータベースを用いて制御対象のシステムを制御する手法である．先行研究では自動車が単一カーブを自動走行する制御の手法を研究した．カーブのみではなく様々な道路の形状に適応するため，本研究では，様々な道路の形状に対応できる自動経路追従について，このジャストインタイムモデリングを活用させた手法を提案することを目的とする．

## 1.3 本論文の構成

本論文の構成を以下に示す．

第1章 本研究の背景や目的，本論文の構成について示す．

第2章 自動運転のレベル分けについて，ジャストインタイムモデリングの概要について示す．

第3章 ジャストインタイムモデリングを用いた自動経路追従制御について示す.

第4章 自動経路追従制御における精度向上と計算量低減について手法を示す.

第5章 結論と今後の課題について示す.

## 第2章 準備

### 2.1 自動運転について

アメリカの Society of Automotive Engineers(SAE International) が自動運転の定義を策定しており、日本の内閣府もこの定義を採用している。自動運転のレベルは0から5までの6段階に分類されている。[2]

**level 0** 車両をすべて人が操作する段階を指す。ハンドル操作、アクセル、ブレーキなどシステムが介入することなくドライバーがすべて操作。

**level 1** システムがステアリング操作、加減速のどちらかをサポートする段階。このレベルでは自動運転とは呼ばずに運転支援と言われる。

**level 2** システムがステアリング操作、加減速の両方をサポートする段階。このレベルでも、ドライバーをアシストする段階のため自動運転とは呼ばずに、運転支援と言われる。日本で発売されている車に搭載されているシステムも現段階ではレベル2までである。

**level 3** 特定の場所でシステムが操作、緊急時のみドライバーが操作する段階。高速道路上のみなど特定の場所という制限がある中で、周りの交通状況などを把握してドライバーの代わりに運転を行う。緊急時やシステムが対応できない場合はドライバーが対応する。このレベルから自動運転と言われる。

**level 4** 特定の場所でシステムがすべてを操作する段階。レベル3の技術に加え、緊急時にもシステムが対応する。

**level 5** 場所の制限なくシステムが全てを操作する段階。システムが場所の制限なしに周りの状況を把握し、運転を行う。

### 2.2 ジャストインタイムモデリングの概要

近年のコンピュータやIT技術の著しい発展に伴い、ビッグデータの活用や、高速計算の処理が可能、容易になっているという背景がある。これに伴い確立された手法がジャストインタイムモデリング(Just-In-Time Modeling)である。システムを制御するには、制御したいシステムの物理的な特性を数理モデルで記述するモデリングというステップが必要となる。従来のモデリング手法には、制御対象のシステムの性質や特性を詳しく観察す



ることで数理モデルを大域的に同定する大域モデルと、システムの動作領域を複数に分割しそれぞれの領域についての数理モデルを同定する局所モデルが存在する。大域的なモデリングは、システム全体の動作点の振る舞いを数理モデルで記述する必要があるため、制御したいシステムが複雑になるほど、精度の高いモデリングは困難となる欠点がある。また、局所的なモデリングは動作領域を分けるため線形モデルを使うことができるなど大域モデルよりメリットは多いが、分割方法の決定方法や局所モデル間での補間方法などで複雑になる。[3]

ジャストインタイムモデリングは局所モデルに分類されるが、制御したいシステムの入出力データを含むデータベースをシステムの数理モデルとしてみなすことができるため、モデルを求める必要がないというメリットがある。また、システムの入出力データがあれば制御できる手法のため、制御対象の事前知識を必要としないというメリットもある。ここで、一般的なジャストインタイムモデリングの手順を示す。

#### ジャストインタイムモデリングの一般的なアルゴリズム

##### (1) データベースの構築

制御したいシステムの入出力データを含むデータベースを構築する。実機を用いてデータ収集が可能な場合は実機でのデータをデータベースに蓄積する。

##### (2) 距離の計算

要求データに対する出力を得るために、データベース内のデータと要求データとのユークリッド距離を計算する。

##### (3) ソート、近傍データの抽出

前ステップで求めた距離を基に、距離が昇順になるようにデータセットを並べ替える。並べ替えたデータセットから上位複数個を近傍データとして抽出し、局所的な線形モデルを構成する。

##### (4) 予測出力の算出

局所線形モデルより、要求データの入力に対する予測出力を算出する。

この流れを図に示すと図 2.1 のようになる。

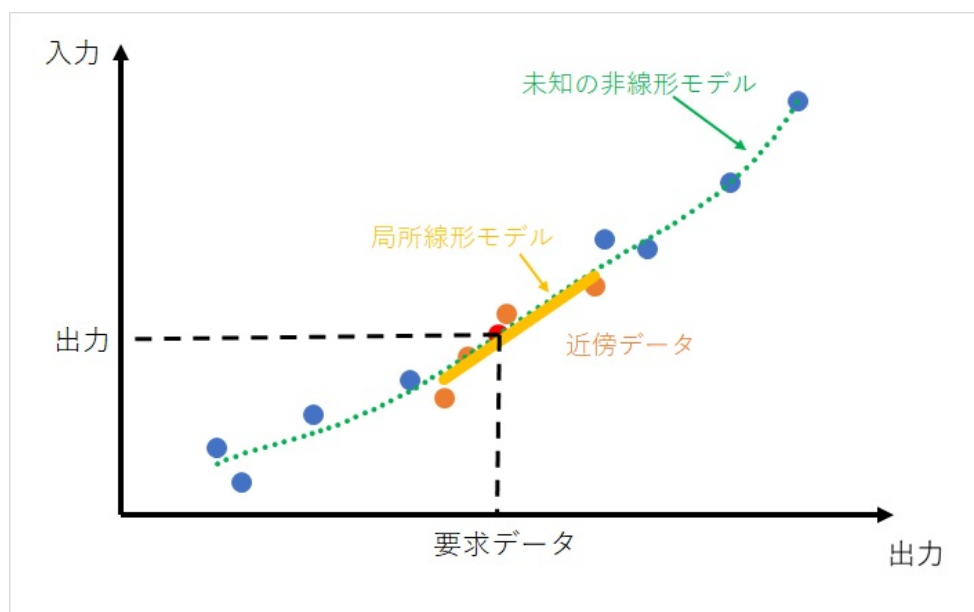


図 2.1: 1 入力 1 出力の場合のジャストインタイムモデリングの例

## 第3章 JITを用いた自動経路追従制御

本章ではジャストインタイムモデリング (JIT) を用いた車の自動経路追従制御法の提案を行う．具体的には，JIT で求める出力を 1 つの場合と 2 つの場合でそれぞれシミュレーションを行い，自動追従制御法に最適な手法を述べる．

### 3.1 問題設定

本節では，JIT を用いた車の自動経路追従制御法の問題設定を行う．例として図 3.1 に追従する経路の概略図を示す．

この図にあるように，任意の経路を等間隔に  $K$  個の細かい区間に分割する． $(X_k, Y_k)$  を

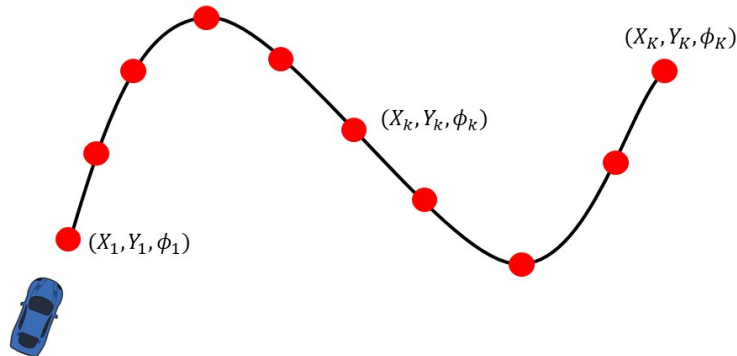


図 3.1: 自動追従する経路の一例

目標軌道上の目標座標とする．また， $\phi_k$  を目標座標における車体の目標操舵角度とする．次に，車の数理モデルを設定する． $xy$  座標系において，車体の中心座標を  $(x, y)$ ，その時の車体の操舵角度 (傾き角) を  $\theta$  という状態変数で表す．図 3.2 のように，車は進行方向に速度  $u_1$ ，操舵角速度  $u_2$  を持つ．進行方向の速度  $u_1$  は，実際の車のアクセルに相当するものと考えられ，また，操舵角速度  $u_2$  は，車のハンドルに相当するものと考えられる．この 2 つを数理モデルの制御入力とする．

