論文 / 著書情報
Article / Book Information

題目(和文)	統計的モデルを用いた自動車ガソリンエンジンの制御モデル同定
Title(English)	
著者(和文)	大山博之
Author(English)	Hiroyuki Oyama
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:甲第10787号, 授与年月日:2018年3月26日, 学位の種別:課程博士, 審査員:山北 昌毅,藤田 政之,三平 満司,早川 朋久,畑中 健志
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第10787号, Conferred date:2018/3/26, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Tokyo Institute of Technology

平成 29 年度 博士論文

統計的モデルを用いた自動車ガソリンエンジンの制御モデル同定

Control Model Identification of Automobile Gasoline Engine Using Statistical Model

2018年2月9日

東京工業大学大学院
理工学研究科　機械制御システム専攻
大山 博之
指導教員 山北 昌毅 准教授

Abstract

To solve the energy issue and the global environmental problem, automakers have developed high functional engines. Although engine control systems have been complicated, the development period is becoming shorter to meet market needs. It is difficult to develop products satisfying requirements in a short term through conventional development environments. For such a background a model based development (MBD), which is a development approach using virtual models and simulators, attracts much attentions as one solution to the problem.

To reduce CO_2 emission, a control region for automotive engine system is pushed toward a boundary between normal and abnormal engine operations such as knocking and misfiring. The abnormal operation may damage engines seriously. Therefore a control method satisfying the boundary constraints robustly is highly demanded. To design such a controller through MBD, indeed, accurate numerical models of these boundaries are required. To identify the boundaries on an engine test bench, conventional methods need a lot of time and are costly. The goal of this research is to obtain accurate boundary models with an efficient input design algorithm.

In a field of automotive engine design, we usually consider static cases. This means that we obtain experimental data after an engine reaches steady states. The approach to determine input data is called design of experiment (DoE). The most common approaches to design the control region are a convex hull-based methods (e.g. Rapid Hull Determination Algorithm) and a way that minimizes the variance of the parameters (e.g. D-optimal design). However, since the static boundary models cannot deal with transient responses, they cannot be used directly for control system designs. In order to obtain accurate control models, it is necessary to use new identification methods of the dynamical system including the boundary detection signal.

In this paper, we develop a series of methods for the efficient engine system identification using statistical model such as Gaussian process (GP). Identification of the dynamical engine model can be divided into four phases : identification of static boundary model, acquisition of data using dynamical DoE, identification of dynamical model and online learning.

In the phase of the static boundary identification problem, we propose a DoE strategy based on the GP classifier with an expectation propagation algorithm. This method can describe accurate boundary models and provide an efficient input design algorithm using the generated model. This method can describe non-convex boundary and can determine the next measurement points near the boundaries. By this direct determination of additional inputs as points, the calibration time is expected to be much shorter.

In the phase of training data acquisition, we propose a new dynamical DoE method based on GP. This method is a model-based online algorithm that sequentially designs the next input by using GP models. Since the input design is performed while estimating dynamical boundary, it is not too conservative. In this algorithm, the input sequence design problem is formulated as an optimization problem that maximizes the sum of long-term prediction variances of the output with the static boundary constraint.

Then we propose a dynamic model structure based on a nonlinear auto-Regressive with exogenous inputs model as a control model of the automotive engine systems. This model is learned using a deep learning or a GP regression. Furthermore, we clarify relationships between the number of learning data, prediction / estimation accuracy and learning time by using numerical simulations. This approach is useful for deciding how to select a model learning method with respect to the number of training data. As a result, it is shown that when the number of learning data is small, the model based on GP is effective, and in the case where the number of learning data is large, the model using deep learning is suitable.

In the phase of online learning, we propose a robust recursive Gaussian process (RGP) as an effective online learning algorithm even when observed values are contaminated by outliers. GP model is suitable for identifying engine models that require modeling with a small amount of data. On the other hand, when online learning is performed using data-based approach such as GP and deep learning, outliers adversely affect model learning. In online learning of automotive engines, it is impossible to avoid the occurrence of outliers. Therefore a robust and fast online learning method is required. As a robust Kalman filter does, robust RGP explicitly estimates outliers by using $l1$ regularization to the RGP update optimization problem.

The validities of the proposed strategies are demonstrated by using an engine benchmark problem provided by the joint research committee of the Society of Automotive Engineers and Society of Instrument and Control Engineers, where the empirical combustion profiles with engine operating conditions and the knock model are implemented in this benchmark.

目 次

図 目 次

表 目 次

第1章　序論

本論文では自動車ガソリンエンジンシステムを対象とし, 異常運転を抑制するなど安全性を満たした効率的な学習データの獲得と, 少量のデータを用いた統計的制御モデル同定手法について議論する. 具体的には, ガウス過程などを応用することで, 静的実験計画法, 動的実験計画法, 制御モデルの同定とオンライン学習という一連の制御モデル同定手法を提案する. この章では研究の背景と目的及び論文の構成についてまとめる.

1.1　研究の背景と目的

1.1.1　自動車エンジン制御開発における統計的モデルの活用

環境保全やエネルギー枯渇対策問題への関心が高まる中, 自動車エンジンの開発においても, 環境負荷低減などの社会的要求が高くなっている. そのため, 高い燃料効率を実現し, 排気ガス規制の要求を満たすエンジンの開発が課題となる. このようなエンジンの開発はその大規模化と複雑化を意味し, 実際にターボチャージャーなどの新たなデバイスが追加されたことにより, Electronic Control Unit (ECU) に求められる性能が近年指数的に増大していることからも見てとれる. 今後の運転自動化や協調運転などへの発展も考えると, 自動車システムは更に大規模で複雑なものになることは容易に予想できる. 加えて, 市場の要求に迅速に答えるために開発期間の短縮も要求されている. このような背景のもと, 制御システムを含むエンジンの効率的な開発手法として, 数値モデルや数値シミュレーションを応用する Model Based Development (MBD) [1]-[6] と呼ばれる開発手法が広く用いられるようになり, 今後も必要不可欠なものになると考えられる. また, 日本の自動車メーカーを例にとると, Japan MBD Automotive Advisory Board (JMAAB) が MBD の技術確立や人材育成を目的としたガイドラインの作成を行っており, MBD は次のように定義される (ETSS-JMAAB フォローアップ WG(2015)[7] より引用).

> MBD の定義 : 「複雑化・高度化した現代の自動車制御システム開発において MATLAB/Simulink 等の Computer Aided Engineering (CAE) ツールによって, 制御装置と制御対象の機能をモデル化し, 開発の全プロセスにおいて,

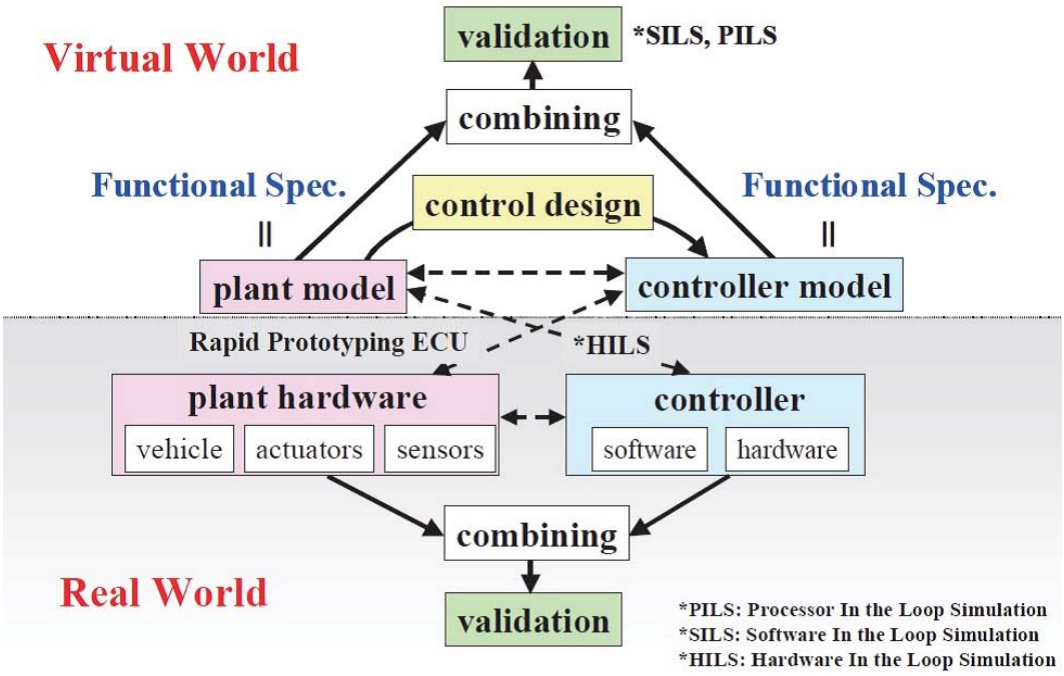


Fig 1.1: Concept of MBD : A. Ohata (2009)[1]

シミュレーション技術を活用することで, 製品ライフサイクル全般にわたった品質向上と開発効率向上を目指した開発のこと.」
モデルの定義 : 「コンピュータが理解できる対象の簡略表現.」

MBD では特に制御対象と制御装置のモデルを機能仕様書として扱うことがポイントとなる. また, MBD では制御対象と制御装置, 実機とモデルの組み合わせで開発手法が分類されており, その関係性は Fig. 1.1 で良く表される. 分類される手法の概要を以下に示す.

- MILS(Model In the Loop Simulation)/SILS (Software In the Loop Simulation)
  制御対象と制御装置の両方をモデル化し, それらを組み合わせ, 仮想環境で評価する手法. 特に実際に ECU に実装するコードを用いたものを SILS と呼ぶ.

- PILS(Processor In the Loop Simulation)
  制御装置のマイコンの挙動もモデル化し, モデル化した制御対象と及びその周辺機器と組み合わせて仮想環境で評価する手法.

- HILS(Hardware In the Loop Simulation)
  制御対象の挙動をリアルタイムの仮想環境で構築し, 制御装置としては実際の ECU を接続し評価する手法.

- Rapid Prototyping ECU
  実機の制御対象に制御装置モデルと汎用 ECU を繋いで評価する手法.

MBD では自動車制御開発プロセスの各工程に適した手法を用いることで, 工程全体の開発効率と品質の向上を図ることができる [6][7]. そして, このような MBD 開発においては特に制御対象のモデルを精度良くかつ簡易に構築することが必要不可欠となる.

正確で簡易な数理モデルの獲得は開発工程の効率だけではなく制御精度の向上にも有効である. 制御対象のモデルを用いた Model Based Control (MBC) は, 従来の制御マップと PID を組み合わせた制御手法では限界があった, むだ時間や遅れを含む対象の制御に効果的であるとされているためである. 一方で制御対象モデルに相応しい制御手法の確立と, MBC を実装するための ECU の高性能化, 制御則及び ECU の並列化は課題となっている [8][9].

MBD 及び MBC で用いられているモデルとしては物理モデル (ホワイトボックスモデル) と統計的モデル (ブラックボックスモデル) 及び両者を併用したモデル (グレーボックスモデル) の 3 つに分類できる. その中でもグレーボックスモデリングは最も現実的なモデリング手法として制御系設計の現場でも多く使用されている [3]. 一方で, エンジンの燃焼行程におけるノック現象 (爆発的な異常燃焼による急激な圧力波の発生現象) など, 物理モデルでの表現が難しい制御対象が存在する. そのような対象に対しては統計的モデルがふさわしく, モデリング手法の選択及び提案が重要となる. また, 近似物理モデルと統計的誤差モデルを組み合わせたエンジンモデリング手法 [10] も提案されており, その中でも統計的モデルの役割は大きい.

1.1.2 境界モデルの重要性と課題

自動車エンジンの制御は最適化が進んでおり, 更なる性能向上の為にはノックや失火といった異常運転を起こす領域近傍での制御が要求される [11]. 例えば, 点火時期はエンジンの性能を決定する重要な制御入力の 1 つであるが, 自動車ガソリンエンジンにおいて出力トルクに対して最適となる点火時期 (Minimum advance for the Best Torque, MBT) はノック現象が頻繁に発生する運転領域となる場合がある. これは, 高い燃費性能が期待できる次世代型ガソリンエンジンで顕著に表れ, ノック境界近傍での制御はエンジン効率化のボトルネックとなっている. エンジン運転の効率化のためにはノックの発生率が十分に小さくなる運転領域で, 点火時期を MBT にできるだけ近づける制御が求められ, その制御はノックリミット制御と呼ばれる (Fig. 1.2). ノックの検知は, ノックセンサーにより得られたノック信号の周波数成分解析によりノック強度を計算し, ノック強度がある閾値を超えた場合にノックが発生したと判断される. そして, ノック現象は同じ運転条件でも確率的に発生することが知られている [13][14]. 実際に大量の実験データから作成した, ノック強度の近似確率モデルの例を Fig.1.3 に示

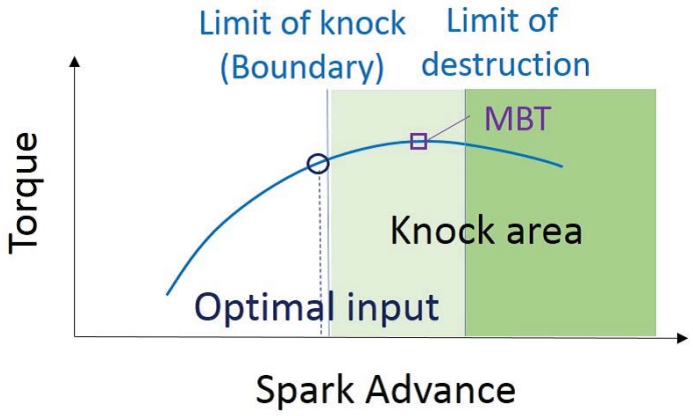
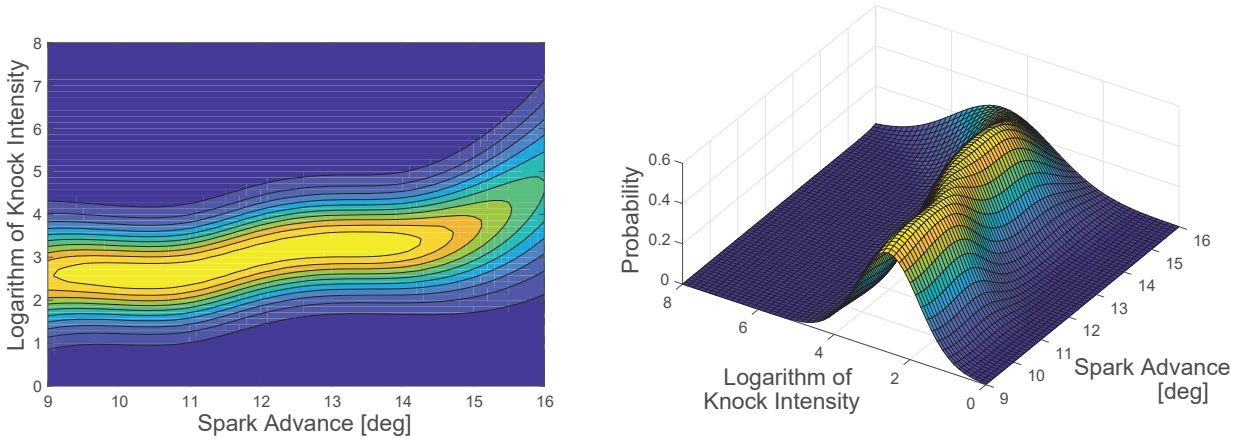


Fig 1.2: Characteristic of torque as a function of spark advance.



(a) Knock probability (2D view).

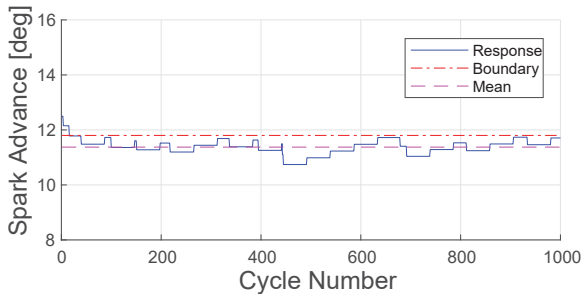
(b) Knock probability (3D view).

Fig 1.3: Numerical model of knock probability.

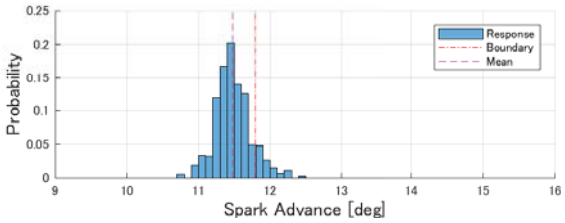
す. ノックリミット制御では統計的制御手法によりノック発生確率を目標値とするようなフィードバックを主体とした統計的制御 [15]-[19] が行われる. そのため, その入力値も確率的なものとなり, 入力の平均値はノック境界からの偏差を持ち, ノック境界を超える入力も発生してしまうことがわかる (Fig.1.4). しかしながら, この問題はノックのモデル化が可能となれば, MBC により解決される可能性がある.

ここでは点火時期制御の例を紹介したが, 実際のエンジンシステムは多入力系であり, 問題はより複雑となる. そのため, 次世代ガソリンエンジン制御において最適な MBC を実現するには, 点火時期以外の運転条件も含めた上で, ノックや失火といった異常運転境界の正確なモデル化が求められている. また, これらのモデルは統計的モデルでの表現がふさわしいと考えられる.

統計的モデリング手法の一般的な課題としては, 全ての運転条件をカバーするため



(a) Spark angle response.



(b) Histogram of spark angle response (10000 cycle).

Fig 1.4: Closed-loop spark angle response using a likelihood-based control [16].

の方法論が確立していないことと, 運転条件を全てカバーするデータを獲得するための実験数が膨大になり, モデルの獲得に必要な時間とコストが大きくなる点が挙げられる. これらの課題は動的モデルや確率モデルが必要となる場合には特に顕著となる. 例えば, Fig.1.3 の場合においては 1 つの運転条件につき約 1500 点の学習データを用いて確率モデルを作成しており, 全ての運転条件をカバーするには作成コストが大きすぎる. 加えてエンジン特有の課題として, 異常運転領域での運転はエンジンに損傷を与えてしまうため避けなければいけないことがある. 従って, エンジン運転の安全性を満たすためには異常運転の発生を抑えるような効率的な学習データの獲得と, 得られた少量のデータを用いた統計的モデルの学習が求められる.

1.1.3 研究の目的と位置付け

以上の背景を踏まえ, 本論文では自動車エンジンの効率的な統計的モデル構築を目的とし, ガウス過程 [20] などを応用した一連の制御モデル同定手法を提案する. この制御モデルは境界モデルとエンジンシステムの動的モデルを含んだものである.

ガウス過程の動的システムへの応用は入出力データから直接的に回帰モデルを得る手法 [21][22] が主流であり, その柔軟な表現力と過学習を起こしにくいという性質からエンジンシステムへの適用もシリンダ内燃焼の定常モデルを中心に行われている [23][24][25]. しかしながら, PILS/HILS 及び制御モデルで用いる際のリアルタイム性に関わる予測計算コスト及び学習コストに関しての議論や, 高効率な制御において特に重要となる動的境界モデル同定に関する議論が不足している.

そこで本論文ではガウス過程モデルを応用し, 境界モデルを考慮した上で効率的なデータ点獲得を行う手法を提案する. この手法ではエンジン運転の安全性を満たした上で, ガウス過程モデルを良く表現できる学習データを逐次的に設計していく. このような効率的な学習データ獲得手法はエンジン適合の工数削減のみならず, モデル精度の向上及びリアルタイム性の確保に貢献できる.