これらから，車を次のモデルで表す．

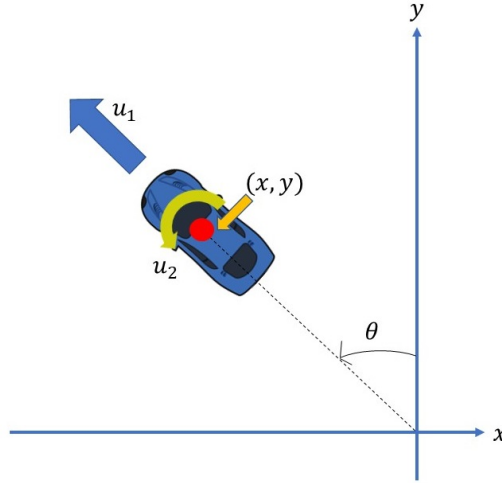


図 3.2: 車両モデル

$$\dot{x} = -\sin \theta \cdot u_1 \quad (3.1)$$

$$\dot{y} = \cos \theta \cdot u_1 \quad (3.2)$$

$$\dot{\theta} = u_2 \quad (3.3)$$

ここで，車の自動経路追従制御を行うために，問題の定式化をする．

#### 車の自動経路追従制御問題

走行したい経路，目標経路上に目標座標  $(X_k, Y_k) (k = 1, \dots, K)$  と，目標座標における車体の目標操舵角度  $\phi_k (k = 1, \dots, K)$  が設定されている．初期位置  $(x_0, y_0, \theta_0)$  から出発した車が  $t$  秒間走行した時の軌道  $(x(t), y(t), \theta(t))$  が，あらかじめ設定した誤差のパラメータ  $\epsilon, \delta > 0$  に対して，以下の2種類の条件

$$\sqrt{(x(t_k) - X_k)^2 + (y(t_k) - Y_k)^2} < \epsilon \quad (3.4)$$

$$|\theta(t_k) - \phi_k| < \delta \quad (3.5)$$

を満たすような時刻  $t_k$  が存在するような制御入力  $u_1$  と  $u_2$  を設計する．

次に，制御入力の設計を行う．まず1つ目の入力  $u_1$  は，車の進行方向の速度を表し， $u_1 = V_c$  とおく．2つ目の入力  $u_2$  は，車の操舵角速度を表している．1回のJITの制御時間を  $T$  と定義すると， $u_2 = \phi/T$  と表せる．これを先ほどの数理モデルに当てはめると次に示す通りとなる．

$$\dot{x} = -\sin \theta \cdot V_c \quad (3.6)$$

$$\dot{y} = \cos \theta \cdot V_c \quad (3.7)$$

$$\dot{\theta} = \frac{\phi}{T} \quad (3.8)$$

式 (3.8) より，時刻  $T$  において  $\theta(T) = \phi$  となることがわかる．この制御入力的设计工程を図示すると 3.3 のようになる．

目標経路の各区間で JIT を用いて適切な制御入力进行设计する．つまり，各区間において

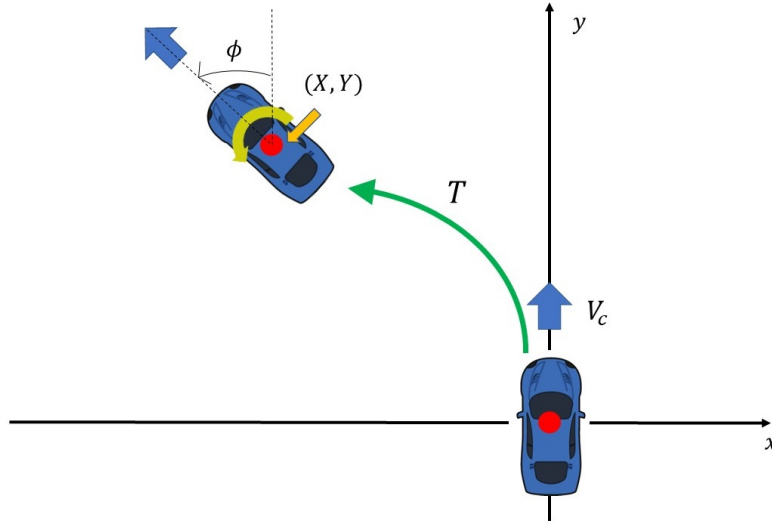


図 3.3: 制御入力の设计

JIT 制御を 1 回用いることで車が駆動するということである．この制御をスタートからゴール地点まで車が到达するまで繰り返す．

## 3.2 提案手法

本節では，データベースの構築方法と制御手法を示す．JIT で求める出力を制御時間  $T$  1 つのみの場合と， $T$  と車の速度  $V_c$  2 つの場合の 2 通りを述べる．

### 3.2.1 データベースの構築

JIT 制御に必要なとなるデータベースの構築方法を述べる．車両の初期位置を  $xy$  座標系の原点とする．

## 1 出力の場合

入力データを  $(V_c, X, Y, \phi)$ , 出力データを  $T$  とする.

図 3.4 に示す通りに, 目標座標を格子状に設定し, 車を  $T$  秒間駆動させる.

目標車体角度は現在地から次の目標座標とその 1 つ先の目標座標の 2 点の傾きから設定

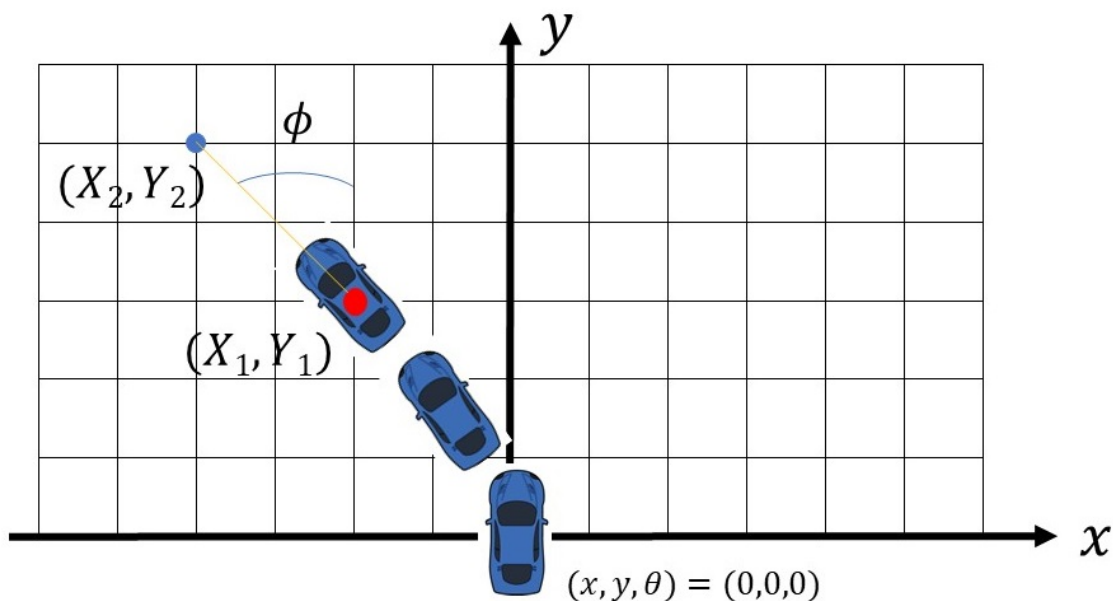


図 3.4: 目標座標のとり方

している. データ  $(V_c, \phi, T)$  を数理モデル (3.6)~(3.8) に印加し, 駆動した車の軌道と目標座標  $(X, Y, \phi)$  が 2 種類の条件

$$\sqrt{(x(T) - X)^2 + (y(T) - Y)^2} < \epsilon \quad (3.9)$$

$$|\theta(T) - \phi| < \delta \quad (3.10)$$

を満たすようなデータセット  $(V_c, X, Y, \phi, T)$  をデータベースに蓄積する. 表 3.1 に蓄積するデータベースを示す.

## 2 出力の場合

入力データを  $(X, Y, \phi)$ , 出力データを  $(T, V_c)$  とする. データベースの蓄積方法は 1 出力の場合と同じである. 表 3.2 に蓄積するデータベースを示す.

表 3.1: データベース

input				output
$V_{c1}$	$X_1$	$Y_1$	$\phi_1$	$T_1$
$V_{c2}$	$X_2$	$Y_2$	$\phi_2$	$T_2$
$V_{c3}$	$X_3$	$Y_3$	$\phi_3$	$T_3$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$V_{cN}$	$X_N$	$Y_N$	$\phi_N$	$T_N$

表 3.2: データベース

input			output	
$X_1$	$Y_1$	$\phi_1$	$T_1$	$V_{c1}$
$X_2$	$Y_2$	$\phi_2$	$T_2$	$V_{c2}$
$X_3$	$Y_3$	$\phi_3$	$T_3$	$V_{c3}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$X_N$	$Y_N$	$\phi_N$	$T_N$	$V_{cN}$

### 3.2.2 JIT を用いた制御手法

JIT を用いた自動経路追従制御法を示す．以下に制御手法の一覧を示す．

JIT を用いた制御手法の一覧

- 1, 座標変換
- 2, 要求データとデータベースの距離計算, ソート
- 3, 近傍データの抽出
- 4, 出力データの算出
- 5, 車の駆動

まず, データベースに格納したデータは, 車を  $xy$  座標の原点から出発させてデータを集めている．そこで, 目標経路の各区間において, 車の初期位置が原点からスタートするように各区間で座標変換をする必要がある．図 3.5 にあるように  $xy$  座標系から  $x'y'$  座標系に変換する．

$$\begin{bmatrix} X'_k \\ Y'_k \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \cdot \begin{bmatrix} X_k - x \\ Y_k - y \end{bmatrix} \quad (3.11)$$

車の座標  $(x, y, \theta)$  を新たな  $x'y'$  座標系において  $(0, 0, 0)$  とし, 目標経路上の目標点  $(X_k, Y_k, \phi_k)$  を  $x'y'$  座標系において  $(X'_k, Y'_k, \phi'_k)$  とする．

目標車体角度は, 座標変換した目標座標 2 点から傾きを求める．次のステップとして, 要求データとデータベースとの距離を計算する．このステップは要求データとデータベース

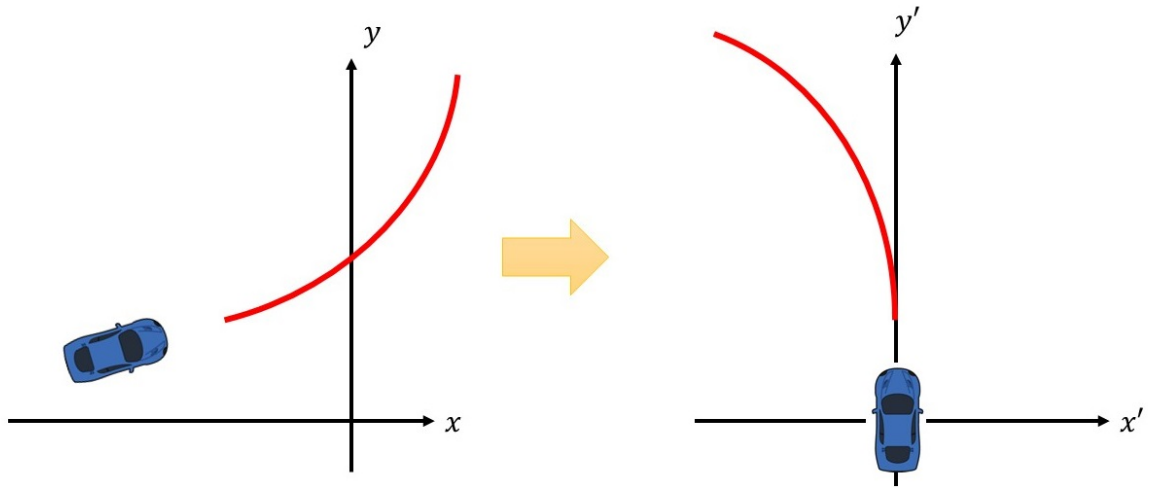


図 3.5: 座標変換

内のデータとの関連度を示すためのステップとなる．1 出力の場合と 2 出力の場合で計算式がことなるため，分けて述べる．

### 1 出力の場合

要求データ  $(V_c^*, X^*, Y^*, \phi^*)$  とデータベース内のデータ  $(V_{ci}, X_i, Y_i, \phi_i) (i = 1, 2, \dots, N)$  との距離を式 (3.12) で計算する．

$$d_i = \sqrt{(V_{ci} - V_c^*)^2 + (X_i - X^*)^2 + (Y_i - Y^*)^2 + (\phi_i - \phi^*)^2} \quad (3.12)$$

計算結果をデータベースに追加し，距離について昇順になるようにデータベース内のデータを並べ替える．


ソートしたデータベースから上位  $M$  個のデータを近傍データとして抽出する．この近傍データを用いて要求データに対する予測出力を算出する．出力は距離の逆数に関する重み付き平均を用いて求める．

$$T^* = \frac{\sum_{i=1}^M \frac{\tilde{T}_i}{d_i}}{\sum_{i=1}^M \frac{1}{d_i}} \quad (3.13)$$

算出した出力データを車のモデルに印加して車を駆動する．



input				output	
$V_{c1}$	$X_1$	$Y_1$	$\phi_1$	$T_1$	$d_1$
$V_{c2}$	$X_2$	$Y_2$	$\phi_2$	$T_2$	$d_2$
$V_{c3}$	$X_3$	$Y_3$	$\phi_3$	$T_3$	$d_3$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$V_{cN}$	$X_N$	$Y_N$	$\phi_N$	$T_N$	$d_N$

Input				output	
$\tilde{V}_{c1}$	$\tilde{X}_1$	$\tilde{Y}_1$	$\tilde{\phi}_1$	$\tilde{T}_1$	$\tilde{d}_1$
$\tilde{V}_{c2}$	$\tilde{X}_2$	$\tilde{Y}_2$	$\tilde{\phi}_2$	$\tilde{T}_2$	$\tilde{d}_2$
$\tilde{V}_{c3}$	$\tilde{X}_3$	$\tilde{Y}_3$	$\tilde{\phi}_3$	$\tilde{T}_3$	$\tilde{d}_3$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\tilde{V}_{cN}$	$\tilde{X}_N$	$\tilde{Y}_N$	$\tilde{\phi}_N$	$\tilde{T}_N$	$\tilde{d}_N$

図 3.6: ソートしたデータベース (1 出力)

## 2 出力の場合

要求データ  $(X^*, Y^*, \phi^*)$  とデータベース内のデータ  $(X_i, Y_i, \phi_i) (i = 1, 2, \dots, N)$  との距離を式 (3.14) で計算する.

$$d_i = \sqrt{(X_i - X^*)^2 + (Y_i - Y^*)^2 + (\phi_i - \phi^*)^2} \quad (3.14)$$

計算結果をデータベースに追加し, 距離について昇順になるようにデータベース内のデータを並べ替える.


ソートしたデータベースから上位  $M$  個のデータを近傍データとして抽出する. この近傍データを用いて要求データに対する予測出力を算出する. 出力は距離の逆数に関する重み付き平均を用いて求める.

$$T^* = \frac{\sum_{i=1}^M \frac{\tilde{T}_i}{\tilde{d}_i}}{\sum_{i=1}^M \frac{1}{\tilde{d}_i}} \quad (3.15)$$

$$V_c^* = \frac{\sum_{i=1}^M \frac{\tilde{V}_{ci}}{\tilde{d}_i}}{\sum_{i=1}^M \frac{1}{\tilde{d}_i}} \quad (3.16)$$

算出した出力データを車のモデルに印加して車を駆動する.

input			output		
$X_1$	$Y_1$	$\phi_1$	$T_1$	$V_{c1}$	$d_1$
$X_2$	$Y_2$	$\phi_2$	$T_2$	$V_{c2}$	$d_2$
$X_3$	$Y_3$	$\phi_3$	$T_3$	$V_{c3}$	$d_3$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$X_N$	$Y_N$	$\phi_N$	$T_N$	$V_{cN}$	$d_N$

input			output		
$\tilde{X}_1$	$\tilde{Y}_1$	$\tilde{\phi}_1$	$\tilde{T}_1$	$\tilde{V}_{c1}$	$\tilde{d}_1$
$\tilde{X}_2$	$\tilde{Y}_2$	$\tilde{\phi}_2$	$\tilde{T}_2$	$\tilde{V}_{c2}$	$\tilde{d}_2$
$\tilde{X}_3$	$\tilde{Y}_3$	$\tilde{\phi}_3$	$\tilde{T}_3$	$\tilde{V}_{c3}$	$\tilde{d}_3$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\tilde{X}_N$	$\tilde{Y}_N$	$\tilde{\phi}_N$	$\tilde{T}_N$	$\tilde{V}_{cN}$	$\tilde{d}_N$

図 3.7: ソートしたデータベース (2 出力)

### 3.3 シミュレーション

#### 3.3.1 データベース構築の条件

##### 1 出力の場合

まず, 1 出力の場合のデータベースを構築するための条件を示す. 各パラメータ ( $V_c, X, Y, \phi, T$ ) を変化させ, 様々なパターンで車を駆動させる. 各パラメータのデータ範囲を図 3.8 に示す.

各パラメータの組み合わせで駆動した車の軌道が 2 つの条件式 (3.9), (3.10) を満たすデータセットをデータベースに格納する. 本シミュレーションでは,  $\epsilon = 0.5, \delta = \pi/90$  とした. 条件をクリアしたデータセットで構築したデータベースの一部を図 3.9 に示す.

##### 2 出力の場合

2 出力の場合のデータベース構築の条件を示す. 各パラメータ ( $V_c, X, Y, \phi, T, V_c$ ) を変化させ, 様々なパターンで車を駆動させる. 各パラメータのデータ範囲を図 3.10 に示す.