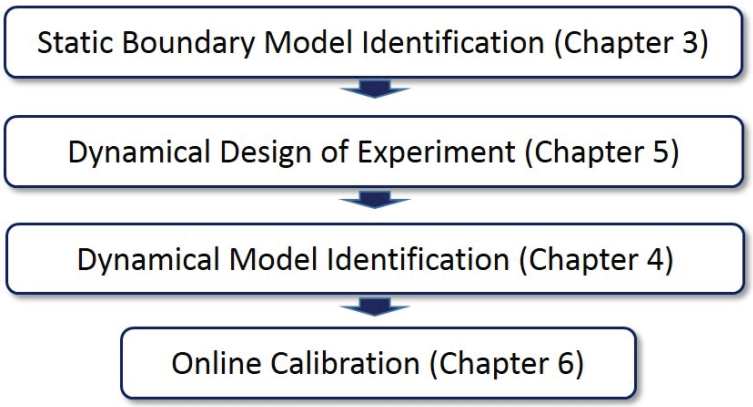


Fig 1.5: Identification of engine dynamical systems

更に, ガウス過程の近似計算手法や深層学習と組み合わせた高速化手法の適用を考え, その学習データ点数とモデル精度, 学習時間との関係性の比較を行う. 一般に学習データ点数が多くなるとモデル精度が向上するため, 適合行程において学習データ点数とモデル精度はトレードオフの関係にある. 特にガウス過程では予測に用いる学習データ点数が予測計算コストに直接関係するため, ECU で制御周期内の確実な予測計算を行う必要があることを考慮しても, 学習データ点数の制御モデルへの影響は大きい. そのため, 学習データ点数とモデル精度との関係性を示し, 要求される精度を獲得するのに必要な学習データ点数の指標を示すことは重要である. また, 統計的モデルの利点として, 新たに獲得した学習データを用いて適応的にモデルを学習・更新できる点も挙げられる. ECU の性能が制御系の複雑さに追いついていない現状では, その学習時間は小さいほど実用的であるため, モデル学習に必要な学習コストの指標を得ることも重要である. 加えて学習データに外れ値が混ざった場合にも適用できるモデル更新手法も必要不可欠となる技術の 1 つであるため, 本論文ではロバストなオンラインモデル更新則についても扱う.

提案するエンジンシステムのモデル同定手法は Fig. 1.5 に示すように大きく 4 つのステップから成る. それぞれの概要を簡単に以下で説明する. ステップ毎の具体的な手法や関連研究との詳細な比較は各章で述べる.

Step1 静的境界モデルの同定

エンジンの定常状態に注目し, 定常状態においての通常運転領域を同定する. その際に異常運転を発生させることなく, 少ないデータ点数で境界モデルの構築を行う.

Step2 学習データの獲得

静的境界モデルをもとに, 異常運転を避けつつも動的モデルの精度を向上させるような学習データを獲得する入力点を設計する.

Step3 動的モデルの同定

獲得した学習データを用いて規範となる動的モデルの学習を行う.

Step4 オンライン学習

学習した規範モデルをもとに, オンライン学習によるモデル更新を行うことで, 個体差によるばらつきや短期的に変化する運転条件に対処する.

提案するエンジンモデル同定の流れは, モデルとしてガウス過程などの応用を考える点と, 積極的に動的モデルを扱う点以外は標準的なエンジン制御系開発工程に基づいたものとなっているため, 部分的に開発行程へ適用することも可能である.

また, 提案手法の有効性は Society of Automotive Engineers (JSAE) と Society of Instrument and Control Engineers (SICE) のジョイントベンチャーから提供されている境界近傍制御ベンチマーク問題で用いられているノッキングと失火のモデルを含むエンジンシミュレーター [12] にて確認する. 提供されたエンジンシミュレータはエネルギーや運動量といった保存則や拘束に基づいた複合物理領域モデリング手法である HLMD (High Level Model Description)[3] により記述されるが, 本論文ではモデルの事前知識は用いず, ブラックボックスモデリングを行っている. この検証は制御系設計における MILS の段階に位置すると考えられ, 数値シミュレーションを用いることで少ない工数でモデル化手法の有効性検証が可能となる. 今後は制御対象であるエンジンや制御装置である ECU 及び実験装置を実機に置き換えることで, 順次実用化に近づけていく必要がある.

1.2 論文構成

本稿の構成を以下に示す.

1 章 序論

論文の全体としての研究背景と目的について，先行研究との対比を行いながら紹介を行った.

2 章 準備

2 章では本論文で扱うガウス過程及びその近似手法の概要を説明した後に, JSAE と SICE から提供されているノッキングと失火のモデルを含むエンジンベンチマーク問題について説明する.

3 章 静的境界モデルの同定と静的実験計画法

自動車エンジンの境界モデルは十分時間の経った定常状態での動作について考えられることが多い. そのモデルを効率的に獲得するための入力列設計手法は, 実験計画法と呼ばれている. 3 章では, ガウス過程を用いた新たな静的実験計画法を提案する. 具体的には, まず確率的モデルである期待値伝播法を用いたガウス過程分類器を応用することで, 静的境界モデルを作成する手法を提案する. 更に生成モデルを用いて逐次的にシステムへの入力列を設計する, 効率的な入力列設計法を提案する. この手法は運転可能領域が非凸な場合でも適用可能であり, 入力候補の値を直接与えることができるため, 適合時間の大幅な短縮が期待できる. 提案手法の有用性はエンジンシミュレータにて検証する.

4 章 動的モデルの同定

4 章ではエンジンの制御モデルとして Nonlinear Auto-Regressive with eXogenous inputs (NARX) モデルをベースとした, 観測出力の動的モデルとトルクと境界推定モデルを統合するシステムモデルを考え, そのモデルを深層学習とガウス過程及びその近似手法を用いて学習する. その中で, 学習データ数と予測・推定精度及び学習時間の関係を数値シミュレーションを用いて明確にすることで, モデル構築手法選択の指標を示す. 更に大量の学習データを持たない場合のモデル更新手法として, ガウス過程の高速近似計算手法である完全独立学習条件近似とオンライン更新アルゴリズムである逐次ガウス過程を併用することを提案している. また, 数値シミュレーションの結果を踏まえ, 深層学習の制御系応用に関して考察を行う. これらの有効性検証はエンジンシミュレータにて行う.

5 章 ガウス過程を用いた動的実験計画法

5 章では, NARX モデルでエンジンの動的制御モデルを表現した上で, 異常運転の発生を抑えた入力列設計手法について述べる. データ点数が少ない場合においてガウス過程及びその近似手法が良い予測性能を持つことが示されたが, 一方で自動車エンジンのモデル同定では安全性を担保した上での効率的な入力列設計手法が求められる. 特にガウス過程モデルの性能は学習データ点によって決定されるため, その性能は入力列設計法に従ったものとなる. 3 章との違いは境界の動的モデルを扱う点と, 境界モデルだけではなく制御モデルを扱うことにある. 具体的には, 静的境界モデルを入力制約とした上で予測ホライゾン内の予測分散の重み付き和が最大となるような入力を計算しつつ, ガウス過程モデルをオンラインで更新していく手法となっている. その有効性の検証はエンジンシミュレータにて行う.

6 章 ロバスト逐次ガウス過程を用いたオンライン学習

6 章では, 観測値に外れ値が混ざるような場合にも有効なガウス過程のオンライン学習として, ロバスト逐次ガウス過程の適用を提案する. ガウス過程は, カーネル関数としてガウスカーネルなどを選択した場合には, 一般に学習によるオーバーフィッティングが起こりにくく, 学習データが少ない場合には, 深層学習に比べ汎化性能に優れる. 一方で, ガウス過程や深層学習といったデータベースのモデルを用いてオンライン学習を行う場合, 外れ値がモデルの学習に大きな影響を及ぼす. 自動車エンジンのオンライン学習を考えた場合, 外れ値の混入は避けては通れないものであり, ロバストかつ高速なオンライン学習手法が望まれる. そこで本章ではオンライン学習手法である逐次ガウス過程にロバストカルマンフィルターの考え方を適用するロバスト逐次ガウス過程について述べた後, その有効性をエンジンシミュレータにて検証する.

7 章 結論

本研究で得られた成果の総括と，我々の研究の今後の発展と課題などについて述べる.

第2章　準備

2.1　はじめに

この章ではガウス過程 (Gaussian Process, 以下 GP)[20] についての概要を説明した後に, Society of Automotive Engineers (JSAE) と Society of Instrument and Control Engineers (SICE) から提供されているノッキングと失火のモデルを含むエンジンベンチマーク問題について説明する. GP は過学習の起こりにくい統計的モデルであり, 少量のデータでも精度の良いモデルの獲得が期待でき, 更に予測値として平均値のみならず, 分散値を獲得できるメリットがある. 一方でその計算コストは少なくなく, 大規模なデータへの適用や複雑で不連続な対象への適用には不向きである. そこで数々の近似計算手法が提案されているが, ここでは本論文で扱う近似手法についてまとめる.

2.2　ガウス過程回帰

2.2.1　ガウス過程

GP はカーネル法をベイズ確率理論に基づいて解析したものである. また, 関数がガウス過程として表され, すべてのデータがガウス分布に従うため, カーネル法により解析や応用が容易にできる. 機械学習における GP は回帰, 分類, 確率的主成分分析などの問題に適用されているが, ここでは回帰問題を考える.

ガウス過程回帰の目的は, 入力値 $\boldsymbol{x}_*$ が与えられた時にそれに相当する未知の出力 y_* を予測することである. N 組の学習データを考えたとき, 入力の学習データを $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{n_x \times N}$, 出力の学習データを $\boldsymbol{y} = [y_1, y_2, \ldots, y_N]^T \in \mathbb{R}^{N \times 1}$ とし, 学習データの集合を $\mathcal{D} = (\boldsymbol{X}, \boldsymbol{y})$ で表す. このときあるカーネル関数 $k(\cdot, \cdot)$ を用いて GP の事前分布を $f(\boldsymbol{x}) \sim \mathcal{GP}(\boldsymbol{0}, k(\boldsymbol{x}, \boldsymbol{x}'))$ で書くとする. また観測ノイズを σ^2, 即ち $y|\boldsymbol{x} \sim \mathcal{N}(f(\boldsymbol{x}), \sigma^2)$ とする. 予測分布はベイズ推論に基づき, 次の同時分布から求めることができる (付録 A.1).

$$\begin{bmatrix} \boldsymbol{f} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} k(\boldsymbol{X}, \boldsymbol{X}) & k(\boldsymbol{X}, \boldsymbol{x}_*) \\ k(\boldsymbol{x}_*, \boldsymbol{X}) & k(\boldsymbol{x}_*, \boldsymbol{x}_*) \end{bmatrix} \right) \tag{2.1}$$

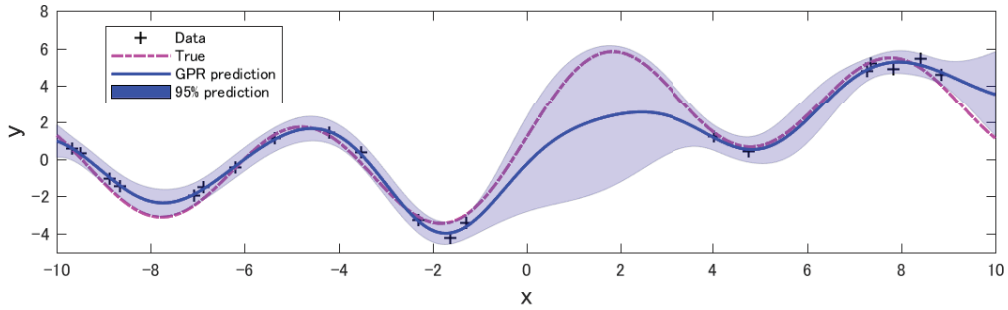


Fig 2.1: Example of Gaussian process regression. The predicted value can be represented by Gaussian distribution.

$$\begin{bmatrix} \boldsymbol{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}\ , \begin{bmatrix} k(\boldsymbol{X},\boldsymbol{X}) + \sigma^2 \boldsymbol{I} & k(\boldsymbol{X},\boldsymbol{x}_*) \\ k(\boldsymbol{x}_*,\boldsymbol{X}) & k(\boldsymbol{x}_*,\boldsymbol{x}_*) \end{bmatrix} \right) \tag{2.2}$$

ここで, $\boldsymbol{f} = [f_1, f_2, \ldots, f_N]$, $f_i = f(\boldsymbol{x}_i)$ である. このとき, 予測分布は以下のように算出できる.

$$\begin{aligned} y_*|\boldsymbol{x}_*, \mathcal{D} &\sim \mathcal{N}(y_*|\mu_{f*}, \sigma^2_{f*} + \sigma^2 \boldsymbol{I}) \\ \mu_{f*} &= k(\boldsymbol{x}_*,\boldsymbol{X})(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y} \\ \sigma^2_{f*} &= k(\boldsymbol{x}_*,\boldsymbol{x}_*) - k(\boldsymbol{x}_*,\boldsymbol{X})(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}k(\boldsymbol{X},\boldsymbol{x}_*) \end{aligned} \tag{2.3}$$

ただし, $\boldsymbol{K} = k(\boldsymbol{X},\boldsymbol{X})$ かつ, 行列 $\boldsymbol{X}$ が与えられたときのカーネル関数 $k(\cdot,\cdot)$ は以下で定義される.

$$\begin{aligned} k(\boldsymbol{X},\boldsymbol{X}) &= [k_{ij}],\ k_{ij} = k(\boldsymbol{x}_i,\boldsymbol{x}_j) \\ k(\boldsymbol{x}_*,\boldsymbol{X}) &= [k(\boldsymbol{x}_*,\boldsymbol{x}_1), \cdots, k(\boldsymbol{x}_*,\boldsymbol{x}_N)] = k(\boldsymbol{X},\boldsymbol{x}_*)^T \end{aligned}$$

GP は主にカーネル関数 $k(\boldsymbol{x},\boldsymbol{x}')$ によりその性質が定まる. 最も一般的なカーネル関数はガウスカーネルであり, SE(Squared Exponential) カーネルとも呼ばれ, 以下で示される.

$$\begin{aligned} k(\boldsymbol{x},\boldsymbol{x}') &= \mathrm{cov}\{f(\boldsymbol{x}), f(\boldsymbol{x}')\} \\ &= \exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{x}')^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{x}')}{2l^2} \right) \end{aligned} \tag{2.4}$$

σ も含んだパラメータ $\theta = [l^2, \sigma^2]$ はハイパーパラメータと呼ばれ、以下の周辺尤度最大化より最適なパラメータを見つける.

$$\begin{aligned} &\theta_{best} = \underset{\theta}{\mathrm{argmax}} \log(p(\boldsymbol{y}|\boldsymbol{X},\theta)) \\ &\log(p(\boldsymbol{y}|\boldsymbol{X},\theta)) \\ &= -\frac{1}{2}\boldsymbol{y}^{\mathrm{T}}(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y} - \frac{1}{2}\log|\boldsymbol{K} + \sigma^2 I| - \frac{N}{2}\log 2\pi \end{aligned} \tag{2.5}$$

この学習に必要な計算コストは $O(N^3)$ となり, GP が大規模データに向かない要因となる.

簡単な GP の例を Fig 2.1 に示す. 学習データを用いてハイパーパラメータを学習することで, 出力の予測値を確率分布で表すことができる. 本論文では特記のないかぎり, GP 及び GP の近似手法は GPML Matlab Code [26] をもとに, MATLAB R2017a 上で実装している.

2.2.2 異分散ガウス過程

通常 GP の観測ノイズ σ^2 は入力値に依存しないスカラー値であるため, 入力値に依存する分散を持つモデルは表現できない. そこで, 入力値に依存する分散を表現できるよう, 次のようなノイズモデルを追加した異分散ガウス過程 (Heteroscedastic Gaussian Process, HGP)[27]-[30] が提案されている.

$$y|\boldsymbol{x} \sim \mathcal{N}(f(\boldsymbol{x}), \sigma_n^2(\boldsymbol{x})) \tag{2.6}$$

$$v = \log(\sigma_n^2(\boldsymbol{x})) \sim \mathcal{GP}(m_n(\boldsymbol{x}), k_n(\boldsymbol{x}, \boldsymbol{x}')) \tag{2.7}$$

$p(v|\mathcal{D})$ はガウス分布にならず, 予測分布を解析的に計算するのは難しいことに注意する. ここでは, 期待値伝播法を応用する近似手法 [28] の例を示す. 期待値伝播法に関しては 3 章で扱うため詳細は省くが, この学習では, v の事後分布 $p(v|\mathcal{D})$ をガウス分布 $q(v|\mathcal{D})$ に近似し, 周辺尤度最大化によりノイズモデルを含めたハイパーパラメータを求める. ここではノイズモデルのカーネル関数を次のように選択する.

$$k_n(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}')^{\mathrm{T}} v^{-1} (\boldsymbol{x} - \boldsymbol{x}')) \tag{2.8}$$

入力 $\boldsymbol{x}_*$ を与えた時の予測出力 y_* の分布はガウス分布で近似した $q(v_*|\boldsymbol{x}_*, \mathcal{D}) = \mathcal{N}(\mu_{v*}, \sigma_{v*}^2)$ を用いて次のように計算できる.

$$\begin{aligned} p(y_*|\boldsymbol{x}_*, \mathcal{D}) &= \int p(y_*, v_*|\boldsymbol{x}_*, \mathcal{D}) dv_* \\ &= \int p(y_*, |v_*) q(v_*, |\boldsymbol{x}_*, \mathcal{D}) dv_* \end{aligned} \tag{2.9}$$

したがって, 予測出力 y_* の計算は以下のようにまとめられる.

$$
\begin{aligned}
y_*|\boldsymbol{x}_*, \mathcal{D} &\sim \mathcal{N}(\mu_*, \sigma_*^2) \\
\mu_* &= \iint y_* p(y_*, |v_*) q(v_*, |\boldsymbol{x}_*, \mathcal{D}) dy_* dv_* = \mu_{f*} \\
\sigma_*^2 &= \iint y_*^2 p(y_*, |v_*) q(v_*, |\boldsymbol{x}_*, \mathcal{D}) dy_* dv_* - \mu_*^2 \\
&= \int (f_*^2 + \exp(v_*)) q(v_*, |\boldsymbol{x}_*, \mathcal{D}) dv_* - \mu_*^2 \\
&= \sigma_{f*}^2 + \int \exp(v_*) q(v_*, |\boldsymbol{x}_*, \mathcal{D}) dv_* \\
&= \sigma_{f*}^2 + \exp(\mu_{v*} + 0.5\sigma_{v*}^2)
\end{aligned}
\tag{2.10}
$$

これは他のカーネル関数を選択した場合においても一般性を失わず適用可能である.

Fig. 1.3 のノックモデルに対し, 学習データ 150 点を用いた HGP モデルを Fig. 2.2 に示す. 一般の GP と比較し, HGP による予測結果では少量の学習データにも関わらず, ノックモデルの傾向をより良く表していることが分かる. 一方でその学習に必要な計算コストは GP と比較しても飛躍的に増加してしまうため, 大量のデータが必要となる複雑なモデルには不向きである. そのため, 今後本論文では HGP ではなく, 観測外乱がスカラー値をとる一般の GP のみを扱うものとする.