各パラメータの組み合わせで駆動した車の軌道が 2 つの条件式 (3.9), (3.10) を満たすデータセットをデータベースに格納する. 本シミュレーションでは,  $\epsilon = 0.1, \delta = \pi/90$  とした. 条件をクリアしたデータセットで構築したデータベースの一部を図 3.11 に示す.

	input				output
	$V_c[\text{m/s}]$	$X[\text{m}]$	$Y[\text{m}]$	$\phi[\text{rad}]$	$T_c[\text{s}]$
幅	10~15	-5~5	0~5	$-\frac{\pi}{2} \sim \frac{\pi}{2}$	0.06~2
刻み幅	1	0.1	0.1	$\frac{\pi}{10}$	0.02

図 3.8: 入力データの範囲

### 3.3.2 シミュレーション結果

ここからシミュレーション結果を示す．目標軌道を図 3.12 に示す．カーブ状の軌道であり，目標座標は 24 点に設定した．

#### 1 出力の場合

初期値の設定を表 3.3 に示す．  
この条件でのシミュレーション結果を図 3.13 に示す．

表 3.3: 設定した初期値

$V_c$	$x$	$y$	$\theta$
10.5	2	0	$-\frac{\pi}{2}$

このシミュレーションにおける，目標軌道と車の軌道の誤差を表すグラフを図 3.14 に示す．この結果より，誤差が最大で 0.72[m] と広がっていることがわかる．

#### 2 出力の場合

初期値の設定を表 3.4 に示す．  
この条件でのシミュレーション結果を図 3.15 に示す．また，このシミュレーションにお

	$v_c[\text{m/s}]$	$x[\text{m}]$	$y[\text{m}]$	$\varphi[\text{rad}]$	$T_c[\text{s}]$
1	10	-5	4.5	1.5708	0.76
2	10	-5	4.6	1.5708	0.76
3	10	-5	4.7	1.5708	0.76
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
24863	15	5	4.8	-1.5708	0.5
24864	15	5	4.9	-1.5708	0.5
24865	15	5	5	-1.5708	0.5

図 3.9: 構築したデータベースの一部

表 3.4: 設定した初期値

$x$	$y$	$\theta$
2.5	0	$-\frac{\pi}{2}$

ける，目標軌道と車の軌道の誤差を表すグラフを図3.16に示す．これら2つのシミュレーションより，シミュレーション精度の課題が挙げられる．また，計算量の大きさも課題として挙げられる．次の章でこの改善策を述べる．

	input			output	
	$X[m]$	$Y[m]$	$\phi[rad]$	$T[s]$	$Vc[m/s]$
幅	-0.5~0.5	0.2~5	$-\frac{\pi}{18} \sim \frac{\pi}{18}$	0.04~2	1~22
刻み幅	0.05	0.05	$\frac{\pi}{90}$	0.02	0.5

図 3.10: 入力データの範囲

	$X[m]$	$Y[m]$	$\phi[rad]$	$T[s]$	$Vc[m/s]$
1	-0.5	4.6	0.17453	0.22	21
2	-0.5	4.65	0.17453	0.24	19.5
3	-0.5	4.7	0.17453	0.22	21.5
⋮	⋮	⋮	⋮	⋮	⋮
4390	0.5	4.9	-0.17453	0.24	20.5
4391	0.5	4.95	-0.17453	0.24	20.5
4392	0.5	5	-0.17453	0.24	21

図 3.11: 構築したデータベースの一部

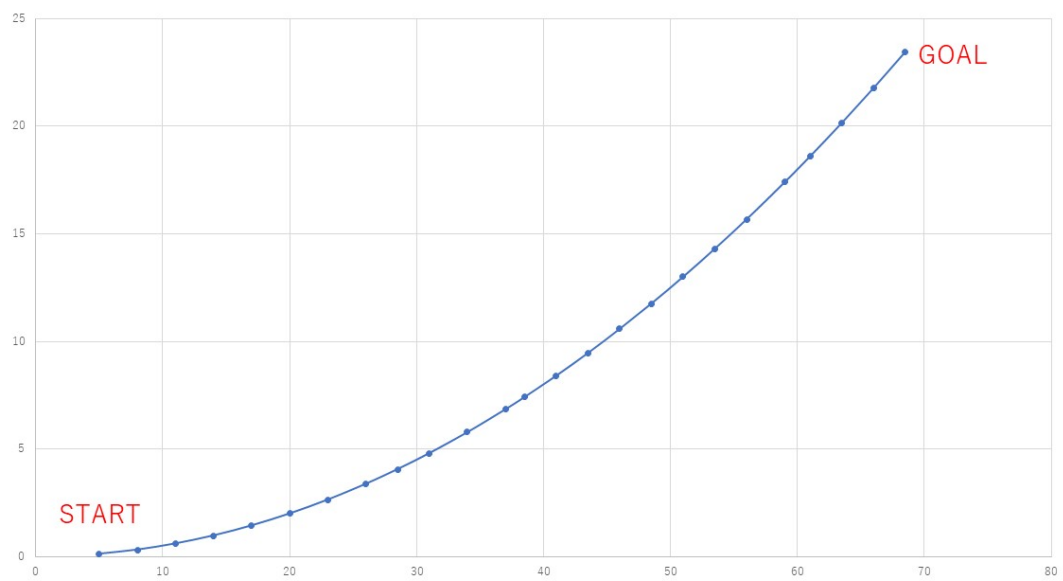


図 3.12: 目標軌道

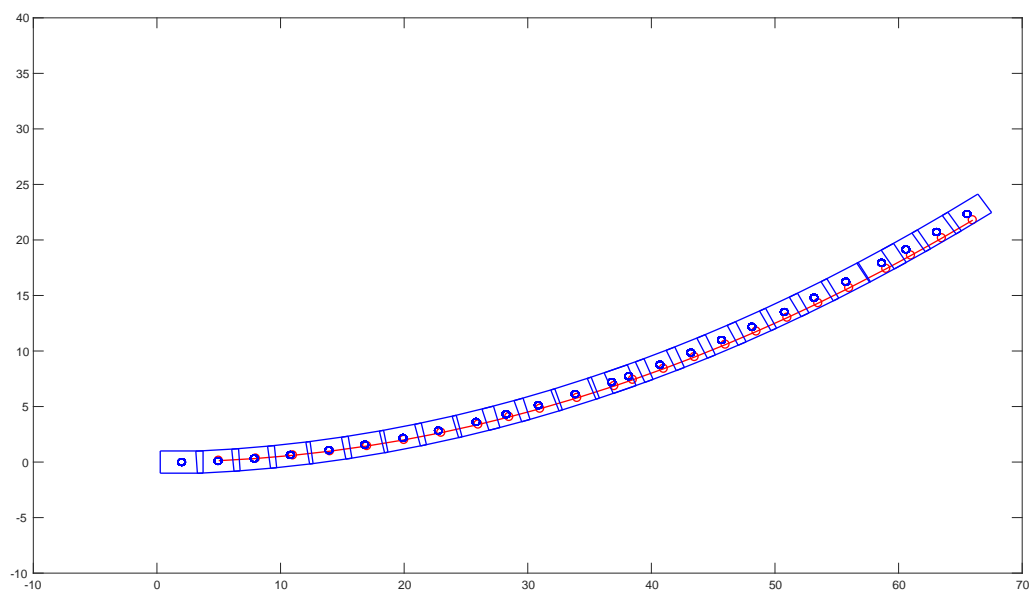


図 3.13: シミュレーション結果

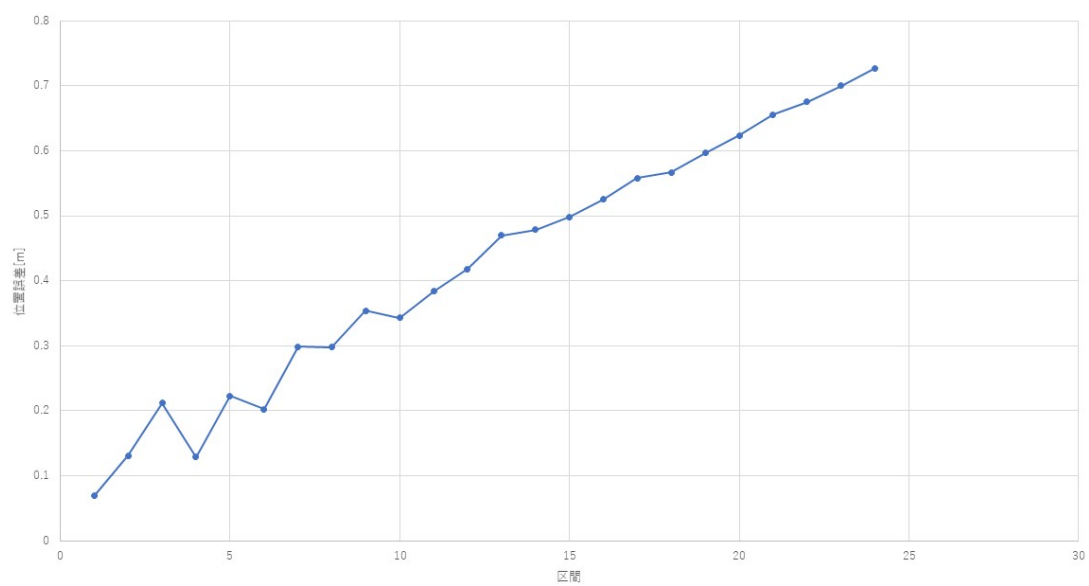


図 3.14: 目標軌道と車の軌道の誤差

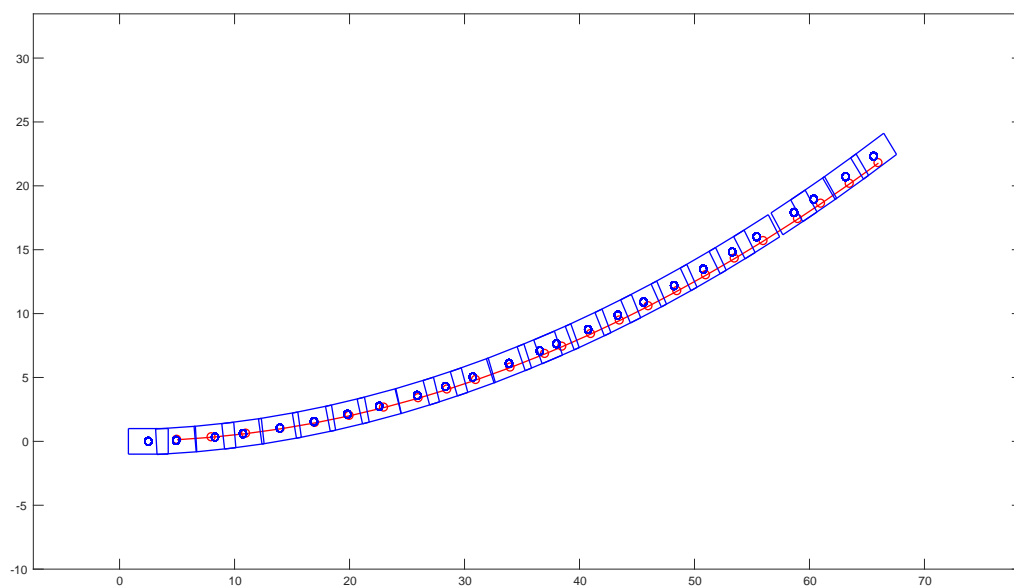


図 3.15: シミュレーション結果

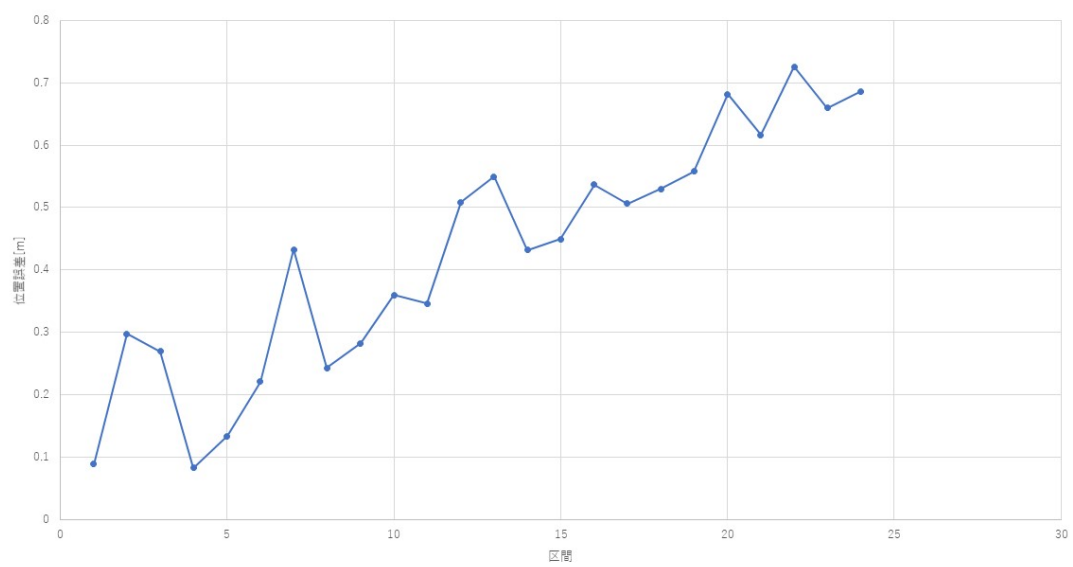


図 3.16: 目標軌道と車の軌道の誤差



## 第4章 JITを用いた自動経路追従制御の精度向上と計算量低減

本章では，第3章で述べた自動経路追従制御法の従来手法において課題として挙げられた，シミュレーション精度の向上と計算量低減を目指した改善手法を述べる．

### 4.1 提案手法

本節では第3章で提案した手法を改善することで，シミュレーション精度の向上と，計算量低減を実現する．具体的には，データベースの蓄積方法を変えることでデータベースのデータ量を増やし，データベースの精度を向上させることと，近傍データの抽出方法を変えることで計算量の低減とシミュレーション精度の向上を目指す．

#### 4.1.1 データベースの構築

データベースの構築方法について述べる．第3章では，データベースを蓄積する際，目標座標  $(X, Y)$  を設定し，車の軌道との距離誤差を計算し，条件に当てはまった場合だけデータベースにデータを格納していた．本章では，この蓄積方法だと計算量が大きくなっており，また蓄積したデータに偏りがあると考え，新しい蓄積方法を述べる．変化させるパラメータを  $(\phi, T, V_c)$  のみとする．変数  $(\phi, T, V_c)$  を変化させ，車の制御入力  $u_1, u_2$  を構成する．この制御入力を車の数理モデル (3.6)~(3.8) に印加させ，車を駆動させる．時刻  $t = T$  の時に車が到達した点をデータ  $(X, Y)$  とする．これを図示すると図 3.3 となる．この時のデータセット  $(X, Y, \phi, T, V_c)$  をデータベースに格納する．表 4.1 に蓄積するデータベースを示す．

表 4.1: データベース

input			output	
$X_1$	$Y_1$	$\phi_1$	$T_1$	$V_{c1}$
$X_2$	$Y_2$	$\phi_2$	$T_2$	$V_{c2}$
$X_3$	$Y_3$	$\phi_3$	$T_3$	$V_{c3}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$X_N$	$Y_N$	$\phi_N$	$T_N$	$V_{cN}$

#### 4.1.2 JIT を用いた制御手法

第3章の手法(従来手法)より改善した制御手法を示していく．制御手法を以下に一覧にして示す．

改善した制御手法の一覧

- 1, 座標変換
- 2, 近傍データ群の抽出
- 3, 要求データと近傍データ群の距離計算
- 4, 出力データの算出
- 5, 車の駆動

従来手法と同じく座標変換をしたあと，要求データに対する近傍データ群を検索し，抽出する．従来手法では，1組の要求データに対して近傍データを得るために，データベース内の全てのデータとのユークリッド距離を計算し，その結果の昇順にデータベースを並べ替える必要があった．このステップにおいて計算量がとても大きくなっており，それに伴い計算に時間がかかっていた．そこで，あらかじめ目標座標(要求データ)に近いデータを抜き出し，近傍データ群を構成することで従来手法の課題を改善できた．近傍データ群の抽出方法について示す．

要求データ  $(X^*, Y^*, phi^*)$  に対して，下記の不等式を満たすデータをデータベースから抜き出し，近傍データ群を構成する．

$$X^* - \alpha \leq X \leq X^* + \alpha \quad (4.1)$$

$$Y^* - \beta \leq Y \leq Y^* + \beta \quad (4.2)$$

$\alpha, \beta$  は，検索の範囲を決めるパラメータである．図4.1にデータベースから抜き出した近傍データ群を示す．

構成した近傍データ群と要求データとの距離を以下の式で計算し，距離のデータを近傍データ群に追加する．

$$\tilde{d}_i = \sqrt{(\tilde{X}_i - X^*)^2 + (\tilde{Y}_i - Y^*)^2 + (\tilde{\phi}_i - \phi^*)^2} \quad (4.3)$$

近傍データ群を用いて，要求データ  $(X^*, Y^*, phi^*)$  に対する予測出力データ  $(T^*, V^*_c)$  を算出する．以下の距離の逆数に関する重み付き平均を用いて予測出力データを計算する．

$$T^* = \frac{\sum_{i=1}^M \frac{\tilde{T}_i}{\tilde{d}_i}}{\sum_{i=1}^M \frac{1}{\tilde{d}_i}} \quad (4.4)$$

$$V^*_c = \frac{\sum_{i=1}^M \frac{\tilde{V}_{ci}}{\tilde{d}_i}}{\sum_{i=1}^M \frac{1}{\tilde{d}_i}} \quad (4.5)$$