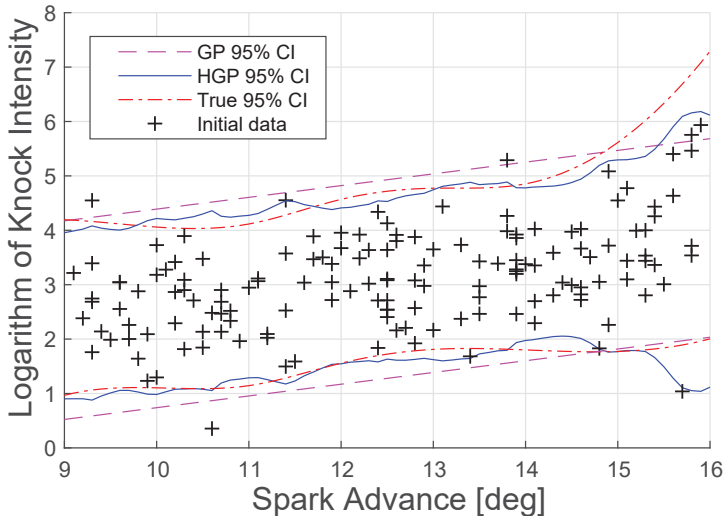
2.3 疑似入力を用いたガウス過程の近似手法

この節では GP に対して擬似入力の概念を導入することで、学習及び予測の計算コストを減らす近似手法 [31] について述べる．擬似入力の考え方では予測データである f_* と学習データとは独立であると仮定する．そして, それらの関係を M 個の疑似入力 $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_M]$ で表現することを考え, M を N より小さくとることで, 計算コストの削減を図る. その考え方を Fig. 2.3a に表す．太線は繋がっている全てのデータが相関を持つことを意味し, 細線は個々が相関を持つことを意味する．その関係を式で表すと以下のようになる．

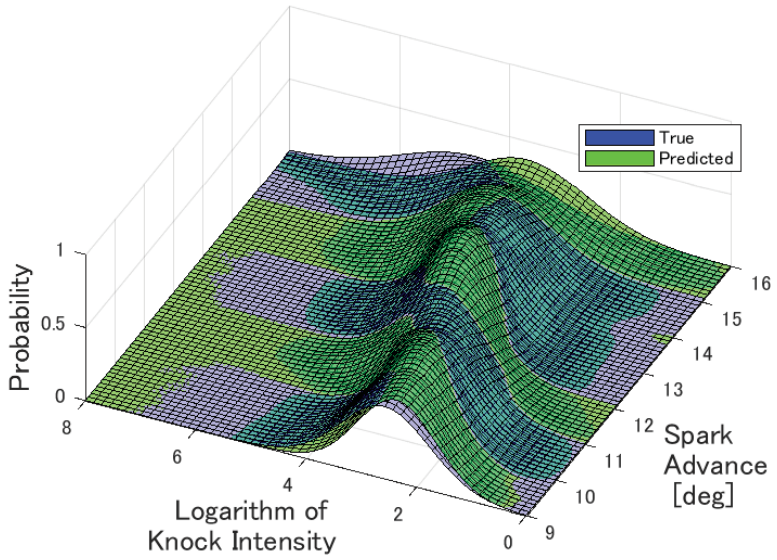
$$
p(f_*, \boldsymbol{f}) \approx q(f_*, \boldsymbol{f}) = \int q(f_*|\boldsymbol{U}) q(\boldsymbol{f}|\boldsymbol{U}) p(\boldsymbol{U}) d\boldsymbol{U} \tag{2.11}
$$

ここで、この関係式を用いた予測分布 $p(f_*|\boldsymbol{y})$ は以下のようになる．

$$
\begin{aligned}
p(f_*|\boldsymbol{y}) = \int p(f_*, \boldsymbol{f}|\boldsymbol{y}) d\boldsymbol{f} &= \frac{1}{p(\boldsymbol{y})} \int p(f_*, \boldsymbol{f}) p(\boldsymbol{y}|\boldsymbol{f}) d\boldsymbol{f} \\
&\approx \frac{1}{p(\boldsymbol{y})} \int q(f_*|\boldsymbol{U}) \int p(\boldsymbol{y}|\boldsymbol{f}) q(\boldsymbol{f}|\boldsymbol{U}) d\boldsymbol{f}\ p(\boldsymbol{U}) d\boldsymbol{U}
\end{aligned}
\tag{2.12}
$$



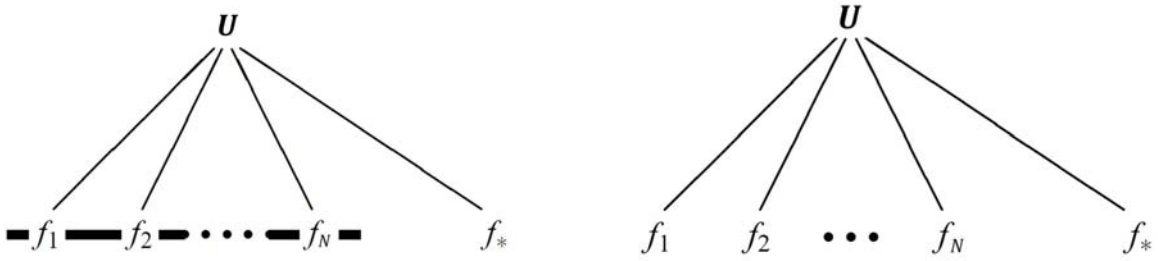
(a) Knock probability (2D view).



(b) Knock probability (3D view).

Fig 2.2: Example of heteroscedastic Gaussian process regression (150 data points).

擬似入力を用いた近似手法は $q(\boldsymbol{f}|\boldsymbol{U})$ の選び方により様々な手法が存在するがここではデータサブセット (Subset of Regressors, SoR) 近似 [32][33] 及び完全独立学習条件 (Fully Independent Training Conditional, FITC) 近似 [34] を紹介する.



(a) Pseudo input (b) FITC

Fig 2.3: Graphical model for GP using pseudo input.

2.3.1 データサブセット近似

SoR 近似は最も単純な近似手法の 1 つである. この手法では疑似入力 u と f_* 及び $\boldsymbol{f}$ の間に確定的な関係があるとしており, その関係式は次のようになる.

$$\begin{aligned} q_{SoR}(\boldsymbol{f}|\boldsymbol{U}) &= \mathcal{N}(K_{\boldsymbol{X},\boldsymbol{U}}K_{\boldsymbol{U},\boldsymbol{U}}^{-1}\boldsymbol{U}, \boldsymbol{0}) \\ q_{SoR}(f_*|\boldsymbol{U}) &= \mathcal{N}(k(\boldsymbol{x}_*, \boldsymbol{U})K_{\boldsymbol{U},\boldsymbol{U}}^{-1}, \boldsymbol{0}) \end{aligned} \tag{2.13}$$

ここで, $K_{\boldsymbol{X},\boldsymbol{X}'} = k(\boldsymbol{X}, \boldsymbol{X}')$ である. このとき SoR 近似の事前分布は次のようになる.

$$q_{SoR}(\boldsymbol{f}, f_*) = \mathcal{N}\left(\boldsymbol{0}\ , \begin{bmatrix} Q_{\boldsymbol{X},\boldsymbol{X}} & Q_{\boldsymbol{X},\boldsymbol{x}_*} \\ Q_{\boldsymbol{x}_*,\boldsymbol{X}} & Q_{\boldsymbol{x}_*,\boldsymbol{x}_*} \end{bmatrix}\right) \tag{2.14}$$

ただし, $Q_{\boldsymbol{X},\boldsymbol{X}'} \stackrel{\triangle}{=} K_{\boldsymbol{X},\boldsymbol{U}}K_{\boldsymbol{U},\boldsymbol{U}}^{-1}K_{\boldsymbol{U},\boldsymbol{X}'}$ と定義している. このとき予測分布は次で与えられる.

$$\begin{aligned} y_*|\boldsymbol{x}_*, \mathcal{D} &\sim \mathcal{N}(y_*|\mu_{f*}, \sigma_{f*}^2 + \sigma^2\boldsymbol{I}) \\ \mu_{f*} &= Q_{\boldsymbol{x}_*,\boldsymbol{X}}(Q_{\boldsymbol{X},\boldsymbol{X}} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y} \\ \sigma_{f*}^2 &= Q_{\boldsymbol{x}_*,\boldsymbol{x}_*} - Q_{\boldsymbol{x}_*,\boldsymbol{X}}(Q_{\boldsymbol{X},\boldsymbol{X}} + \sigma^2\boldsymbol{I})^{-1}Q_{\boldsymbol{X},\boldsymbol{x}_*} \end{aligned} \tag{2.15}$$

したがって, SoR 近似は GP において, そのカーネル関数を以下とすることと同義である (Fig. 2.4).

$$k_{SoR}(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}, \boldsymbol{U})K_{\boldsymbol{U},\boldsymbol{U}}^{-1}k(\boldsymbol{U}, \boldsymbol{x}') \tag{2.16}$$

この手法は非常に単純な構造を持つ近似手法であるものの, 予測分散値が大きくなり易い傾向にある.

2.3.2 完全独立学習条件近似

次に最も一般的に用いられる FITC 近似について述べる. これは SPGPs (Sparse Pseudo-input Gaussian Processes) とも呼ばれる [34]. Fig. 2.3b のようにこの手法では

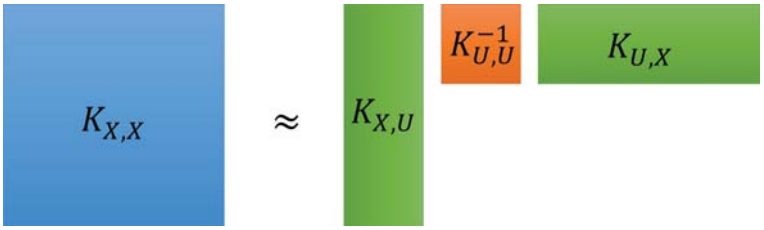


Fig 2.4: Concept of SoR approximation.

f_i は独立であると考え, その関係式を次で与える.

$$\begin{aligned} q_{FITC}(\boldsymbol{f}|\boldsymbol{U}) &= \Pi_{i=1}^{n} p(f_i|\boldsymbol{U}) = \mathcal{N}(K_{\boldsymbol{X},\boldsymbol{U}} K_{\boldsymbol{U},\boldsymbol{U}}^{-1} \boldsymbol{U}, \mathrm{diag}[K_{\boldsymbol{X},\boldsymbol{X}} - Q_{\boldsymbol{X},\boldsymbol{X}}]) \\ q_{FITC}(f_*|\boldsymbol{U}) &= p(f_*|\boldsymbol{U}) \end{aligned} \tag{2.17}$$

SoR 近似とは反対に, $\boldsymbol{f}$ と $\boldsymbol{U}$ との関係は確率的なものとして表現されている. このとき, FITC 近似の事前分布は次のようになる.

$$q_{FITC}(\boldsymbol{f}, f_*) = \mathcal{N}\left(\boldsymbol{0}\ , \begin{bmatrix} Q_{\boldsymbol{X},\boldsymbol{X}} - \mathrm{diag}[Q_{\boldsymbol{X},\boldsymbol{X}} - K_{\boldsymbol{X},\boldsymbol{X}}] & Q_{\boldsymbol{X},\boldsymbol{x}_*} \\ Q_{\boldsymbol{x}_*,\boldsymbol{X}} & K_{\boldsymbol{x}_*,\boldsymbol{x}_*} \end{bmatrix}\right) \tag{2.18}$$

また予測分布は次で与えられる.

$$\begin{aligned} y_*|\boldsymbol{x}_*, \mathcal{D} &\sim \mathcal{N}(y_*|\mu_{f*}, \sigma_{f*}^2 + \sigma^2 \boldsymbol{I}) \\ \mu_{f*} &= Q_{\boldsymbol{x}_*,\boldsymbol{X}}(Q_{\boldsymbol{X},\boldsymbol{X}} - \mathrm{diag}[Q_{\boldsymbol{X},\boldsymbol{X}} - K_{\boldsymbol{X},\boldsymbol{X}}] + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{y} \\ \sigma_{f*}^2 &= K_{\boldsymbol{x}_*,\boldsymbol{x}_*} - Q_{\boldsymbol{x}_*,\boldsymbol{X}}(Q_{\boldsymbol{X},\boldsymbol{X}} - \mathrm{diag}[Q_{\boldsymbol{X},\boldsymbol{X}} - K_{\boldsymbol{X},\boldsymbol{X}}] + \sigma^2 \boldsymbol{I})^{-1} Q_{\boldsymbol{X},\boldsymbol{x}_*} \end{aligned} \tag{2.19}$$

以上より, FITC 近似は GP において, そのカーネル関数を以下とすることと同義となる.

$$k_{FITC}(\boldsymbol{x}, \boldsymbol{x}') = k_{SoR}(\boldsymbol{x}, \boldsymbol{x}') + \delta_{\boldsymbol{x},\boldsymbol{x}'}\left(k(\boldsymbol{x}, \boldsymbol{x}') + k_{SoR}(\boldsymbol{x}, \boldsymbol{x}')\right) \tag{2.20}$$

FITC 近似において, その学習計算のレートは $O(M^2 N + M^3)$ であり, 学習後の平均値予測は $O(M)$, 分散値予測は $O(M^2)$ の計算量となる. 更に疑似入力 $\boldsymbol{U}$ はランダムに選択するだけでなく, 貪欲法を適用したり, ハイパーパラメータとして周辺尤度最大化により学習することもできる [33][34]. また, SoR や FITC においてはその計算コストは疑似入力点数 M に依存するが, あまり M を小さくしてしまうと予測精度に影響が出る (Fig. 2.6a, Fig. 2.6b) ため, 予測精度と計算コストのトレードオフにより M を決定することになる.

2.4 構造化カーネルを用いたガウス過程

GP 学習 (2.5) において計算上のボトルネックは $(\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{y}$ と $\log |\boldsymbol{K} + \sigma^2 \boldsymbol{I}|$ の算出となるが, カーネルがテプリッツ構造 [35][36] やクロネッカー構造 [37] といった特別な

構造を持つ場合, 効率的な計算が可能となる. 一方で, 格子点上に配置された学習データ点が必要になるため, 一般には適用が困難である. この節では, SoR や FITC といった疑似入力ベース GP の考え方を応用し, 疑似入力を格子点上に配置した上で構造化カーネルを適用することで効率的な GP 学習の実現を考える KISS-GP(Kernel interpolation for scalable structured Gaussian processes)[38] を紹介する.

2.4.1 テプリッツ構造

テプリッツ法は学習データ $\boldsymbol{x}_i$ が 1 次元上に等間隔に配置されており, カーネル関数が

$$k(x, x') = k(x - x') \tag{2.21}$$

を満たす場合に適用が可能である. このときテプリッツ行列を循環行列に埋め込むことができ, 高速フーリエ変換を用いることで行列とベクトルの積を高速に計算することができる (付録 B). 更に GP 学習の最適化問題 (2.5) も効率的に計算できる. その際に必要な $(\boldsymbol{K}+\sigma^2\boldsymbol{I})^{-1}\boldsymbol{y}$ の計算コストは $O(N \log N)$ のオーダーとなるが, 一方で $\log|\boldsymbol{K}+\sigma^2\boldsymbol{I}|$ の計算に必要なコストは $O(N^2)$ のオーダーとなる.

2.4.2 クロネッカー構造

クロネッカー法では多次元の学習データが格子上に配置されている ($\boldsymbol{x}_i \in \chi_1 \times \cdots \times \chi_P \subset \mathbb{R}^P$) 場合に次を満たすカーネル関数を考える.

$$k(\boldsymbol{x}, \boldsymbol{x}') = \prod_{p=1}^{P} k(\boldsymbol{x}^{(p)}, \boldsymbol{x}'^{(p)}) \tag{2.22}$$

このカーネル関数条件は一般的に使用されているガウスカーネルをはじめ多くの場合に適用可能である. 学習データ点数 N は $N = \prod_{p=1}^{P} n_p$ (n_p は次元毎の格子点の個数) で与えられる. このとき共分散行列 $\boldsymbol{K}$ はクロネッカー積を用いて, $\boldsymbol{K} = \boldsymbol{K}_1 \otimes \cdots \otimes \boldsymbol{K}_P$ と分解できる. ただし, $\boldsymbol{K}_p$ は次元毎に計算した共分散行列である. 更に, $\boldsymbol{K}_p$ 毎の固有値分解の結果を $\boldsymbol{K}_p = \boldsymbol{Q}_p\boldsymbol{V}_p\boldsymbol{Q}_p^T$ とすると, $\boldsymbol{Q} = \boldsymbol{Q}_1 \otimes \cdots \otimes \boldsymbol{Q}_P$ 及び $\boldsymbol{V} = \boldsymbol{V}_1 \otimes \cdots \otimes \boldsymbol{V}_P$ を用いて $\boldsymbol{K} = \boldsymbol{Q}\boldsymbol{V}\boldsymbol{Q}^T$ で計算できる. この分解に必要な計算コストは $O(PN^{\frac{3}{P}})$ となる. また, $(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y} = (\boldsymbol{Q}\boldsymbol{V}\boldsymbol{Q}^T + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y} = \boldsymbol{Q}(\boldsymbol{V} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{Q}^T\boldsymbol{y}$ かつ $\log|\boldsymbol{K} + \sigma^2\boldsymbol{I}| = \sum_i \log(\boldsymbol{V}_{ii} + \sigma^2)$ で計算でき, $\boldsymbol{V}$ は固有値から成る対角行列であるため, 関連する逆行列は簡単に計算できる. 最終的に $(\boldsymbol{K}+\sigma^2\boldsymbol{I})^{-1}\boldsymbol{y}$ の計算に必要なコストは $O(PN^{1+\frac{1}{P}})$ となる.

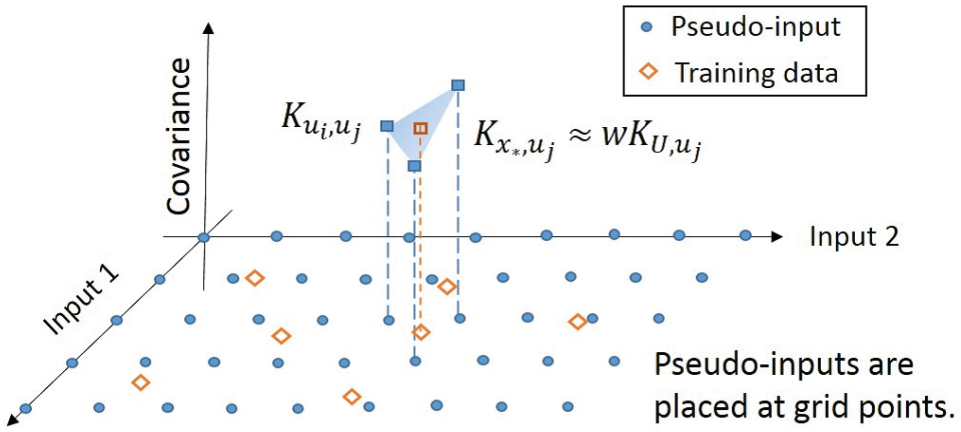


Fig 2.5: Concept of local kernel interpolation using SKI (linear case).

2.4.3 構造化カーネル補間

テプリッツ法やクロネッカー法といった構造化カーネルベース GP の適用を考えた場合, カーネル関数の制約以上に格子点上に配置された学習データが必要であることが大きな制約となる. 1 つの解決法は, 疑似入力を格子点上に配置することであるが, 必要な疑似入力の個数 M は一般に少なくなく, 場合によっては $M \gg N$ となる. そこで, 構造化カーネル補間 (Structured Kernel Interpolation, SKI)[38] によるスパース近似手法が提案されている (Fig. 2.5). この手法はガウスカーネルなどの一般のカーネル関数において, その滑らかさから, 学習点でのカーネルがその周辺の疑似入力のカーネルで局所的によく近似できるという仮定に基づいている. つまり次の近似を考える.

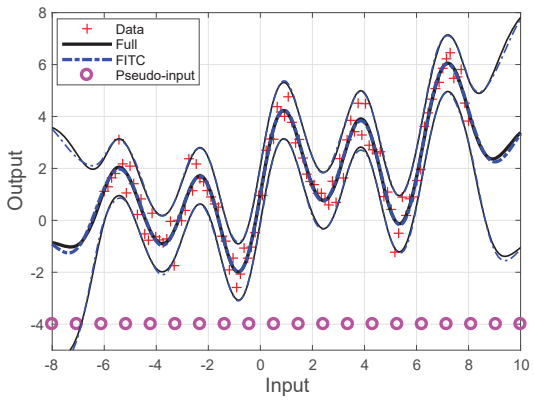
$$K_{\boldsymbol{X},\boldsymbol{U}} \approx \boldsymbol{W} K_{\boldsymbol{U},\boldsymbol{U}} \tag{2.23}$$

ここで, 重み行列 $\boldsymbol{W}$ は $N \times M$ の行列である. この重み行列は局所的な線形近似や 3 次関数近似 [39] などの近似手法により行毎の非零となる要素数が異なるものの, 一般に非常にスパースな行列となる. 特に 3 次関数近似は線形近似と比較し, 良い近似性能を示す (Fig. 2.6c, Fig. 2.6d).