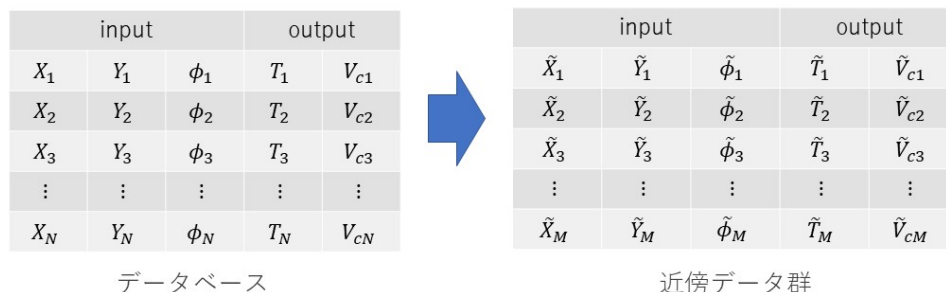


図 4.1: 抜き出した近傍データ群

算出した出力データを車のモデルに印加して車を駆動する。

## 4.2 本手法での改善点・メリット

従来手法と本手法での改善点とメリットについてまとめる。まず、データベース構築の方法について述べる。従来手法では、目標座標と車の軌道の誤差に関する条件と、目標角度と車の傾き角の誤差に関する条件をデータベース構築の条件としていたため、この条件を満たさないデータが複数存在していた。そのため、条件を満たさない無駄なデータが存在し、データの分布に空白部分が生じ、データベースの精度が低くなる原因となっていた。そこで、データベースを構築するための2種類の条件をなくし、車の軌道を目標座標としてデータベースに蓄積した。このことで、条件式を計算する時間が削減できたとともに、条件を満たさない無駄なデータというのが存在しなくなり、データの分布に偏りがなくなりまんべんなく分布することでデータベース自体の精度を上げることができた。

次に近傍データの抽出方法の変更について述べる。従来手法では近傍データを得るために要求データとデータベース内の全てのデータでユークリッド距離を計算し、データベースを並べ替えていた。この手法では、データベースの精度を良くする、つまりデータベースのデータ数を増やせば増やすほど計算量が大きくなってしまいう問題があった。そこで、要求データに近いデータをあらかじめデータベース内から検索し、近傍データ群を構成することでデータベース内の全てのデータとユークリッド距離を計算する必要がなくなり、計算量を低減させることができた。また、データベースのデータ数を増やしても計算量が

大きくならないというメリットがある．さらに，近傍データ群を構成しているため，データの並べ替えの手順が不必要となり，計算時間の低減にもつながっている．

## 4.3 シミュレーション

本節ではシミュレーション結果を示していく．

### 4.3.1 データベース構築条件

まず，データベース構築の条件を示す．各パラメータ ( $\phi, T, V_c$ ) を変化させ，制御入力を構成し車のモデルに印加する．様々なパターンで車を駆動させ，時刻  $t = T$  の時の車の座標 ( $X, Y$ ) とし，データ ( $X, Y, \phi, T, V_c$ ) をデータベースに格納する．各入力データの範囲を図 4.2 に示す．

これらから構築したデータベースを図 4.3 に示す．

	$\phi[\text{rad}]$	$T[\text{s}]$	$V_c[\text{m/s}]$
幅	$-\frac{\pi}{2} \sim \frac{\pi}{2}$	0.04~2	11~18
刻み幅	$\frac{\pi}{90}$	0.02	0.5

図 4.2: 入力データの範囲

データベースの精度を良くしても計算量はあまり大きくならないので，今回のデータベースは約 13 万点のデータを集めた．

	$X$	$Y$	$\phi$	$T$	$V_c$
1	0.28011	0.28011	-1.5708	0.04	11
2	0.29285	0.29285	-1.5708	0.04	11.5
3	0.30558	0.30558	-1.5708	0.04	12
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
135133	-21.645	21.645	1.5708	2	17
135134	-22.282	22.282	1.5708	2	17.5
135135	-22.918	22.918	1.5708	2	18

図 4.3: 構築したデータベースの一部

### 4.3.2 シミュレーション結果 1

1 つ目のシミュレーション結果を示す．まず 1 つ目の目標経路はカーブ状の経路である．目標経路を図 4.4 に示す．

目標座標は従来手法よりも各区間よりも細かくとり，98 点を設定した．各区間の間隔を細かくすることで追従制御の精度を上げた．初期値の設定を表 4.2 に示す．

この設定でシミュレーションした結果を図 4.5 に示す．

表 4.2: 設定した初期値

$x$	$y$	$\theta$
0	0	$-\frac{\pi}{2}$

ゴール付近の様子を拡大したものを図 4.6 に示す．

目標経路と車の軌道の誤差を表すグラフを図 4.7 に示す．

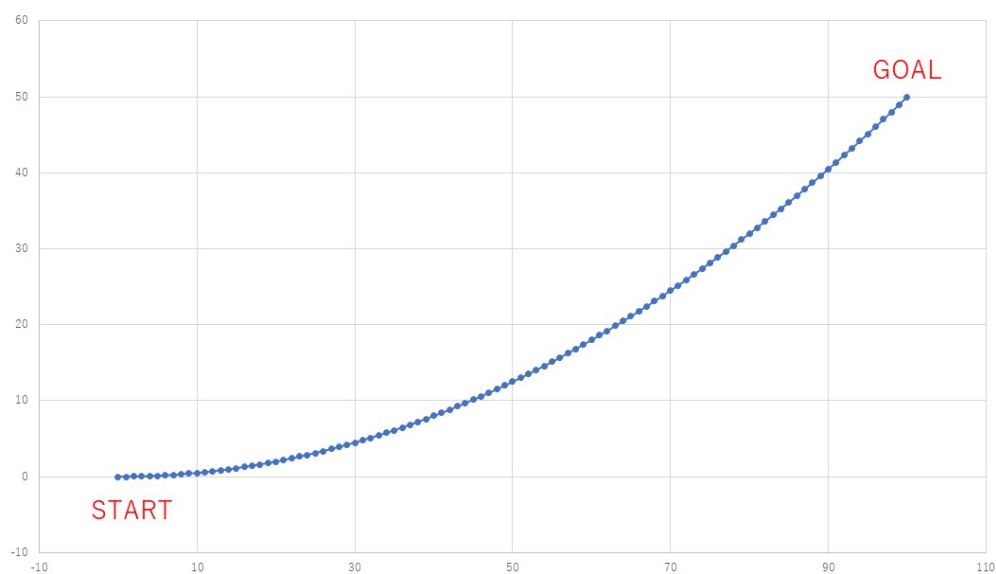


図 4.4: 目標経路 (カーブ状)