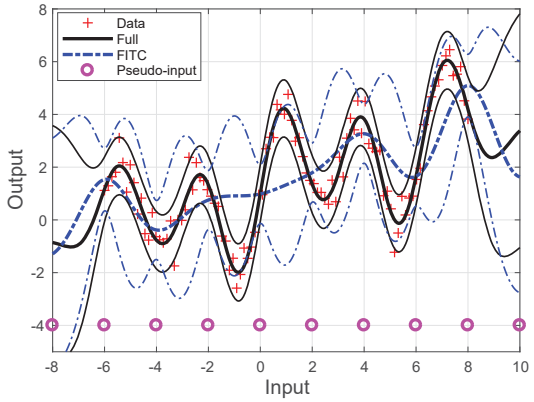
例えば, SoR 近似カーネル (2.16) を用いると, 共分散行列 $\boldsymbol{K}$ は以下で計算できる.

$$\begin{aligned} \boldsymbol{K} &\overset{SoR}{\approx} K_{\boldsymbol{X},\boldsymbol{U}} K_{\boldsymbol{U},\boldsymbol{U}}^{-1} K_{\boldsymbol{U},\boldsymbol{X}} \approx \boldsymbol{W} K_{\boldsymbol{U},\boldsymbol{U}} K_{\boldsymbol{U},\boldsymbol{U}}^{-1} K_{\boldsymbol{U},\boldsymbol{U}} \boldsymbol{W}^T \\ &= \boldsymbol{W} K_{\boldsymbol{U},\boldsymbol{U}} \boldsymbol{W}^T = \boldsymbol{K}_{SKI} \end{aligned} \tag{2.24}$$

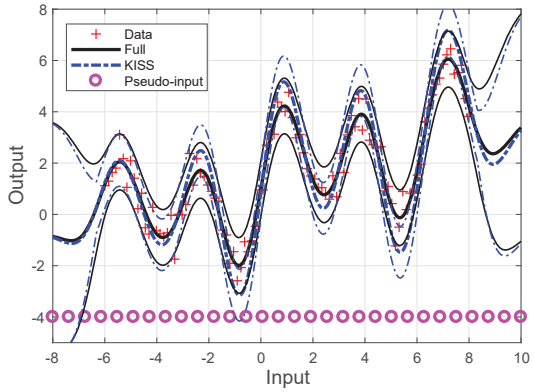
近似されたカーネル関数は SKI と呼ばれる. $\boldsymbol{K}_{SKI}\boldsymbol{y}$ を考えると, その計算コストは $O(N+M^2)$ のオーダーとなり, 更にカーネル関数がテプリッツ構造を持つ場合は $O(N+M\log M)$, クロネッカー構造を持つ場合は $O(PM^{1+\frac{1}{P}})$ となる. FITC 近似カーネル (2.20) も同様に適用できるが, $M > N$ の場合は GP 近似の場合ほど SoR との差はでないことに注意する.



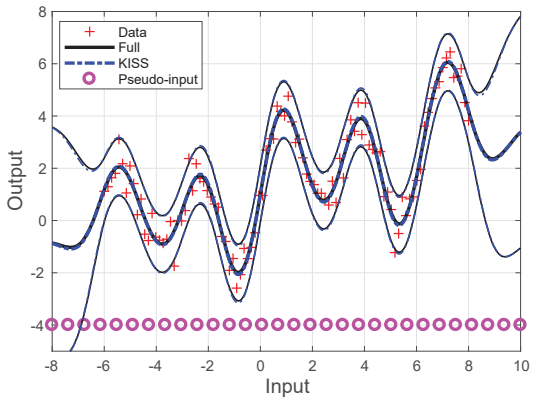
(a) FITC approximation (N=50, M=20)



(b) FITC approximation (N=50, M=10)



(c) KISS-GP (N=50, M=30, linear)



(d) KISS-GP (N=50, M=30, cubic)

Fig 2.6: Examples of toy model training using GP approximation methods.

構造化カーネルを用いた GP において, SKI に基づいたスパース近似を行ったものは KISS-GP と呼ばれる. しかしながら, クロネッカー構造は入力の次元 n_x が大きくなるにつれて, 必要な格子点上の入力点 M が多くなるので, 疑似入力ベース近似と比較して効果的でなくなるという課題がある.

2.5 深層カーネルの応用

この節では深層ニューラルネットワーク (Deep Neural Network, DNN) をカーネルの一部として構成する GP について述べる. ここでの目的は DNN によって学習データを低次元化することにより計算コストの削減を図ることである. 特に低次元化の効果は KISS-GP を適用する際に効果的である. 更に, 複雑な対象に対して良い回帰性能が得られるという利点もある.

深層学習と GP の組み合わせとしては, 深層信念ネットワーク (Deep Belief Network, DBN) と GP[40](DBN-GP), DNN と GP[41][42](DNN-GP) などが提案されている. DBN-

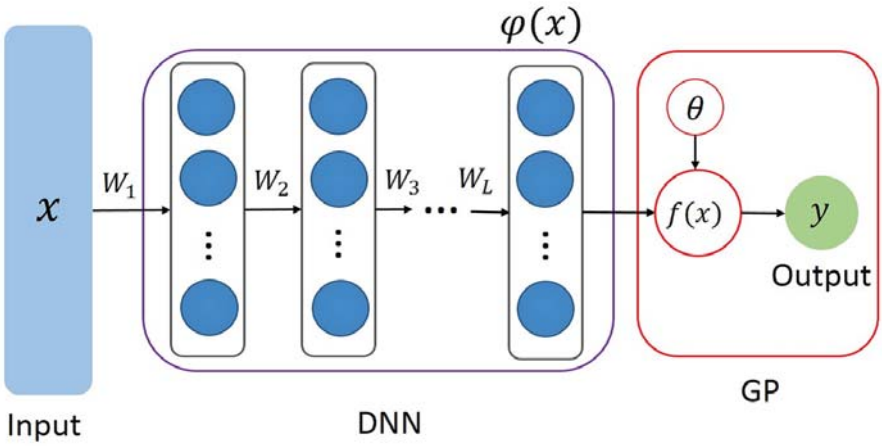


Fig 2.7: The illustration of DNN-GP model.

GP においては半教師有り学習に応用されているものの, DBN の本質は教師なし学習による事前学習にあり, 低次元化においての議論は不十分である. DNN-GP においては学習データの低次元化に注目しており, 特に文献 [41] では KISS-GP の応用を提案している. DNN-GP モデルの概要は Fig. 2.7 に示される. これは次のようなカーネル関数を用いた GP を考えることと同義である.

$$k_{DNN}(\boldsymbol{x}, \boldsymbol{x}') = k(\varphi(\boldsymbol{x}), \varphi(\boldsymbol{x}')) \tag{2.25}$$

このとき, DNN の重み行列を $\boldsymbol{W}$ とおくと, 新たに $\gamma = \{\theta, \boldsymbol{W}\}$ をハイパーパラメータとした GP の学習を考える. ただし, DNN は予め事前学習を行っているとする. ハイパーパラメータの学習においては次の対数周辺尤度を考える.

$$\begin{aligned}\mathcal{L} &= \log(p(\boldsymbol{y}|\boldsymbol{X}, \gamma)) \\ &= -\frac{1}{2}\boldsymbol{y}^{\mathrm{T}}(\boldsymbol{K}_{DNN} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y} - \frac{1}{2}\log|\boldsymbol{K}_{DNN} + \sigma^2 I| - \frac{N}{2}\log 2\pi \\ &= -\frac{1}{2}\boldsymbol{y}^{\mathrm{T}}\boldsymbol{K}_{\gamma}^{-1}\boldsymbol{y} - \frac{1}{2}\log|\boldsymbol{K}_{\gamma}| - \frac{N}{2}\log 2\pi\end{aligned} \tag{2.26}$$

この微分は次の連鎖律により計算できる.

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\theta} &= \frac{\partial\mathcal{L}}{\partial\boldsymbol{K}_{\gamma}}\frac{\partial\boldsymbol{K}_{\gamma}}{\partial\theta} \\ \frac{\partial\mathcal{L}}{\partial\boldsymbol{W}} &= \frac{\partial\mathcal{L}}{\partial\boldsymbol{K}_{\gamma}}\frac{\partial\boldsymbol{K}_{\gamma}}{\partial\varphi(\boldsymbol{x})}\frac{\partial\varphi(\boldsymbol{x})}{\partial\boldsymbol{W}} \\ \frac{\partial\mathcal{L}}{\partial\boldsymbol{K}_{\gamma}} &= \frac{1}{2}(\boldsymbol{K}_{\gamma}^{-1}\boldsymbol{y}\boldsymbol{y}^{T}\boldsymbol{K}_{\gamma}^{-1} - \boldsymbol{K}_{\gamma}^{-1})\end{aligned} \tag{2.27}$$

ここで, $\frac{\partial\varphi(\boldsymbol{x})}{\partial\boldsymbol{W}}$ は DNN 学習に用いられる一般の誤差逆伝播法により求めることができる.

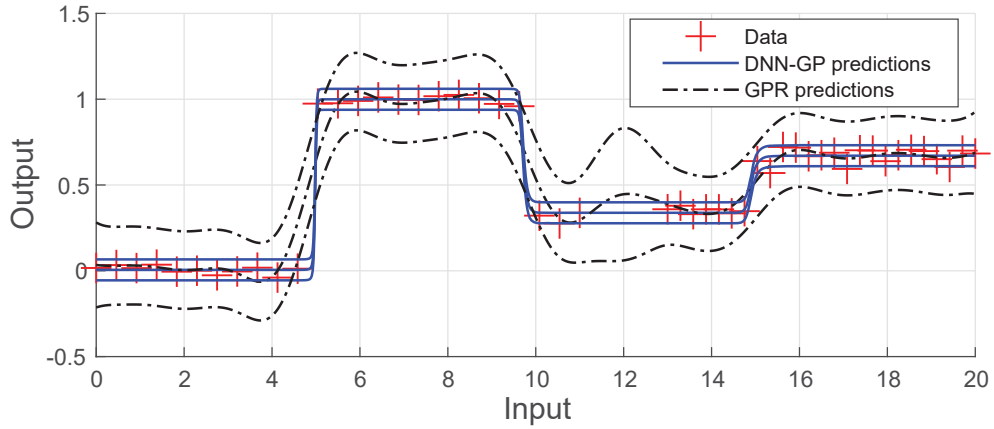


Fig 2.8: Example of toy model training using DNN-GP. DNN-GP can express discontinuous functions.

DNN は学習データからその特徴量を自動で抽出し, その低次元化が可能である. 前節で扱った KISS-GP は入力の次元が小さい場合には非常に高速な近似計算が可能であるが, 入力の次元が大きい場合には必要な疑似入力の多さから計算コストの削減効率が落ちてしまうという課題があった. そこで, DNN で獲得した低次元の特徴量に関して KISS-GP を適用することは合理的である (このような手法を以後 DGP と呼ぶことする). 勿論, DNN の事前学習が必要になることと GP 学習で扱うハイパーパラメータの数が多くなるため, カーネル学習に必要な計算コストは増加する傾向がある. しかしながら, KISS-GP の学習データに関する計算コストのオーダーは魅力的であり, DGP を採用するか否かは学習データの次元と学習データ点数, 確率的なモデルの必要性などで判断されることとなる.

簡単な対象への適用例を Fig. 2.8 に示す. DNN-GP は通常の GP での表現が難しい不連続なモデルに対しても適用が可能であることが分かる. 一方で低次元化された特徴量の影響が大きくなるため, 特徴量抽出が適切でない場合には予測が上手くいかない. また, ガウスカーネルなどと比べ過学習が起こりやすい傾向もある.

2.6 エンジンベンチマーク問題

JSAE-SICE のジョイントベンチャーから提供されているエンジンベンチマーク問題とエンジンシミュレータ [11] [12] の概要を述べた後, 本論文で扱う制御モデルの提案を行う.

2.6.1 境界モデル

自動車エンジンの制御の最適化が進むにつれ, ノッキングや失火といった異常運転領域近傍での制御が要求されている. 提供されているエンジンベンチマーク問題は, 境界近傍制御に有効なモデルベース制御系の開発促進を目的としたものである. ベンチマーク問題は, 境界モデルの構築と, 境界近傍制御系設計という 2 つの問題から構成される. ここで, 境界モデル $h : \mathbb{R}^{n_x} \to \mathbb{R}$ はある回帰ベクトル $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ を用いて以下で定義される.

$$h(\boldsymbol{x}_k) \begin{cases} > 0, & \text{outside of admissible domain} \\ = 0, & \text{on boundary} \\ < 0, & \text{inside of admissible domain} \end{cases} \tag{2.28}$$

k はエンジンサイクル数であり, 集合 $H := \{\boldsymbol{x}_k | h(\boldsymbol{x}_k) \leq 0\}$ は連結集合であると仮定する. また, 境界近傍制御問題は, $h(\boldsymbol{x}_k) \leq 0$ を制約条件とし, 目標値をエンジン実効トルク (以後トルクと表記する) とした, 消費燃料最小化問題と捉えることができる.

2.6.2 エンジンシミュレータ

ベンチマーク問題では, エンジンモデルと制御器から構成されるエンジンシミュレータ (Fig 2.9, Fig 2.10, Table 2.1) を扱う. エンジンモデルを MathWorks 社から提供されている MATLAB/Simulink を用いた数値モデルとして与えることで, 数値シミュレーションを用いた効率的な境界モデル設計と制御器設計検討が可能となる.

A. エンジンモデル

エンジンモデルでは, 4 気筒の自動車ガソリンエンジンを考えており, 吸気システムモデル, 筒内ガスモデル, 排気システムモデル, メカニカルシステムモデル, 筒内壁モデル, 外気モデルから構成される. 特に筒内ガスモデルにおいては, その燃焼プロファイルは実験データを使用した, 実機に近いものとなっており, 異常運転の 1 つであるノックモデルも導入されている.

ノック強度 $\boldsymbol{R\_KC} \in \mathbb{R}^l$ と失火検出信号 $\boldsymbol{M\_HC} \in \mathbb{R}^l$, トルク値 $\boldsymbol{Tor} \in \mathbb{R}^l$ は制御モデル同定には使用できるものの, 制御器が用いる出力としては使用できないことに注意する. ここで, l は気筒の数を示す. また, 下付き文字 $i = 1, 2, \ldots, l$ は対応する気筒の番号を示す. 例えば, 第 i 気筒のトルク値は $\boldsymbol{Tor}_i$ と記述する. これらの信号には, 実車には十分な精度をもつセンサを組み込めないものの, テストベンチでは詳細な測定が可能となる信号が該当する. 例えば, トルク値は制御目標値として用いるため, 観測出力からこれらの信号値を正確に推定できる数理モデルを獲得することは制御系設計

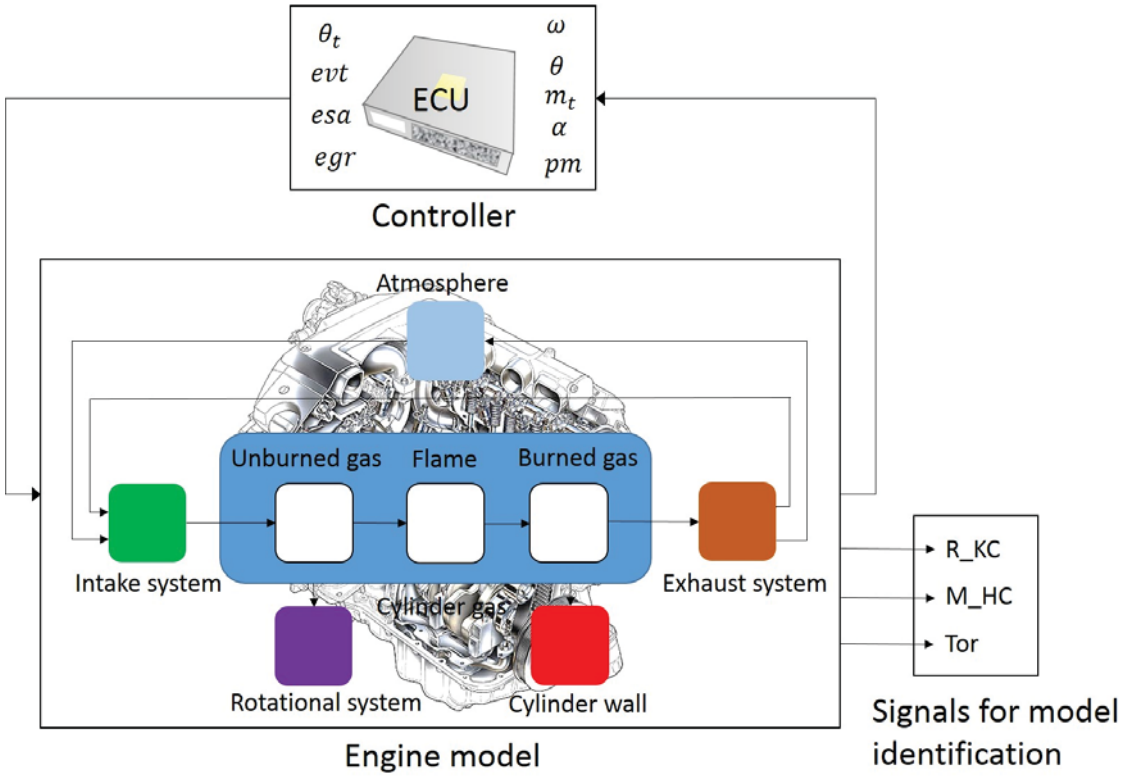


Fig 2.9: Configuration of the engine simulator.

において必要不可欠である. 推定が必要となる各気筒の評価信号をまとめたベクトル $\boldsymbol{z} \in \mathbb{R}^{n_z=3l}$ を以下で定義する.

$$\boldsymbol{z} = [\ \boldsymbol{R\_KC}^T,\ \boldsymbol{M\_HC}^T,\ \boldsymbol{Tor}^T\ ]^T \tag{2.29}$$

尚, 制御指標の 1 つである燃料消費量は観測出力から容易に計算できるとし, ここでは扱わない.

異常運転に関して, 本稿では, 未燃焼ガスがある状態で排気バルブが開いた場合を失火として定義する. また, ノックの発生は次で定義されるノック積分値が燃焼が終わるまでに 1 に到達するか否かで判定される.

$$\int_{t_{is}}^{t} \frac{1}{\phi_1 P_c(t)^{\phi_2} \exp\left(-\frac{\phi_3}{T_c(t)}\right)} dt \tag{2.30}$$

ここで t_{is} は燃焼が開始された時間, P_c はシリンダー内圧力, T_c は未燃ガス温度, ϕ_1, ϕ_2, ϕ_3 は定数である. 統計的モデルを同定する場合はエンジンモデルをブラックボックスとして扱うため, これらの異常運転モデルを陽に考慮する必要はないが, このエンジンモデルにおいてノックは確率的な現象として発生しないことに注意する.

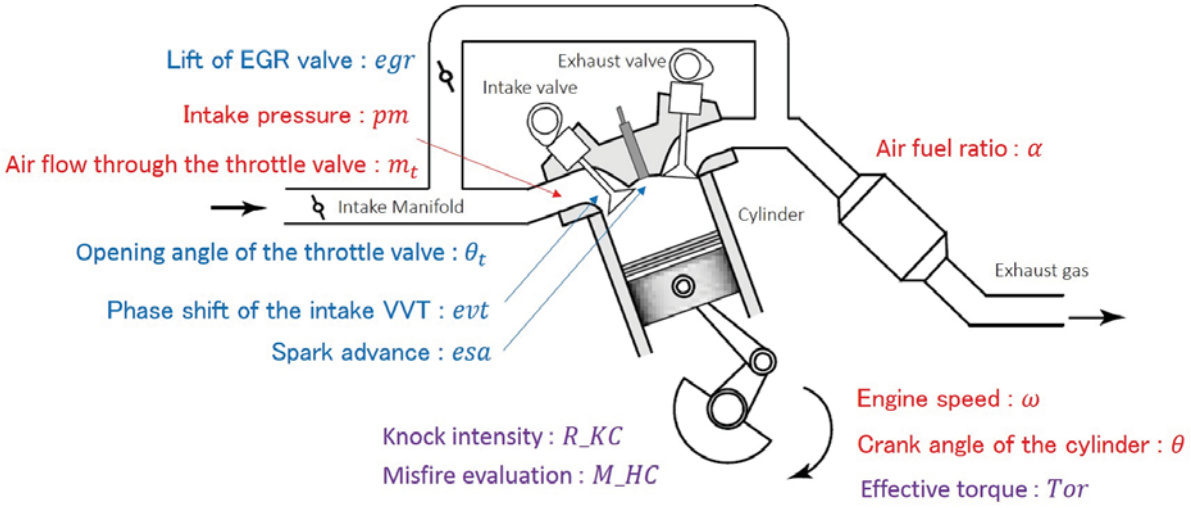


Fig 2.10: Overview of engine model.

Table 2.1: Definition of main variables.

Cateory	Symbol	Name
制御入力	θ_t	スロットルバルブ開度
	evt	インテーク VVT の位相シフト
	esa	点火時期
	egr	EGR バルブのリフト量
観測出力	ω	エンジン回転速度
	θ	第一気筒のクランク角度
	m_t	吸入空気量
	α	A/F センサからの信号値
	pm	吸気マニホルド内圧力
評価信号	$\boldsymbol{R\_KC}$	ノック強度信号値
	$\boldsymbol{M\_HC}$	失火検出信号値
	$\boldsymbol{Tor}$	平均実効トルク

B. 制御器

図 2.9 で示されるように, 制御器である Engine Control Unit (ECU) は 5 つの入力と 4 つの出力を持ち, これらの入出力の組を変更することはできない. また, 第一気筒の上死点からのクランク角度 θ を用いることで, サイクルベースの信号データを獲得することができる. ここでは簡単のため, 第一気筒の吸気・圧縮・燃焼・排気の 4 工程から 1 サイクルを構成するとし, 同サイクル内においては他気筒への入力も第一気筒と同じものを用いた制御器を考える. このとき, 以下のようなシステムの入力 $\boldsymbol{u}$ と観測出

力 $\boldsymbol{y}$ を定義する.

$$\begin{aligned} \boldsymbol{u} &= [\ \theta_t,\ evt,\ esa,\ egr\ ]^T && (2.31) \\ \boldsymbol{y} &= [\ \omega,\ m_t,\ \alpha,\ pm\ ]^T && (2.32) \end{aligned}$$

エンジンシステムにおいて特に重要な要素である, 空燃比 (Air/fuel ratio, A/F) α に関しては, 理想空燃比を目標値とした燃料噴射制御器が組み込まれているものの, 過渡応答においては, 必ずしも目標値に追従しない. その為, 燃料噴射系についても閉ループシステムのダイナミクスを考慮し, 制御入力を決定する必要がある. 但し, エンジンの代表的な運転条件は以下となる.

- エンジン回転速度 ω : 600[rpm] から 3000[rpm]
- スロットルバルブ開度 θ_t : 0[deg] から 90[deg]
- スロットルバルブ開度の開閉速度 : 90[deg]/150[ms] 以下
- インテーク VVT の位相シフト evt : 0 [deg] から 60 [deg]
- EGR バルブのリフト量 egr : 0[mm] から 10[mm]

2.6.3 制御モデル

エンジンの制御系設計においては正確で簡易な制御モデルの構築が必要であり, 実際のエンジン制御では物理モデルと実験データを用いた統計的モデルを組み合わせたグレーボックスモデルを構成することが多い [2]. 一方でノック現象など物理モデルでの表現が困難な対象に対して統計的モデルの適用が期待されている. そこで本論文では, 統計的モデルにのみ注目し, ブラックボックスモデリングの手法について述べる. 即ち, 与えられたエンジンモデルの事前知識を用いずに入出力データから直接制御モデルを同定することを考える.

入出力データから動的な構造を学習する基本的な手法の 1 つとして, Nonlinear Auto-Regressive with eXogenous inputs (NARX) モデルがある. 本論文では, トルクと境界推定モデルと組み合わせたシステム (2.33) をエンジンシステムの NARX モデルと定義し, その学習を試みる.

$$\begin{aligned} \boldsymbol{y}_{k+1} &= f(\boldsymbol{x}_k) + \boldsymbol{w}_k \\ \boldsymbol{z}_k &= g(\boldsymbol{x}_k) + \boldsymbol{v}_k && (2.33) \\ \boldsymbol{x}_k &= [\boldsymbol{y}_k^T, \cdots, \boldsymbol{y}_{k-p_o}^T, \boldsymbol{u}_k^T, \cdots, \boldsymbol{u}_{k-p_i}^T]^T \end{aligned}$$

ここで, $p_o \in \mathbb{N}$ 及び $p_i \in \mathbb{N}$ は NARX モデルの次数, $\boldsymbol{y}_k \in \mathbb{R}^{n_y}$, $\boldsymbol{z}_k \in \mathbb{R}^{n_z}$ は出力と信号の観測値, $\boldsymbol{x}_k \in \mathbb{R}^{n_x}, n_x = n_y(p_o+1) + n_u(p_i+1)$ はリグレッサーベクトル, $\boldsymbol{w}_k \in \mathbb{R}^{n_y}$ と $\boldsymbol{v}_k \in \mathbb{R}^{n_z}$ は外乱ベクトルである. 関数 $f(\cdot) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ は動的モデル, 関数 $g(\cdot) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_z}$ は境界及びトルクの推定モデルである. ノック強度 $\boldsymbol{R\_KC}$ と失火検知信号 $\boldsymbol{M\_HC}$ は, 通常運転値は 0 を, 異常運転値は正の連続値を出力するため, もし十分正確な関数 g が獲得できれば, 境界モデル h を求めることができる.

NARX モデルでは, 状態空間表現における状態変数といった潜在変数を推定せずに, 動的モデルを獲得することができる. また, 非線形系, むだ時間系, ハイブリッド系, 周期系といった特徴を持つ複雑なエンジンシステムに対しても直接適用が可能であるという利点を持つ.

また, このベンチマークにおいては自動車のメカニカルモデルが与えられていないため, 運転条件のうちエンジン回転速度は実験装置により任意に設定できるとしてモデル同定を行うこととする.

第3章　静的境界モデルの同定と静的実験計画法

3.1　はじめに

現在, 境界モデルは, 実機を用いたテストベンチにより多くの時間とコストをかけて作成されている. 本章の目的は, 正確な境界モデルを効率的に作成するアルゴリズムを提案することにある. 現状, 自動車エンジンの境界モデルは十分時間の経った, 静的状態での動作について考えられている. このモデルを獲得する為の入力列設計手法は, 実験計画法と呼ばれている. 自動車エンジンにおける実験計画法としては, 領域を凸包として仮定した手法 (e.g. Rapid Hull Determination Algorithm [44]) や多項式モデルの係数の分散を最小化する手法 (e.g. D 最適計画法 [45]) が広く知られている. しかしながら実際には運転可能領域は凸包とは仮定できず (Fig. 3.1), 多項式モデルで表現できるのも簡単なモデルだけであるため, これらの手法だけでは正確な境界モデルを獲得できない.

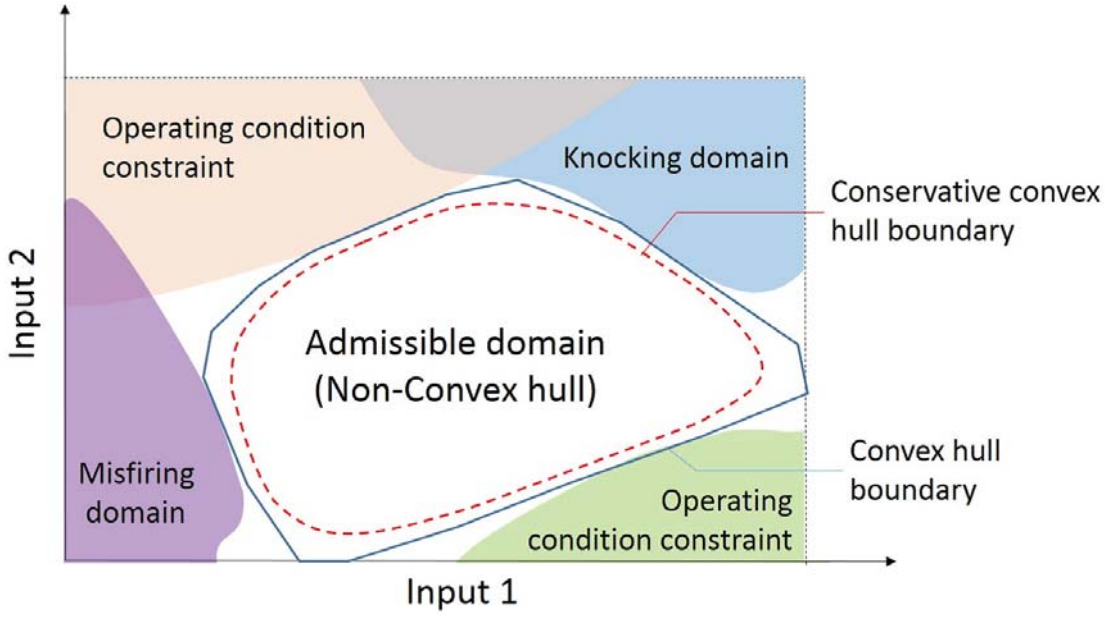


Fig 3.1: Non-convex admissible domain and conventional convex hull boundary model.

そこで文献 [46] では複雑な領域に対しても適用可能な, サポートベクターマシンを応用した実験計画法が提案されている. しかしながら, この手法で計画できるのは入力の探索方向のみであり, 静的境界の判別には探索方向に沿って十分な時間をかけて入力を加えていく必要がある. 本章では, 確率的モデルである期待値伝播法を用いたガウス過程分類器 (Gaussian Process Classifier, GPC)[20][47]-[49] を応用することで, 静的境界モデルを作成する手法を提案する. 更に生成モデルを用いて逐次的にシステムへの入力列を設計する, 効率的な入力列設計法を提案する. この手法は運転可能領域が非凸な場合でも適用可能であり, 入力候補の値を直接与えることができるため, 適合時間の大幅な短縮が期待できる. 提案手法の有用性は Society of Automotive Engineers (JSAE) と Society of Instrument and Control Engineers (SICE) から提供されているノッキングと失火のモデルを含むエンジンベンチマーク問題 [11][12] にて検証する.

3.2 静的境界モデリング

境界モデル同定では, (2.28) のモデル $h(\boldsymbol{x}_k)$ を決定することが目的となるが, 対象が複雑となると, リグレッサーベクトル $\boldsymbol{x}_k$ の選定方法や同定に必要な動的なデータ獲得手法など多くの課題が残る. よって, 実際的には十分時間が経過した定常状態での境界モデルを同定する場合が多い. これを通常の境界モデリングと区別するために静的境界モデリングと定義する.

静的境界モデルでは, 十分な時間が経過した時の観測出力を $\boldsymbol{y}_\infty \in \mathbb{R}^{n_y}$, 一定入力ベクトルを $\boldsymbol{u}_\infty \in \mathbb{R}^{n_u}$ と定義すると, 以下のモデルを同定することとなる.

$$h_S\left(\begin{bmatrix} \boldsymbol{y}_\infty \\ \boldsymbol{u}_\infty \end{bmatrix}\right)\begin{cases} > 0, & \text{outside of admissible domain} \\ = 0, & \text{on boundary} \\ < 0, & \text{inside of admissible domain} \end{cases} \tag{3.1}$$

また, 観測出力 $\boldsymbol{y}_\infty$ の要素が一定入力 $\boldsymbol{u}_\infty$ によって決定されるとき, その要素はリグレッサーベクトルから省くことができる. 特別な場合として $\boldsymbol{y}_\infty$ が一定入力により決定されるとき, 静的境界モデルは入力のみの関数 $h(\boldsymbol{u}_\infty)$ で与えられる.

$$h(\boldsymbol{u}_\infty)\begin{cases} > 0, & \text{outside of admissible domain} \\ = 0, & \text{on boundary} \\ < 0, & \text{inside of admissible domain} \end{cases} \tag{3.2}$$

自動車エンジンシステムの静的境界モデル同定では式 (3.2) のモデルを考えるとする. 自動車エンジンの静的境界モデリング手法としては, Rapid Hull Determination Algorithm [44] や D 最適計画法 [45] が代表的な手法として挙げられる. 従来手法の共通の課題としては, 探索方向のみが与えられるため, エンジンにダメージを与えないよう, 入力を

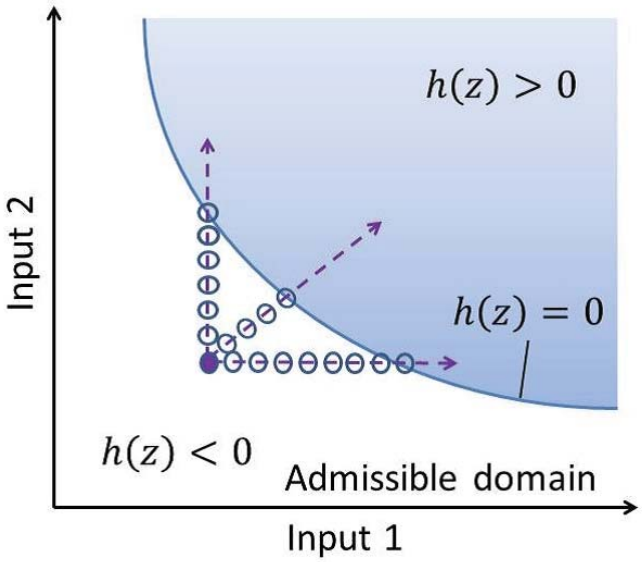


Fig 3.2: Example of boundary modeling strategy. To avoid engine damage, the inputs are changed slowly and usually in small discrete steps.

慎重に変化させていく必要があり, 探索に時間がかかることがある. $\mathbf{z} = \boldsymbol{u}_\infty$ とおいた場合の基本的な考え方を Fig. 3.2 に示す.

3.3 GP による分類

この節では, GP を応用した, 確率的なアプローチによる分類 [20][47] の簡単な概要を述べる. n 個の観測された学習データの集合を $\mathcal{D} = \{(\mathbf{z}_i, a_i)|i = 1, \ldots, n\}$ とする. ここで, $\mathbf{z}_i \in \mathbb{R}^{n_u}$ は入力ベクトル, 出力データ $a_i \in \{-1, 1\}$ は対応するスカラー値である. このとき, $\mathbf{Z} = [\mathbf{z}_1\ \mathbf{z}_2\ \cdots\ \mathbf{z}_n]$ かつ $\mathbf{a} = [a_1\ a_2\ \cdots\ a_n]$ とおくと, 学習データ集合は $\mathcal{D} = (\mathbf{Z}, \mathbf{a})$ と書ける. GPC では, 集合 $\mathcal{D}$ が与えられたとき, 新たなサンプル $\mathbf{z}_*$ のクラスラベルを事後確率 $p(a_*|\mathcal{D}, \mathbf{z}_*)$ を用いて予測する.

GPC において, クラスラベルの確率はある $\mathbf{z}$ の潜在関数 $f(\mathbf{z}) \in \mathbb{R}$ で表すことができると仮定する. すなわち, 2 値分類において, $p(a = +1|\mathbf{z}, f(\mathbf{z}), \mathcal{D}) = p(a = +1|f(\mathbf{z}))$ であるとする. $a = +1$ の観測される確率が $f(\mathbf{z})$ の単調増加関数で書けるとすると, ロジスティック関数やプロビット関数等のいくつかの形式で記述できる. $f_i = f(\mathbf{z}_i)$ を例にすると,

$$p(a_i = +1|f_i) = \begin{cases} \frac{1}{1+\exp(-a_i f_i)}, & \text{logistic function} \\ \Phi(a_i f_i), & \text{probit function} \end{cases} \tag{3.3}$$

と書ける. ここで, Φ はガウス累積分布関数 $\Phi(z) = \int_{-\infty}^{z} (1/\sqrt{2\pi}) \exp(-(z^2/2))dz$ である.

新たなサンプル $\mathbf{z}_*$ の出力確率予測は次のような潜在関数 f_* の積分で与えられる.

$$p(a_* = +1|\mathcal{D}, \mathbf{z}_*) = \int p(a_*|f_*)p(f_*|\mathcal{D}, \mathbf{z}_*)df_* \tag{3.4}$$

積分内の第 2 項はサンプル $\mathbf{z}$ に対する潜在変数の分布を示しており, $\mathbf{f} = [f_1\ f_2\ \cdots\ f_n]$ を周辺化することによって得られる.

$$p(f_*|\mathcal{D}, \mathbf{z}_*) = \int p(f_*|\mathbf{Z}, \mathbf{z}_*, \mathbf{f})p(\mathbf{f}|\mathcal{D})d\mathbf{f} \tag{3.5}$$

$p(\mathbf{f}|\mathcal{D})$ は潜在変数の事後確率であり, ベイズの定理より以下のように変形できる.

$$\begin{aligned} p(\mathbf{f}|\mathcal{D}) &= p(a|\mathbf{f})p(\mathbf{f}|\mathbf{Z})/p(a|\mathbf{Z}) \\ &= \left(\prod_{i=1}^{n} p(a_i|f_i)\right) p(\mathbf{f}|\mathbf{Z})/p(a|\mathbf{Z}) \end{aligned} \tag{3.6}$$

この $p(a|\mathbf{f})$ は尤度関数と呼ばれ, 式 (3.3) のいづれかを用いて表される. $p(a|\mathbf{Z})$ は周辺尤度, $p(\mathbf{f}|\mathbf{Z})$ は潜在変数に関する GP の事前分布であり, 次のように書ける.

$$p(\mathbf{f}|\mathbf{Z}) = \frac{1}{(2\pi)^{n/2}\,|\mathbf{K}|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\mathbf{f}-\mu)^T\mathbf{K}^{-1}(\mathbf{f}-\mu)\right\} \tag{3.7}$$

平均値 μ は通常 $\mu = \mathbf{0}$ と仮定される. 共分散行列の項 $\mathbf{K}_{ij}$ は $\mathbf{z}_i$ と $\mathbf{z}_j$ との関数 $k(\mathbf{z}_i, \mathbf{z}_j)$ である. この共分散関数はカーネルであり, データから学習したハイパーパラメータにより決定される.

3.4 期待値伝播法

通常, 尤度はガウス分布とならない為, 式 (3.4) と式 (3.5) の積分は解析的に計算できず, 出力確率を予測するには, 解析的な近似やサンプリング法を用いる必要がある. 解析的な近似手法としてはラプラス近似や期待値伝播法があるが, この節では入力列設計において特に良い結果を示した期待値伝播法 [20][48] のアルゴリズムをまとめる.