### 4.3.3 シミュレーション結果 2

2つ目のシミュレーション結果を示す．2つ目はS字を模した経路である．目標経路を図 4.8 に示す．

目標座標は 571 点設定した．初期値の設定を表 4.3 に示す．

この設定でシミュレーションした結果を図 4.9 に示す．

表 4.3: 設定した初期値

$x$	$y$	$\theta$
0	0	$-\frac{7\pi}{18}$

ゴール付近の様子を拡大したものを図 4.10 に示す．

目標経路と車の軌道の誤差を表すグラフを図 4.11 に示す．

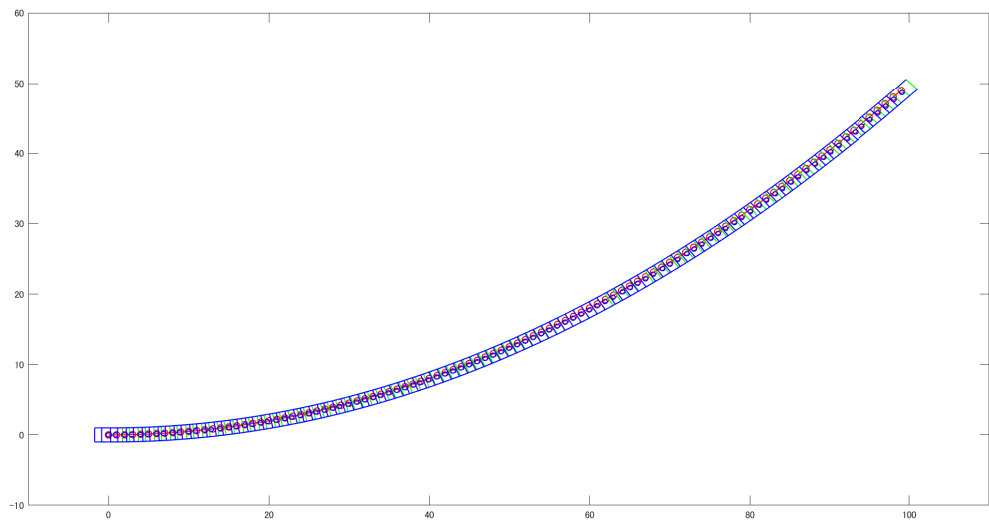


図 4.5: シミュレーション結果 1

#### 4.3.4 シミュレーション結果 3

3つ目のシミュレーション結果を示す．3つ目は少し複雑な経路である．左右にハンドルを数回切る経路にしている．目標経路を図 4.12 に示す．

目標座標は 272 点設定した．初期値の設定を表 4.4 に示す．

この設定でシミュレーションした結果を図 4.13 に示す．

表 4.4: 設定した初期値

$x$	$y$	$\theta$
-15	-170	0

ゴール付近の様子を拡大したものを図 4.14 に示す．

目標経路と車の軌道の誤差を表すグラフを図 4.15 に示す．

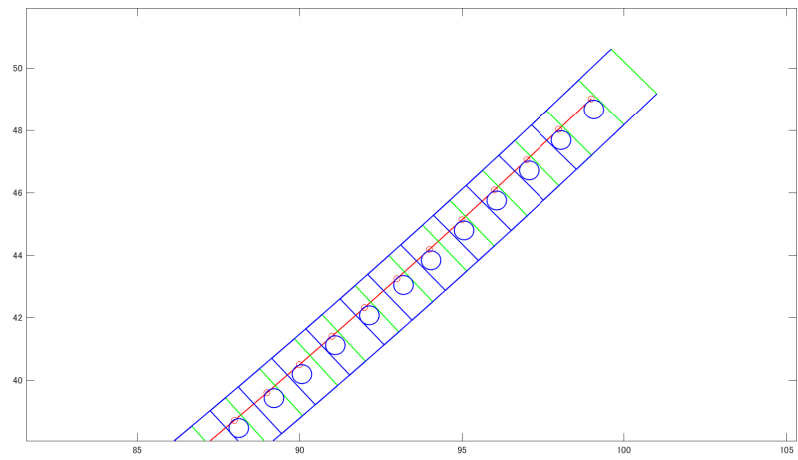


図 4.6: シミュレーション結果1のゴール付近の様子

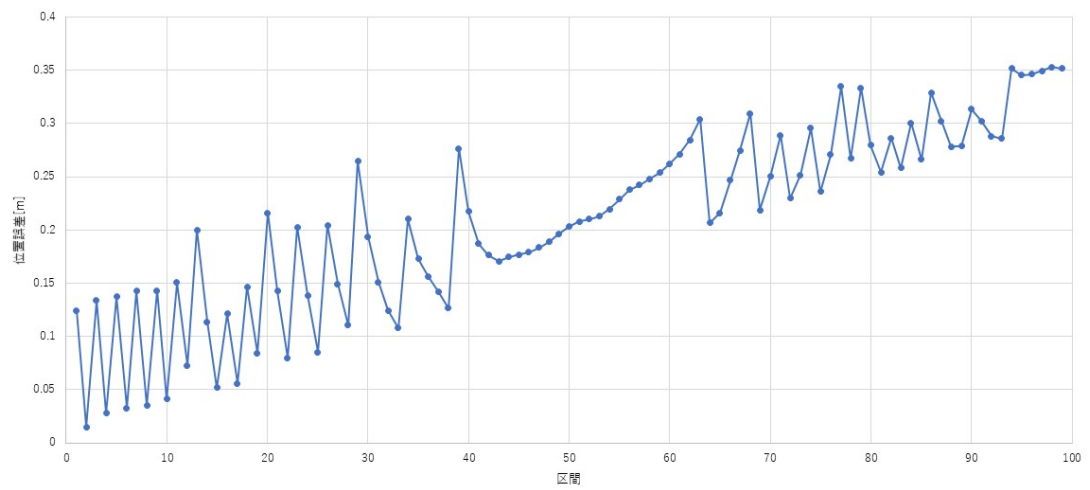


図 4.7: 目標軌道と車の軌道の誤差



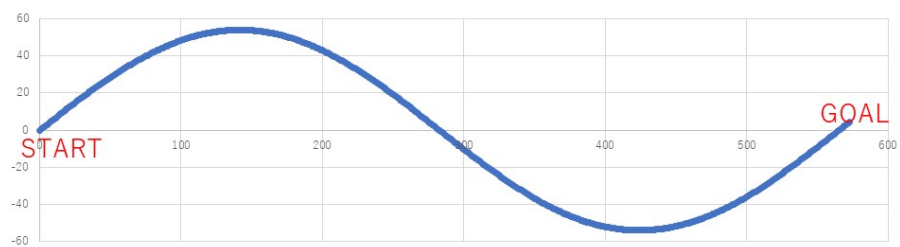


図 4.8: 目標経路 (S 字カーブ)