期待値伝播法は真の事後分布 $p(\mathbf{f}|\mathcal{D})$ とそのガウス近似 $q(\mathbf{f}|\mathcal{D})$ とのカルバック・ライブラー情報量 $\mathrm{KL}(p(\mathbf{f}|\mathcal{D})\|q(\mathbf{f}|\mathcal{D}))$ を局所的に最小化する. ベイズの定理より事後分布 $p(\mathbf{f}|\mathcal{D})$ は次のように書ける.

$$p(\mathbf{f}|\mathcal{D}) = \frac{1}{L}p(\mathbf{f}|\mathbf{Z})\prod_{i=1}^{n} p(a_i|f_i) \tag{3.8}$$

ここで正規化項 L は周辺尤度 $p(a|\mathbf{Z})$ である. 期待値伝播法では尤度を近似する為に, 潜在変数 f_i の関数を通して局所的な尤度 t_i を掛け合わせる.

$$p(a_i|f_i) \cong t_i(f_i|\tilde{L}_i, \tilde{\mu}_i, \tilde{\sigma}_i^2) = \tilde{L}_i\mathcal{N}(f_i|\tilde{\mu}_i, \tilde{\sigma}_i^2) \tag{3.9}$$

この $\tilde{L}_i, \tilde{\mu}_i, \tilde{\sigma}_i^2$ はサイトパラメータと呼ばれる．このとき尤度の近似は次式で与えられる．

$$\prod_{i=1}^{n} t_i(f_i|\tilde{L}_i, \tilde{\mu}_i, \tilde{\sigma}_i^2) = \mathcal{N}(\tilde{\mu}, \tilde{\mathbf{\Sigma}}) \prod_{i=1}^{n} \tilde{L}_i \tag{3.10}$$

ここで $\tilde{\mu} = [\mu_1\ \mu_2\ \cdots\ \mu_n]^T, \tilde{\mathbf{\Sigma}} = \mathrm{diag}\left[\tilde{\sigma}_1^2\ \tilde{\sigma}_2^2\ \cdots\ \tilde{\sigma}_n^2\right]$ である．$p(\mathbf{f}|\mathcal{D})$ のガウス近似により，事後分布は，

$$\begin{aligned} q(\mathbf{f}|\mathcal{D}) &= \frac{1}{L_{EP}} p(\mathbf{f}|\mathbf{Z}) \prod_{i=1}^{n} t_i(f_i|\tilde{L}_i, \tilde{\mu}_i, \tilde{\sigma}_i^2) = \mathcal{N}(\tilde{\mu}, \mathbf{\Sigma}) \\ &\text{with}\ \ \mu = \mathbf{\Sigma}\tilde{\mathbf{\Sigma}}^{-1}\tilde{\mu}\ \ \text{and}\ \ \mathbf{\Sigma} = (\mathbf{K}^{-1} + \tilde{\mathbf{\Sigma}}^{-1})^{-1} \end{aligned} \tag{3.11}$$

となる．ここで，$L_{EP} = p(a|\mathbf{Z})$ は正規化項である．期待値伝播法では各 t_i の近似を以下の空洞分布を用いて反復的に推定する．

$$q_{-i}(f_i) \propto \int p(\mathbf{f}|\mathbf{Z}) \prod_{j \neq i} t_j(f_i|\tilde{L}_j, \tilde{\mu}_j, \tilde{\sigma}_j^2) df_j \tag{3.12}$$

ここで，下付き文字 $-i$ は i が除かれる場合を示す．推定は学習データ毎に下記の 4 つのステップを繰り返すことで収束するまで行われる．

1. 空洞分布を計算する．

$$\begin{aligned} &q_{-i}(f_i|z_i, a_i) = \mathcal{N}(f_i|\mu_{-i}, \sigma_{-i}^2) \\ &\text{with}\ \ \mu_{-i} = \sigma_{-i}^2(\sigma_i^{-2}\mu_i - \tilde{\sigma}_i^{-2}\tilde{\mu}_i) \\ &\ \ \text{and}\ \ \sigma_{-i} = (\sigma_i^{-2} - \tilde{\sigma}_i^{-2})^{-1} \end{aligned} \tag{3.13}$$

2. 空洞分布を真の尤度と組み合わせ，目的の非ガウス周辺分布を得る

$$r(f_i) = q_{-i}(f_i|z_i, a_i) p(a_i|f_i) \tag{3.14}$$

3. 非ガウス周辺分布をガウス分布で近似する

$$\begin{aligned} &\hat{q}_i(f_i) = \hat{L}_i \mathcal{N}(\hat{\mu}_i, \hat{\sigma}_i^2) \cong r(f_i|z_i, a_i) \\ &\hat{L} = \Phi(z_i), \quad \hat{\mu}_i = \mu_{-1} + \frac{a_i \hat{\sigma}_{-i}^2 \mathcal{N}(z_i)}{\Phi(z_i)\sqrt{1 + \sigma_{-i}^2}} \\ &\hat{\sigma}_i^2 = \sigma_{-i}^2 - \frac{\hat{\sigma}_{-i}^4 \mathcal{N}(z_i)}{(1 + \sigma_{-i}^2)\Phi(z_i)} \left(z_i + \frac{\mathcal{N}(z_i)}{\Phi(z_i)} \right) \\ &\text{with}\ \ z_i = a_i \mu_{-i} / \sqrt{1 + \sigma_{-i}^2} \end{aligned} \tag{3.15}$$

4. 事後分布が目的の分布になるよう近似パラメータ t_i を更新する,

$$\tilde{\mu}_i = \tilde{\sigma}_i^2(\hat{\sigma}_i^{-2}\hat{\mu}_i - \sigma_{-i}^{-2}\tilde{\mu}_{-i}), \quad \tilde{\sigma}_i = (\hat{\sigma}_i^{-2} - \sigma_{-i}^{-2})^{-1}$$
$$\tilde{L}_i = \hat{L}_i\sqrt{2\pi}\sqrt{\sigma_{-i}^2 + \tilde{\sigma}_{-i}^2}\exp\left(\frac{(\mu_{-i} - \tilde{\mu}_i)^2}{2(\sigma_{-i}^2 + \tilde{\sigma}_{-i}^2)}\right) \tag{3.16}$$

3.4.1 予測

最終的にサンプル点 $\mathbf{z}_*$ の予測は以下で近似できる,

$$p(a_* = +1|\mathcal{D}, \mathbf{z}_*) \cong q(a_* = +1|\mathcal{D}, \mathbf{z}_*) \tag{3.17}$$
$$= \int p(a_*|f_*)q(f_*|\mathcal{D}, \mathbf{z}_*)df_*$$

ここで,

$$\mu_* = k(\mathbf{z}_*, \mathbf{Z})(\boldsymbol{K} + \tilde{\Sigma})^{-1}\tilde{\mu} \tag{3.18}$$
$$\sigma_*^2 = k(z_*, z_*) - k(\mathbf{z}_*, \mathbf{Z})(\boldsymbol{K} + \tilde{\Sigma})^{-1}k(\mathbf{Z}, \mathbf{z}_*) \tag{3.19}$$

特にプロビット関数で変換する場合には次のように解析的に与えられる.

$$q(a_* = +1|\mathcal{D}, \mathbf{z}_*) = \Phi\left(\frac{\mu_*}{\sqrt{1 + \sigma_*^2}}\right) \tag{3.20}$$

3.4.2 ハイパーパラメータの学習

ハイパーパラメータベクトルを θ とする. ハイパーパラメータは生成モデルの正確さに影響を与えるが, 通常は周辺尤度を最大化するような θ を選択する. 近似された周辺尤度は次の正規化因子 L_{EP} で定まる

$$L_{EP} = q(\mathbf{a}|\mathbf{Z}, \theta)$$
$$= \int p(\mathbf{a}|\mathbf{Z}, \theta)\prod_{i=1}^{n} t_i(f_i|\tilde{L}_i, \tilde{\mu}_i, \tilde{\sigma}_i^2)d\mathbf{f} \tag{3.21}$$

このとき対数周辺尤度は以下で記述できる

$$\log L_{EP} = -\frac{1}{2}\log|\mathbf{K} + \tilde{\mathbf{\Sigma}}| - \frac{1}{2}\tilde{\mu}^T(\mathbf{K} + \tilde{\mathbf{\Sigma}})^{-1}\tilde{\mu}$$
$$+ \sum_{i=1}^{n}\log\Phi\left(\frac{a_i\mu_{-i}}{\sqrt{1 + \sigma_{-i}^2}}\right) + \frac{1}{2}\sum_{i=1}^{n}\log(\sigma_{-i}^2 + \tilde{\sigma}_{-i}^2)$$
$$+ \sum_{i=1}^{n}\frac{(\mu_{-i} - \tilde{\mu}_i)^2}{2(\sigma_{-i}^2 + \sigma_{-i}^2)} \tag{3.22}$$

ハイパーパラメータベクトル θ_* は式 (3.22) の対数周辺尤度を最大化することで得られる.

3.5 静的実験計画法

GPC においては, 確率モデルだけではなく, 予測分散の近似モデルを与えることができる. そこで, 予測分散の情報を用いて境界を効率的に探索する手法を提案する. これは, 境界を横切るときに出力データ値 a の変化が発生する為, その近傍では予測値の分散が大きくなるという考えに基づいている. 今後, 領域内の出力データは -1 で, 領域外の出力データは $+1$ で獲得し, 領域外となる確率 $p(a_* = +1|\mathcal{D}, \mathbf{z}_*)$ を考えることとする. 追加入力の数を l とした場合の入力列設計アルゴリズムを Algorithm 1 にまとめる. また, その概要を Fig. 3.3 に示す. ここで, パラメータ $P \in \mathbb{R}$ は境界モデルの保守性を表すパラメータであり, この値が小さいほど, 領域外となる確率の低い領域で追加点を探索することになる. この値を大きくした場合, より積極的な境界探索を行い, モデリングに必要な追加入力点数を少なくできる可能性があるが, 同時に境界から大きく離れた領域外の点を探索する恐れもでてくる.

Algorithm 1 Static Boundary DoE Based on Gaussian Process Classifier

1: **Step 0**: Set some classified points.
2: **Loop**: For $k = 0, 1, \ldots, l-1$, perform:

Step 1. Generate a predictive probabilities model by GPC algorithm and estimate the admissible domain $H_k := \{\mathbf{z}|p(a_* = +1|\mathcal{D}, \mathbf{z}_*) < P\}$.

Step 2. Determine a next measurement point $\mathbf{z}_{k+1} = \mathrm{argmax}_{\mathbf{z}_* \in H_k} \sigma_*$ (e.g. in (3.19)).

Step 3. Apply the point $\mathbf{z}_{k+1}$ to the system and get an output data a_k.

End loop

3: **Output:** Generate a predictive probabilities model by GPC and estimate the boundary model $h(\mathbf{z}_*) =: p(a_* = +1|\mathcal{D}, \mathbf{z}_*) - P$.

3.6 適用例

本節ではまず GPC で扱う近似手法について, 簡単な対象を用いて比較を行い, 期待値伝播法の有用性を示す. その後に期待値伝播法で近似を行う GPC を応用した入力列設計手法をエンジンベンチマーク問題に適用し, 静的境界モデルを作成する. カーネ

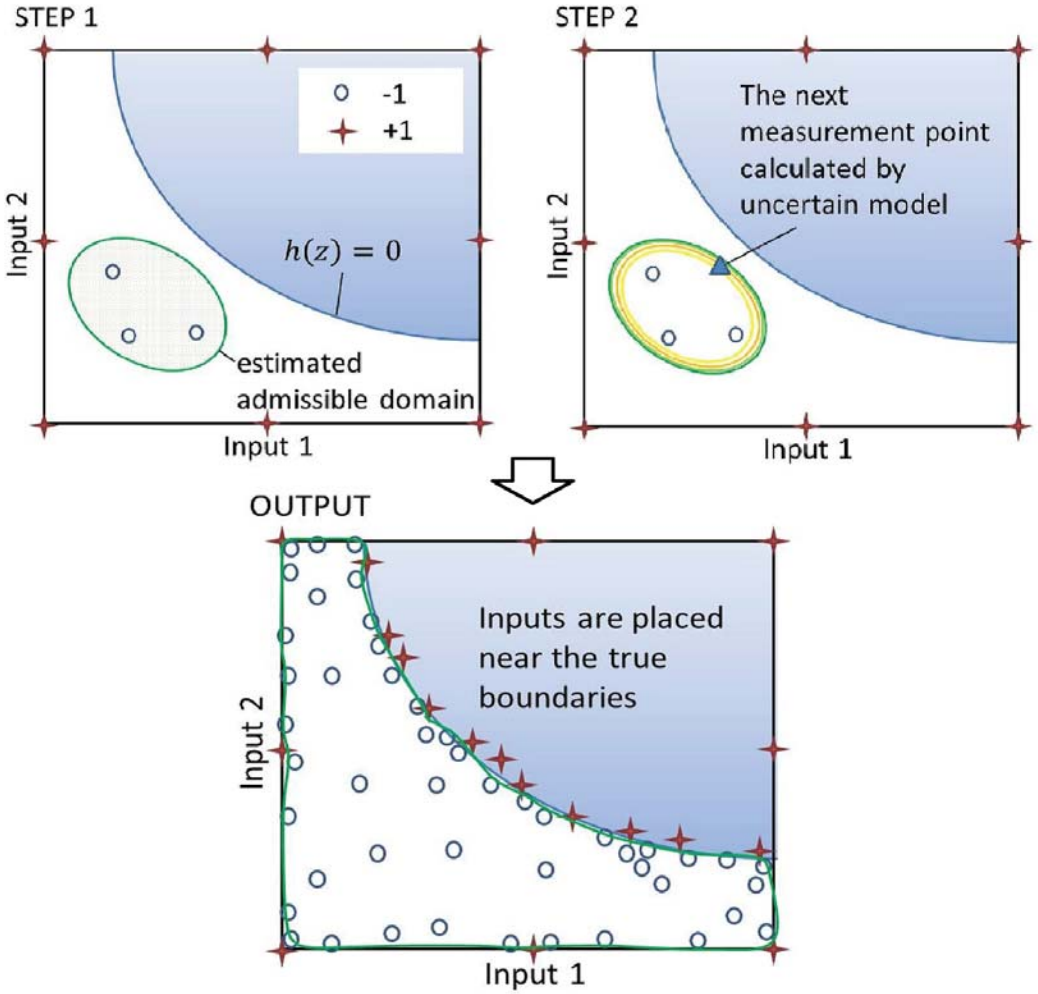


Fig 3.3: Input design based on GPC. This approach can determine the next measurement "point" near the boundaries.

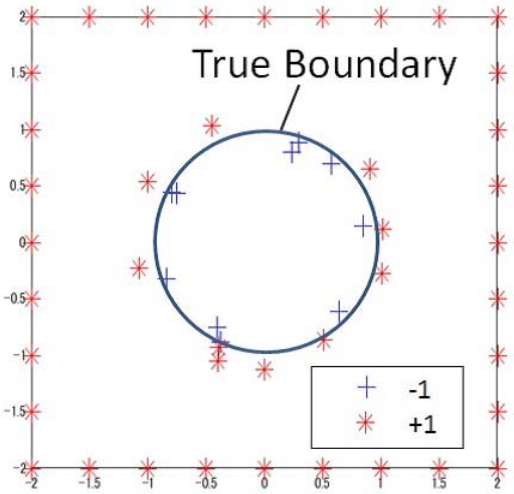


Fig 3.4: Initial condition of unit circle modeling. We set some classified points.

ル関数として以下のものを考え, 関連度自動決定 (Automated Relevance Determination, ARD) の枠組みを加える [20].

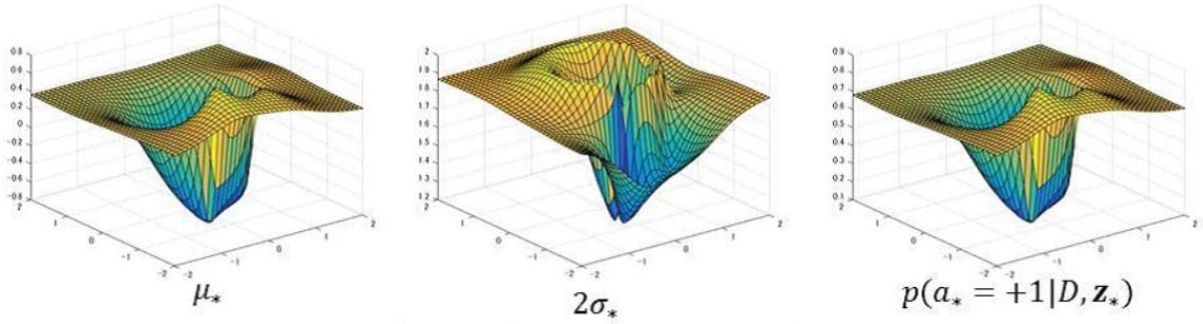
$$k(\mathbf{z}_i, \mathbf{z}_j) = \theta_0 \exp\left\{-\frac{1}{2}\sum_{p=1}^{n_u} \theta_m (z_i^p, z_j^p)\right\} \tag{3.23}$$

ここで z_i^m は $\boldsymbol{z}_i$ の m 番目の要素である. この手法では各入力変数に対して別々のパラメータを与えるよう拡張されている. 数値シミュレーションにおいては GPML Matlab Code [26] をもとに, MATLAB R2017a 上で実装している.

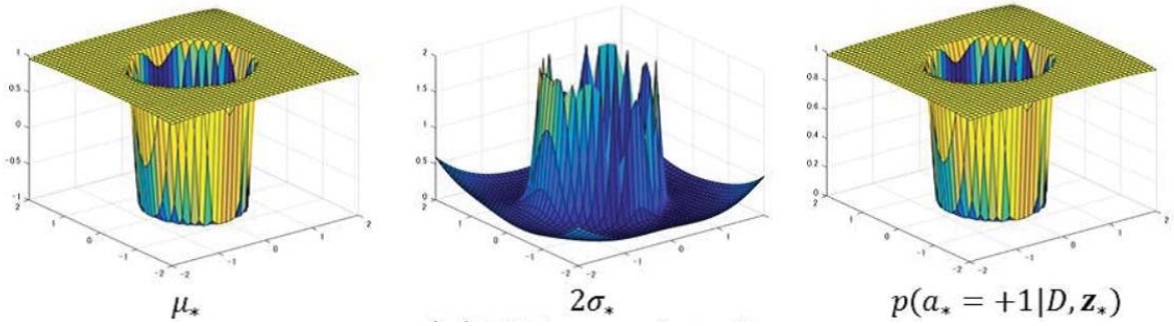
3.6.1 単位円境界の同定

期待値伝播法を用いた分類手法は 2 値分類問題において良い性能を持つことが知られており, 境界モデルの同定においても良い性能を示すことが期待できる. ここでは GPC の近似において代表的な手法であるラプラス近似法 [20][47] と期待値伝播法を単位円境界の同定問題へ適用し, 近似手法による分類器の性能を検証する. 真の境界と初期条件を Fig. 3.4 に示す.

ラプラス近似法での入力列設計手法が安定するように, 入力拘束条件に対応する点の他に境界近傍にもいくつかの点を配置している. この初期状態から逐次的に 10 点を追加した後の予測平均値, 予測標準偏差 (2σ), 予測確率モデルを Fig. 3.5 に示す. Fig. 3.5 からわかるようにラプラス近似法に比べ, 期待値伝播法は正確な分類を行っていることがわかる. 特に標準偏差予測については違いが顕著である. ラプラス近似が領域外においてある程度高い値をとっているのに対し, 期待値伝播法では境界付近以外の



(a) Laplace approximation



(b) Expectation propagation algorithm

Fig 3.5: (a) GPC with Laplace approximations, and (b) GPC with EP approximations (10 additional inputs). GPC-EP provides large score near the boundary.

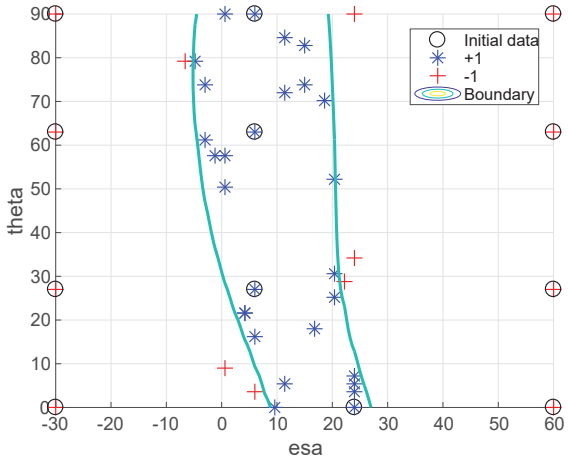
値は非常に小さく, 境界近傍のみで標準偏差が大きくなっている. この期待値伝播法の性質は分散を用いて境界を判別する提案手法において非常に有用である.

3.6.2 エンジンベンチマーク問題への適用

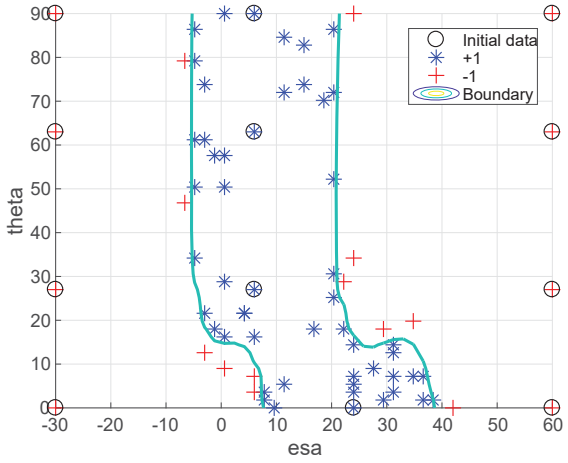
2 章で取り扱ったエンジンベンチマーク問題に対して, 提案手法を適用する. ここでは視覚化のため, 2 入力 1 出力及び 3 入力 1 出力の適用例を扱う.

A. 2 入力の適用例

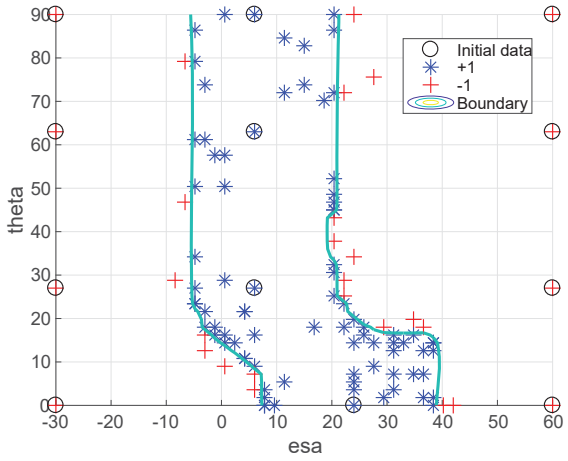
まずスロットル開度, 点火時期の 2 入力に対する運転可能領域の静的境界モデルを同定した例 Fig. 3.6 に示す. この適用例では, GPC-EP($P = 0.3$) を使用している. 追加入力が GPC により予測された境界近傍に配置され, 少ない異常運転データにより, 境界モデルを同定できていることがわかる.