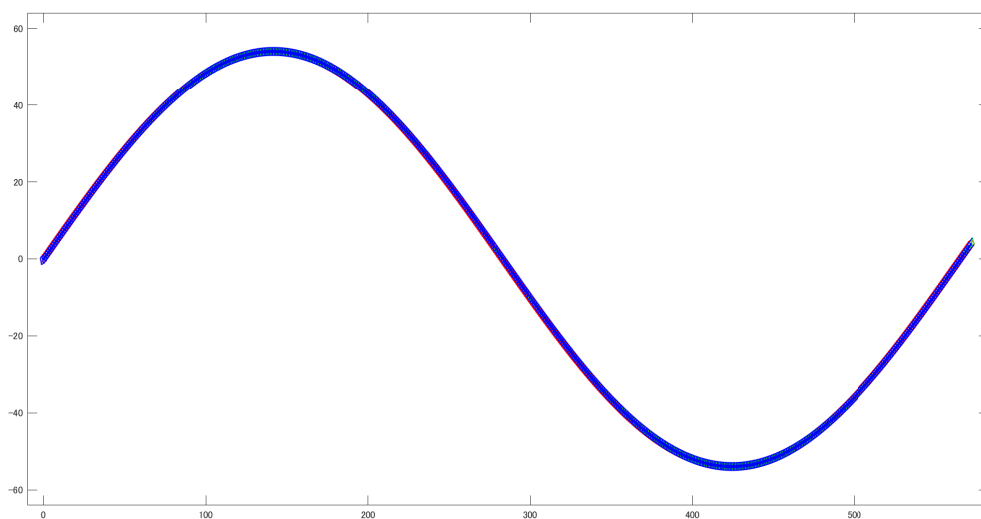


図 4.9: シミュレーション結果 2

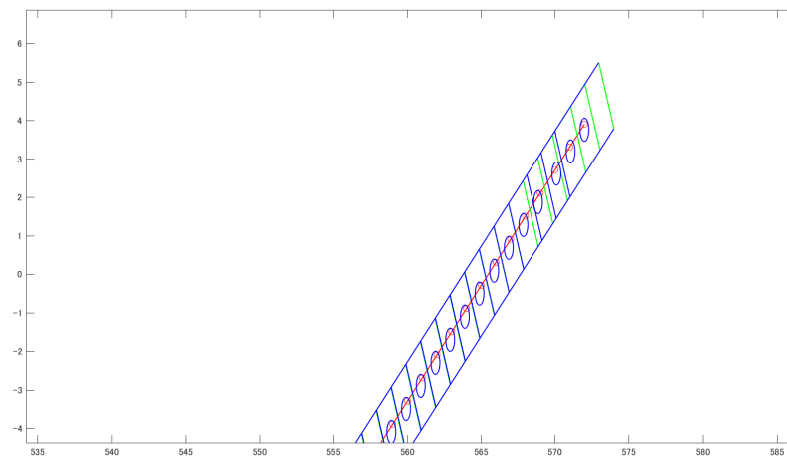


図 4.10: シミュレーション結果2のゴール付近の様子

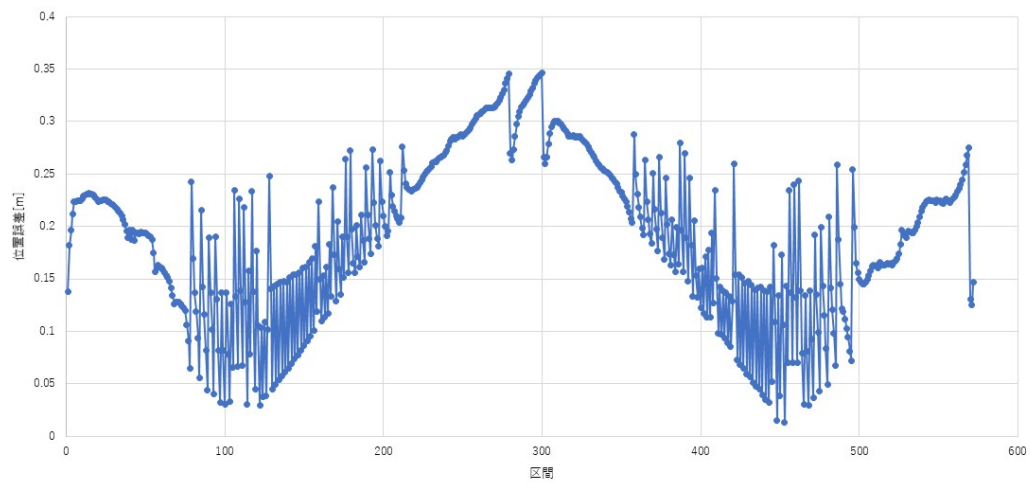


図 4.11: 目標軌道と車の軌道の誤差

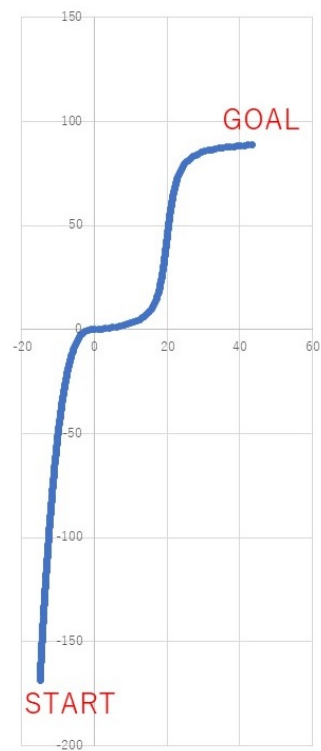


図 4.12: 目標経路 (複雑な経路)

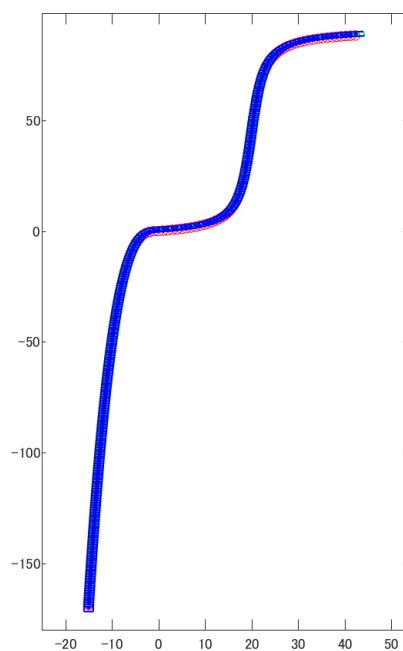


図 4.13: シミュレーション結果 3

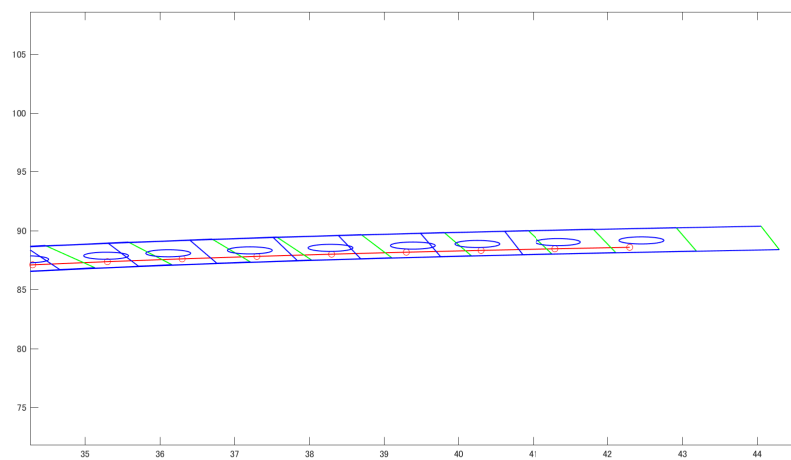


図 4.14: シミュレーション結果3のゴール付近の様子

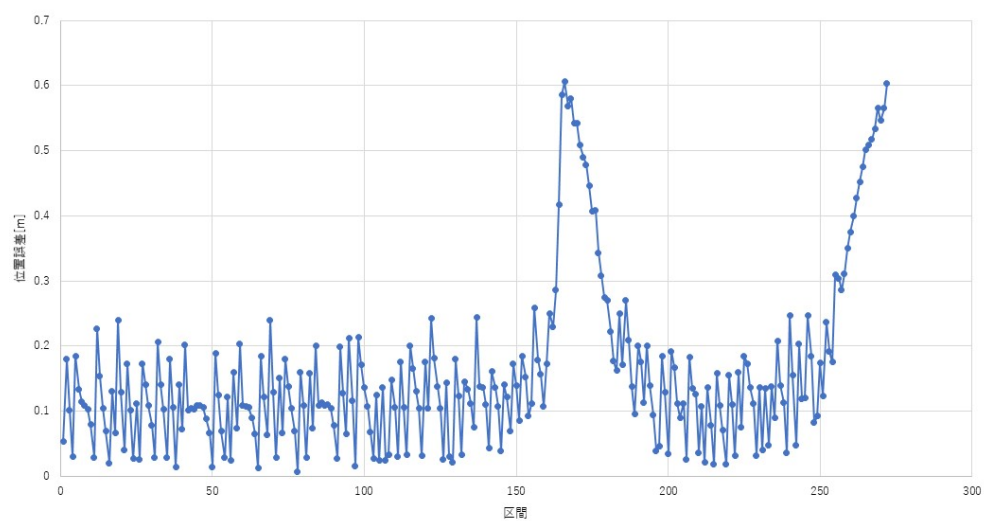


図 4.15: 目標軌道と車の軌道の誤差

## 第5章 結論

### 5.1 本論文のまとめ

本論文では、データを用いた制御手法であるジャストインタイムモデリングを用いて、車の自動経路追従制御法の提案を行った。第3章で示した制御手法で自動経路追従制御法の有効性を確認したが、制御精度の問題と計算量の問題があり、実機実験をする段階までは応用できないという問題があった。そこで、第4章では計算量低減と精度向上に向けた改善手法を提案した。要求データに関連性が高い、近傍データをあらかじめ抜き出し、近傍データ群を構成することでデータベースの精度を向上させても計算量をあまり大きくせず制御が可能となり、制御の精度の向上につながった。さらに、データベースを並び替える必要もなくなったので、計算時間の低減にもつながっている。また、データベースの構築方法も改善することで、構築時間の大幅な短縮につながったとともに、データの分布に穴がなくなり、データをまんべんなく分布させることができ、データベースの精度を上げることでシミュレーションの精度も向上させることができた。

第3章では制御時間を出力とする場合と、制御時間と車の速度を出力とする2パターンを示した。出力が1つの場合は、車の速度が一定となってしまったため、実際の車の動きと似つかないことが予想できる。しかし、車の速度を出力に加えることでアクセルの増減を考慮することができるようになった。

第4章のシミュレーション結果より、目標軌道と車の軌道の誤差を約0.3mまで小さくすることができながら自動追従制御が可能となったことがわかる。さらに、単一カーブのみでなく、複雑な経路においても自動追従制御が可能となっていることを示した。

## 5.2 今後の展望

今後の課題としては、さらなる精度の向上と計算量の低減、実機実験の2点が挙げられる。前者においては、現時点において、車の軌道と目標軌道の誤差が0.3m、複雑な経路においては0.6mとなっている。実際に車を自動運転させるとなると、この誤差によって事故が起きると考えられなくもない。この位置の誤差の目標を0.1mを目標として研究を行ってきたが、実現には至らなかった。そのため、今後の課題として精度を向上して、誤差をより小さくできる制御法を考えたい。

実機実験については、まずRoboCarなどで室内で行いたいと考えている。本研究は車にコンピュータを搭載している想定で制御を行っていたが、RoboCarなどではコンピュータを搭載することは不可能だと考えられる。そのため、実機とコンピュータで通信が必要となると考えられる。本論文の制御手法では通信時間などは考慮していないため、通信時間などを考慮したシミュレーション、実験を行いたいと考えている。

# 謝辞

本研究にあたり，ご多忙の中多大なるご指導を頂きました，甲斐 健也 准教授に深く感謝いたします。また，本研究を進めるにあたってご協力いただいた甲斐研究室の皆様にも深く感謝の意を申し上げます。

# 参考文献

- [1] 国土交通省, ”自動運転を巡る動き”  
(<https://www.mlit.go.jp/common/001155023.pdf>)  
2020 年 1 月 27 日参照
  
- [2] カーナリズム, ”自動運転”  
(<https://matome.response.jp/articles/1294>)  
2020 年 2 月 2 日参照
  
- [3] 牛田 俊, 木村英紀 : ”Just-In-Time モデリング技術を用いた非線形システムの同定と制御”, 計測と制御, 第 44 巻, 第 2 号, pp.102-106, 2005