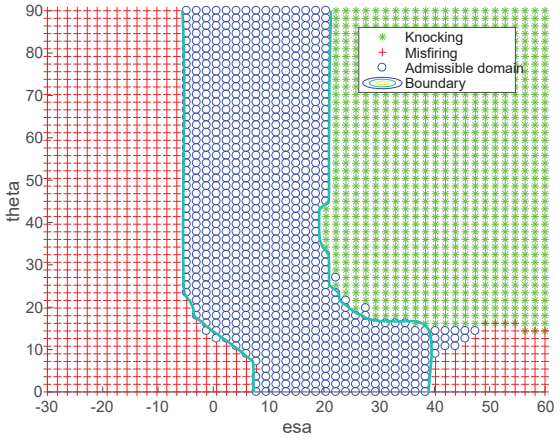
(a) Static boundary model after 30 iterations



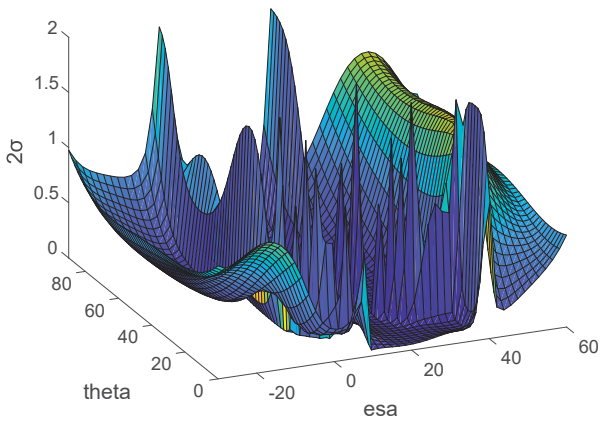
(b) Static boundary model after 60 iterations



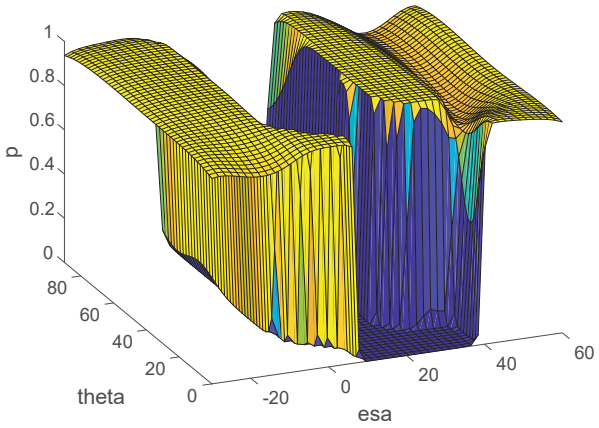
(c) Static boundary model after 100 iterations



(d) Result of verification by test data set



(e) $2\sigma_*$ after 100 iterations



(f) $p(a_* = +1|\mathcal{D}, \mathbf{z}_*)$ after 100 iterations

Fig 3.6: Example of static boundary identification with Algorithm 1 (2 input case).

B. 3 入力の適用例

この例ではエンジン回転速度, スロットル開度, 点火時期の 3 入力に対して, GPC-EP($P = 0.3$) を応用し静的境界をモデル化する. 適用結果は Fig. 3.7 に示す. 各変数は上下限値が定められているため, とり得る値の下限値を 0, 上限値を 100 としてモデル化している. また, 各入力の上下限値をとるいくつかの点を異常運転が発生する初期データとして用いている. 適用例より, 提案手法は境界から大きく逸脱することなく, 境界近傍を中心に入力を決定し, 境界モデルを生成していることがわかる. 加えて, その他の運転条件で獲得した静的境界モデルを Fig. 3.8 に示す.

また, あらかじめ取得したテストデータ 900 点を用いて正答率検証を行った. 境界モデルとテストデータとの正答率を Fig. 3.9 に示す. 追加入力が増えるほど適合率があがる傾向になるが, 追加入力がある一定数を超えると, 正答率が頭打ちになっていることがわかる. また, 異常運転領域の誤認率は非常に低く, 追加入力が増加するにつれて減少していることもわかる.

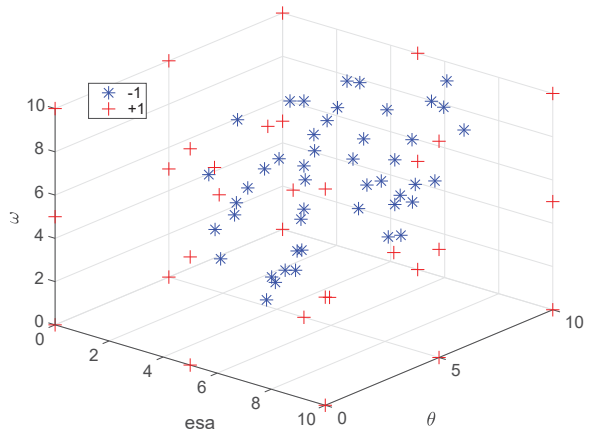
今回扱ったエンジンシミュレータにおいてノックは確率的に発生しないが, 実際のノック現象は確率的に発生する. ノック境界を p_{ref} とした場合, ノック境界上で N サイクルに m 回ノックが起こる確率は以下で与えられる.

$$P_N(m) = \left(\begin{array}{c} N \\ m \end{array} \right) p_{ref}^N (1 - p_{ref})^{(N-m)} \tag{3.24}$$

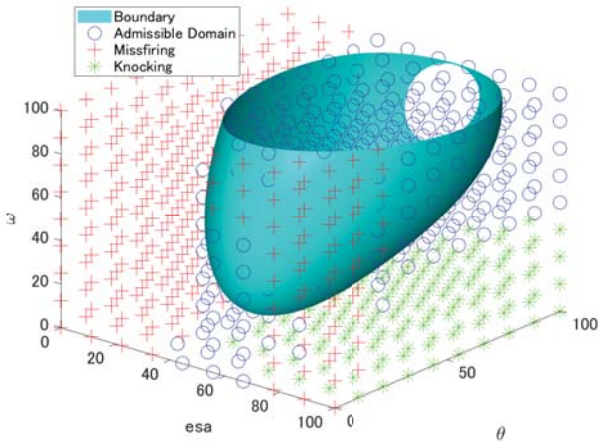
Algorithm 1 ではノック発生確率が $p = 0.5$ となる点を優先して探索するため, $P_N(m) = 0.5$ となるように N 及び m を設定し, 入力点毎に N サイクルの観測を行い, ノックが m 以上の場合には $+1$ を, m 未満の場合には -1 を出力することにより実機への適用も考慮できる. しかしながら, 確率的現象の観測においては 1 点につき N サイクルの観測を必要とするため, 適合行程に必要なコストが増加してしまうという課題があり, これは p_{ref} が小さいほど顕著になる.

3.7 おわりに

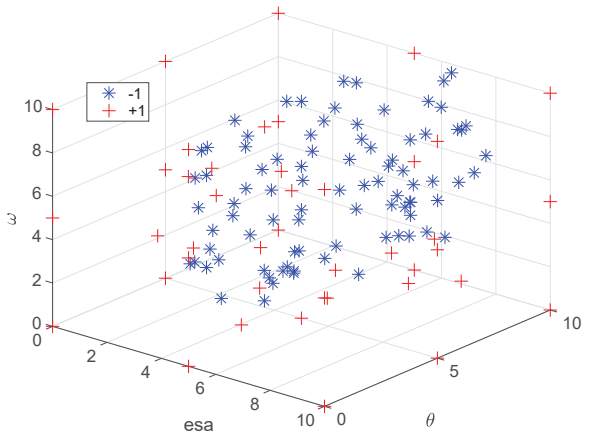
本章では精度の良い非凸な静的境界モデルを少ない適合コストで生成する為に, 確率的モデルである期待値伝播法を用いた GPC を応用したシステムの境界モデル生成法と入力列設計法を提案した. この手法は探索入力の候補を”点”で求めることができるため, 従来の探索方向のみを求める手法と比べて, 大幅な効率化が期待できる. 適用例では, 期待値伝播法の有効性を簡単な例で検証した後に, ノッキングと失火のモデルを含むエンジンベンチマーク問題に適用し, 静的境界モデルを生成した. 提案手法により効率的に境界付近の入力データが生成されることを確認した.



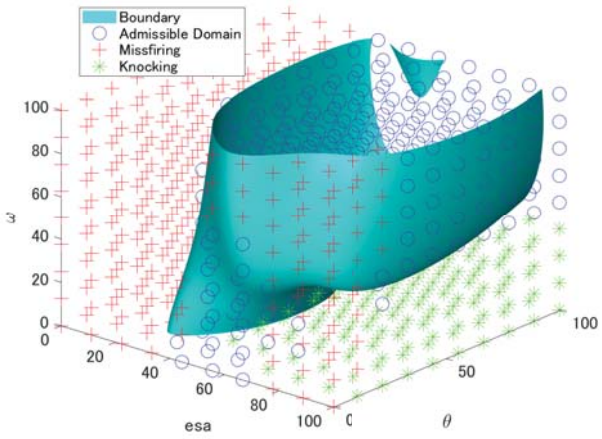
(a) Input data after 50 iterations



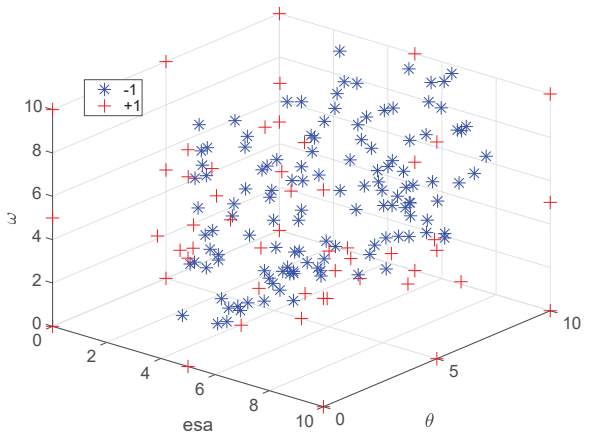
(b) Static boundary model after 50 iterations



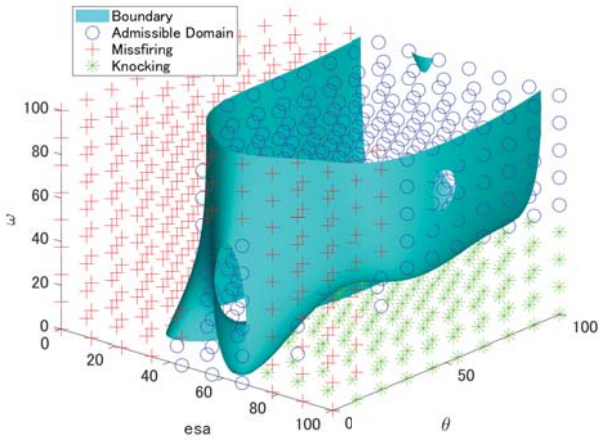
(c) Input data after 100 iterations



(d) Static boundary model after 100 iterations

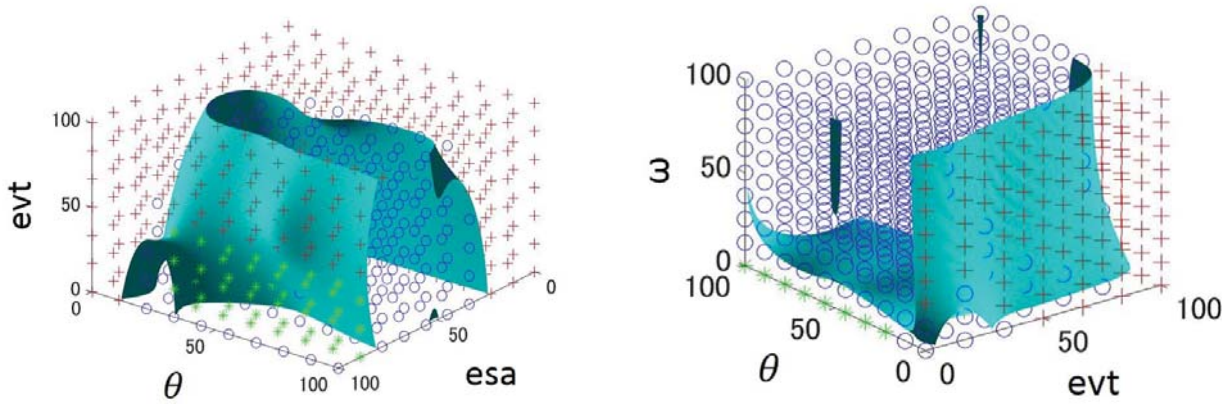


(e) Input data after 150 iterations



(f) Static boundary model after 150 iterations

Fig 3.7: Example of static boundary identification with Algorithm 1 (3 input case).



(a) Static boundary model after 150 iterations ($\omega = 1500$[rpm], $egr = 0$[mm])

(b) Static boundary model after 150 iterations ($esa = 0$[deg], $egr = 0$[mm])

Fig 3.8: Example of static boundary identification models (3 input case).

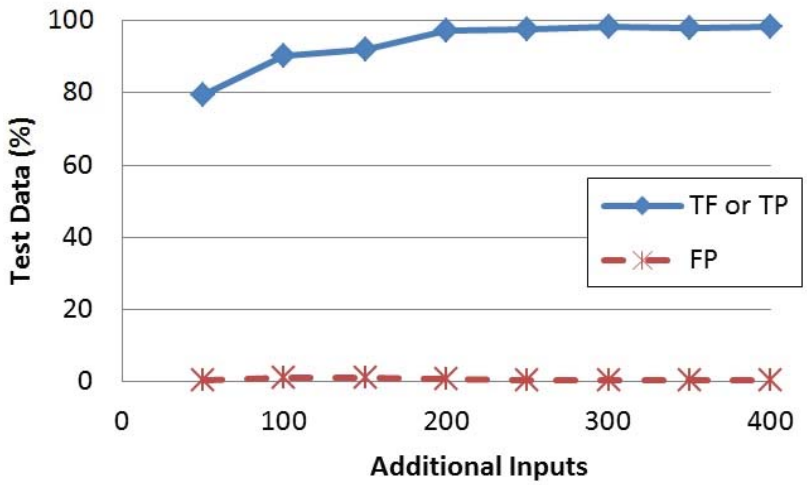


Fig 3.9: Classification accuracy between the identified boundary model and test data. The accuracy is validated with True Positive (TP) and True Negative (TN). The False Positive (FP) is 0.3-1.0 % of test data set.

第4章　動的モデルの同定

4.1　はじめに

本章では, 統計モデルにのみ注目し, 与えられたエンジンモデルの事前知識を用いず, 入出力データから直接的に制御モデルを同定することを考える. システムの構造としては, 入出力データから動的な構造を記述する, Nonlinear Auto-Regressive with eXogenous inputs (NARX) モデルを用い, その統計的モデルによる学習を試みる. ここではエンジンの制御モデルとして, NARX モデルをベースとした, 観測出力の動的モデルとトルク及び境界推定モデルを統合したシステム (2.33) に対し, そのモデルを深層学習と GP を用いて学習する. 更に, 学習データ数と予測・推定精度及び学習時間の関係を数値シミュレーションを用いて明確にすることで, モデル構築手法選択の指標を示す. 特に, 大量の学習データを持たない場合のモデル更新手法として, GP の高速近似計算手法である FITC 近似とオンライン更新アルゴリズムである逐次ガウス過程を併用することを提案している. また, 数値シミュレーションの結果を踏まえ, 深層学習の制御系応用に関して考察を行う. この検証には Society of Automotive Engineers (JSAE) と Society of Instrument and Control Engineers (SICE) から提供されているノッキング及び失火モデルを含むエンジンベンチマーク問題 [11] [12] を用いる. また, この章ではモデル構築手法にのみ注目しており, 学習データとしては一様乱数列による入力列で作成した異常運転を含むデータを用いていることに注意する.

4.2　深層学習を用いたモデリング

最初に NARX モデル (2.33) の学習アルゴリズムとして, 深層学習を使用した例について述べる. ニューラルネットワークは, 多層構造を学習することで, 高次のデータから自動で特徴量を検出できるようになった [50]-[53]. 実際に, 多層のニューラルネットワークは非常に柔軟なモデルであり, 多くの分野で実績を残している.

制御系設計における動的モデルでは, 出力が過去の入力や状態に依存する, 時系列データを扱う必要がある. このような, 時系列データに対するニューラルネットワークモデルとしては, 再帰型ニューラルネットワーク (Recurrent Neural Network, RNN) が広く知られている. しかしながら, その学習は容易ではなく, long short-term memory

(LSTM)[54], NARX-RNN モデル [55] をはじめとした, 拡張モデルが提案されている.

4.2.1 NARX-DNN モデル

ここでは, 深層ニューラルネットワーク (Deep Neural Network, DNN)[56] で, NARX モデル (2.33) を表現した NARX-DNN モデルを考える. NARX-DNN モデルにおいては, 通常の RNN で考慮が必要な潜在変数を用いないため学習し易い. また, 制御系設計においては, 1 サイクル後の予測精度が特に重要であるが, 1 サイクル後の出力にのみ注目するのであれば, 過去の入出力を制御器に記憶可能であるという仮定のもと, 単純な DNN モデルで表現, 学習を行うことができる. 即ち, 入力を $\boldsymbol{x}_k$, 出力を $\boldsymbol{y}_{k+1}$ 及び $\boldsymbol{z}_k$ とした DNN を構築すればよい. 出力が多出力となるが, ここでは簡単化のため, 出力同士の相関は考慮せず, 出力の要素を独立に学習する (Fig. 4.1).

4.2.2 エンジンベンチマーク問題への適用例

エンジンシミュレータにおいて, NARX モデルを NARX-DNN モデルで学習した例を示す. NARX-DNN モデルの場合, 外乱ベクトル $\boldsymbol{w}_k$ 及び $\boldsymbol{v}_k$ は推定しない. 自動車自体のシステムや走行環境の条件は与えられていないため, 制御問題は, エンジン回転速度 ω を一定にしたときの, トルク追従問題として与えられる. ここでは, ω =1500 [rpm], 入力列を機械的に取り得る範囲の一様乱数列とし, NARX-DNN モデルの予測・推定値と, 実際の応答を比較する. 各出力要素に対して MATLAB R2017a の Neural Network Toolbox を用いて, 異なる DNN を構成している.

学習データのサイクル数を 10000, 検証データを 300 サイクルとした場合の比較結果を Fig. 4.2 に示す. また, DNN では自動で特徴量を抽出, 低次元化を行うことができ, 大量のデータも扱えるため, モデル次数 p_o 及び p_i は十分に大きく選択すればよく, ここでは, $p_o = p_i = 10$ としている. Fig. 4.2(a)-(c) は, NARX-DNN モデルの 1 サイクル後の予測値と実際の応答を比較しており, 関数 f に相当している. Fig. 4.2(d)-(f) は, 第 1 気筒の出力信号の推定値と, 実際の信号値を比較しており, 関数 g に相当する. Fig. 4.2 からわかるように, NARX-DNN モデルは十分な量のデータがある場合には非常に良い結果を示す. 一方で, 学習データ点数が少ない場合には良い結果が得られず (Fig. 4.3), 他の学習アルゴリズムの考案が必要となる. 次の章では, 学習データが少ない場合に良い結果を示す, ガウス過程ベースのモデリング手法について説明する.

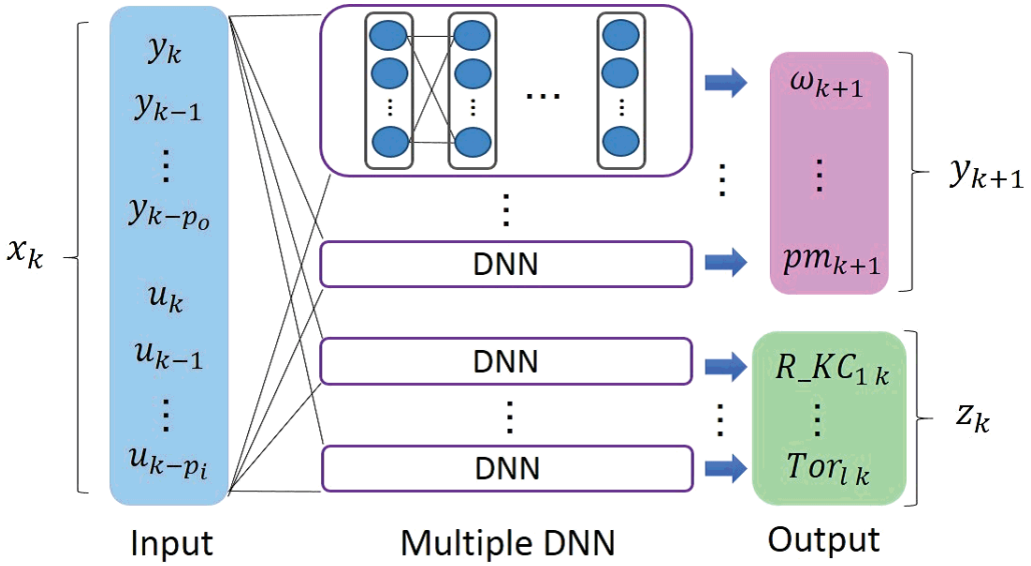


Fig 4.1: NARX-DNN model.

4.3 ガウス過程を用いたモデリング

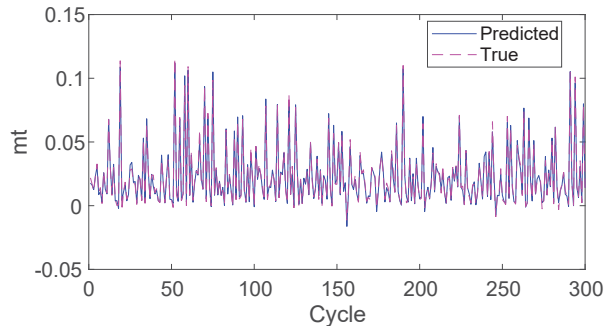
次に NARX モデルを GP を用いて表現する手法について議論する. ガウス過程の動的モデルとしてはガウス過程潜在変数モデル (Gaussian Process Latent Variable Model, GPLVM)[57] を応用したガウス過程動的モデル [58][59] や再帰型 GP[60] が提案されている. また, 状態空間表現を構築するような GP[61][62] も提案されているものの, 入力の考慮が難しく, 閉ループ系での解析が行われているため, 入力項を含む制御モデル同定への直接的な応用が困難である. また, 状態の次元数が設計パラメータとして必要など, ある程度のモデル知識も必要となる.

一方で, NARX モデルはモデル次数以外のパラメータの事前知識は必要なく, より直接的にモデルを構築できるメリットがある. しかしながら, その学習においては, 観測出力 $\boldsymbol{y}$ から入力 $\boldsymbol{x}$ に伝播する不確定性に関して陽に考慮していないことに注意する. ここでは特にモデル次数の決定法や, 計算コストの削減法について述べる.

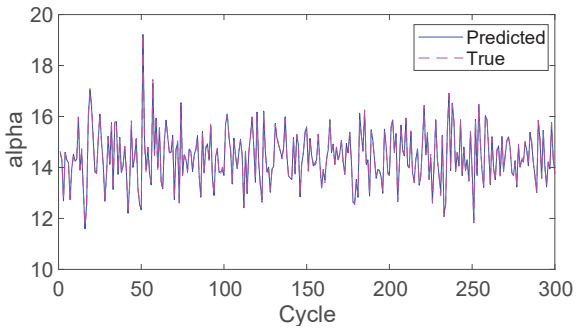
4.3.1 関連度自動決定によるモデル次数の決定

GP は主に共分散行列 $k(\boldsymbol{X}, \boldsymbol{X})$ よりその性質が定まる. 本稿では以下のような, ガウスカーネルを拡張した, 関連度自動決定 (Automated Relevance Determination, ARD) カーネルを用いる.

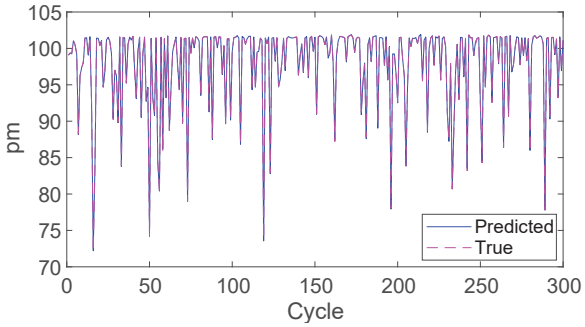
$$\begin{aligned} k(\boldsymbol{x}, \boldsymbol{x}') &= \mathrm{cov}\{f(\boldsymbol{x}), f(\boldsymbol{x}')\}, \\ &= \lambda^2 \exp(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}')^{\mathrm{T}} \Lambda^{-1} (\boldsymbol{x} - \boldsymbol{x}')). \end{aligned} \tag{4.1}$$



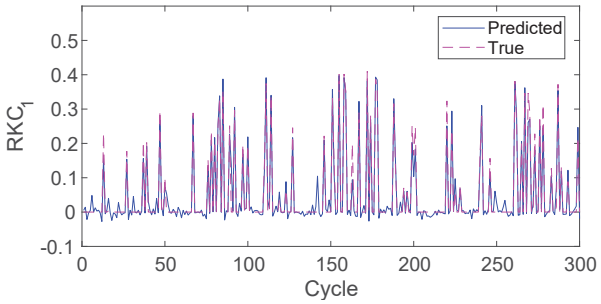
(a) Predicted m_t



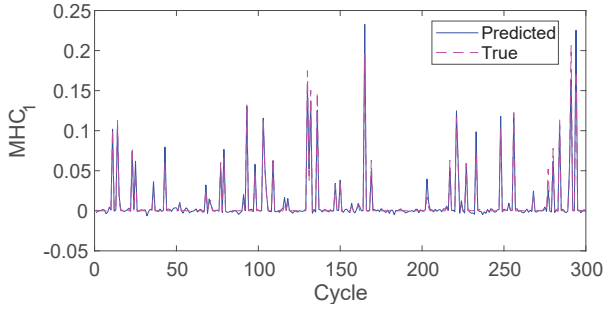
(b) Predicted α



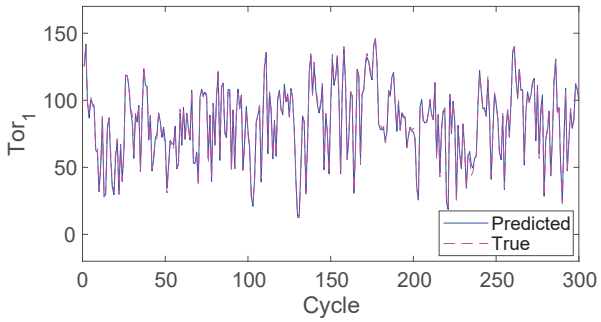
(c) Predicted pm



(d) Estimated $\boldsymbol{R\_KC}$

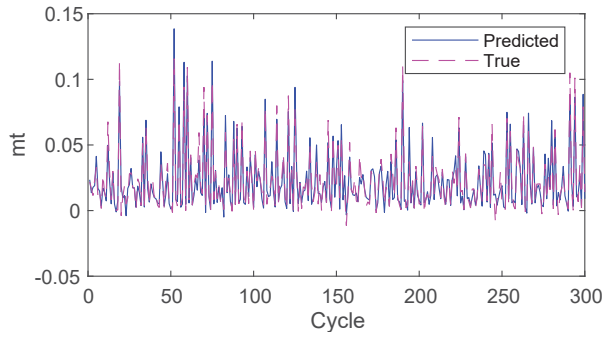


(e) Estimated $\boldsymbol{M\_HC}$

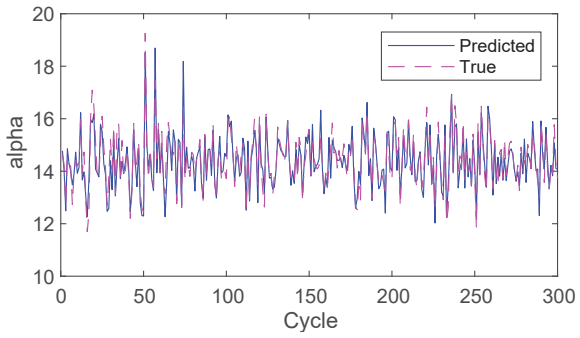


(f) Estimated $\boldsymbol{Tor}$

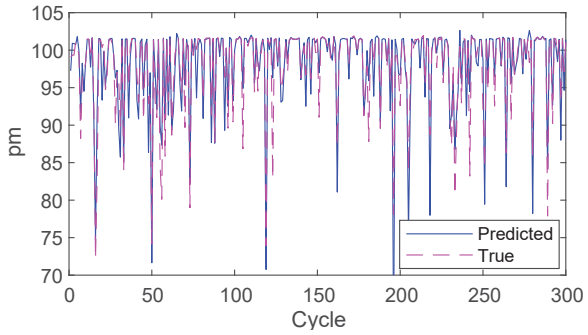
Fig 4.2: Modeling example using DNN (training data: 10000 cycles).



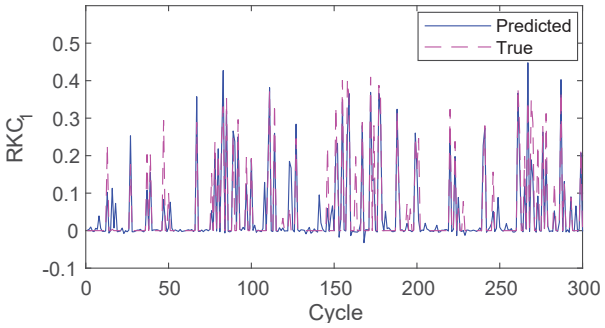
(a) Predicted m_t



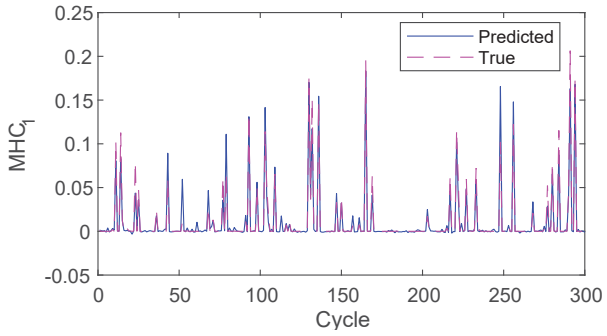
(b) Predicted α



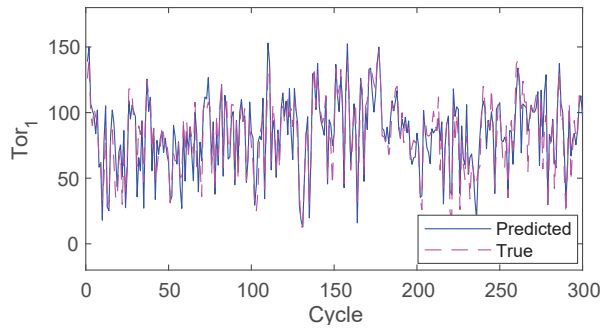
(c) Predicted pm



(d) Estimated $\boldsymbol{R\_KC}$



(e) Estimated $\boldsymbol{M\_HC}$



(f) Estimated $\boldsymbol{Tor}$

Fig 4.3: Modeling example using DNN (training data: 1000 cycles).

ここで、$\Lambda = \mathrm{diag}(l_1^2, l_2^2, \ldots, l_{n_x}^2)$ は $\boldsymbol{x}$ の各要素間の関係を特徴づけるスケールであり、λ^2 は潜在関数の分散を表すパラメータである．すべてのパラメータ $\theta = [l_1^2, l_2^2, \ldots, l_{n_x}^2, \lambda^2, \sigma^2]$ はハイパーパラメータであり, (2.5) の周辺尤度最大化より最適なパラメータを見つけることができる．最適化後, $l_1^2, l_2^2, \ldots, l_{n_x}^2$ の値によって, 入力の要素が出力の予測にどれほど寄与しているかがわかる. 即ち, 重みパラメータが十分に小さい値をもつ入力要素に関しては, 予測に寄与しないとして, 入力から削除することができる. このように, ARD の枠組みによって, 尤度ベースの次元削減とモデル次数の決定が可能になる.

4.3.2 NARX モデルへの適用

NARX-DNN モデルの場合と同様に, 入力を $\boldsymbol{x}_k$, 出力を $\boldsymbol{y}_{k+1}$ 及び $\boldsymbol{z}_k$ とし, NARX モデルの出力要素を独立に学習する. NARX-GP モデルでは, 外乱ベクトル $\boldsymbol{w}_k$ 及び $\boldsymbol{v}_k$ も推定することが可能である. また, 出力の予測値がガウス分布として表されることが大きな特徴であり, その分散値は, 予測値がどれだけ信頼できるかの指標にも用いられる. カーネル関数としてガウスカーネルや ARD を選択した場合には, 学習によるオーバーフィッティングが起こりにくく, 学習データが少ない場合には, DNN に比べ汎化性能に優れる.

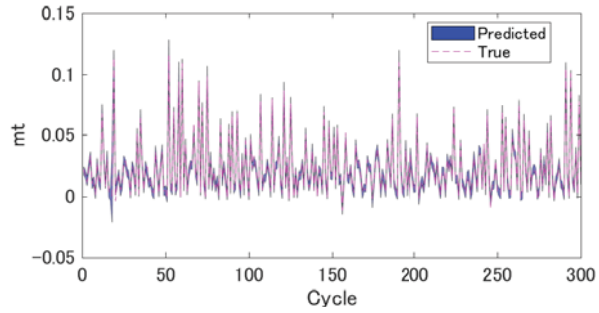
一方で, GP は DNN と比較して, 学習に必要な計算コストが大きい. そのため, 最初にモデル次数を大きくとった少量のデータを用いて, ARD によるモデル次数の決定を行う. これにより, 学習に用いる入力の次元を削減でき, 学習の計算コストを小さくすることが可能である.

また, 制御を行うにあたり, 長期予測が必要になる場合には, モーメントマッチング [63] と呼ばれる手法が提案されており, 将来の予測確率分布が解析的に計算できることも特徴である.

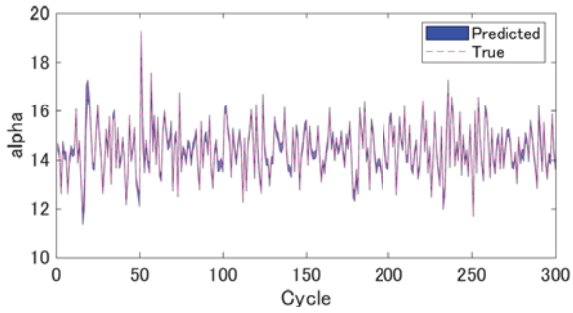
4.3.3 エンジンベンチマーク問題への適用例

エンジンシミュレータにおいて, NARX モデルを NARX-GP モデルで学習した結果を Fig. 4.4 に示す. Fig. 4.4 では, 予測・推定値として, 95%信頼区間の区間を示している. シミュレーション条件と検証データは, NARX-DNN モデルと同様のものとしている. ARD を用いて, $p_o = p_i = 5$ を選択しており, 学習データは Fig. 4.2 で用いたデータのうち, 最初の 1000 点を選択している. 学習データ点数が 10 分の 1 になったにもかかわらず, Fig. 4.4(a)-(c),(f) においては, 良い予測・推定値を示していることが分かる. 一方で, Fig. 4.4(d)-(e) においては, その予測分散が大きくなってしまっている. これは, 信頼度の高い予測ができていないと捉えることができ, より多くの学習データを用いるか, モデルの変更により改善される可能性がある. しかしながら, 実際の信号値のほ

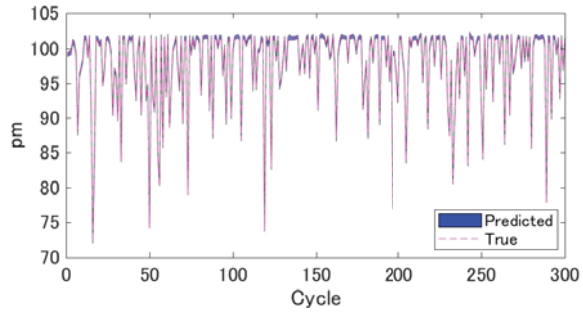
とんどが, 推定された 95%信頼区間に収まっており, GP による推定値は妥当なものであるといえる.



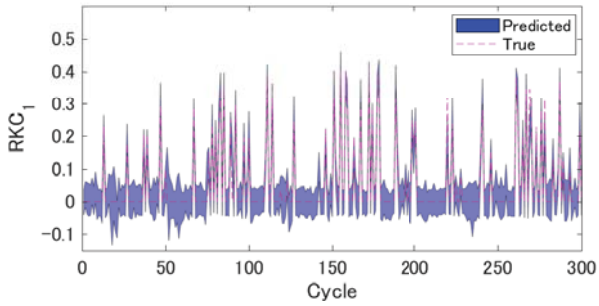
(a) Predicted m_t



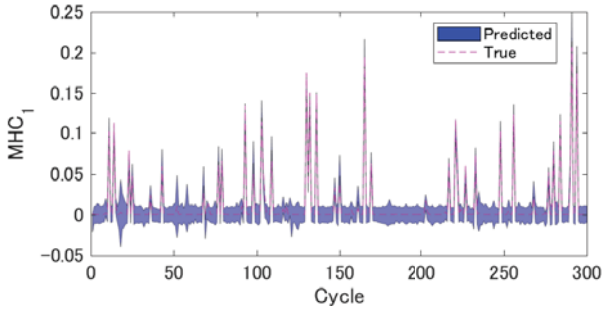
(b) Predicted α



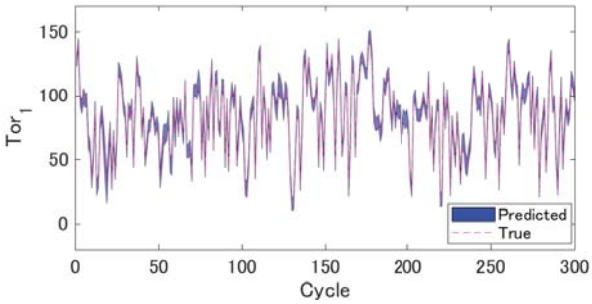
(c) Predicted pm



(d) Estimated $\boldsymbol{R\_KC}$



(e) Estimated $\boldsymbol{M\_HC}$



(f) Estimated $\boldsymbol{Tor}$

Fig 4.4: Modeling example using GP (training data: 1000 cycles).

4.3.4 ガウス過程モデルの学習高速化手法

通常の GP において, モデル学習に必要な計算コストが, 学習データ点数を N として, $O(N^3)$ のオーダーとなることが知られている. その計算コストにより, そのままでは大量のデータを扱うことができないため, 様々な学習計算の高速化手法が提案され

ている. 本章では, 特に以下の 4 つの手法に着目し, エンジンの NARX-GP モデル学習に適用した結果の比較を行う.

a) **Fully Independent Training Conditional method** (**FITC**) [31][34] : 学習データ数 N よりも少ない M 点の疑似入力を用いて GP を近似的に計算する, GP 高速化の最も標準的な手法. 学習計算のレートは $O(M^2N + M^3)$ となり, 大量のデータも扱うことが可能となる.

b) **DNN – GP with KISS – GP** (**DGP**)[41] : カーネル関数に DNN モデルを組み込むことで, 入力が低次元の場合に, $O(N + h(M))$(通常 $h(M)$ は M に関する線形関数に近い) という高速な近似計算である KISS-GP を適用する手法. M は疑似入力の数であるが, FITC のように M は N より少ない必要はない. DNN モデルの重みはカーネル関数のハイパーパラメータとして最尤法により更新され, DNN により低次元化された特徴量に KISS-GP が適用される. ここでは, https://people.orie.cornell.edu/andrew/code/ で公開されている, Python ベースの深層学習ライブラリである Keras を用いたサンプルコードをもとに実装している.

c) **Recursive Gaussian Process** (**RGP**)[65][66] : RGP は GP のオンライン回帰を目的とした, 近似手法であり, カルマンフィルターのように予測・更新ステップを通じて潜在関数 f を推定する. その詳細は 6.2 で述べる. 本稿では, ハイパーパラメータを更新しない高速計算手法を適用しており, その学習レートは $O(m^2N)$ となる. ここで m は基底ベクトルの数であり, ハイパーパラメータを求めた際の学習データの数に対応する.

d) **RGP with FITC** : RGP のカーネルとして, 大量のデータを扱える FITC で学習したカーネル (2.20) を用いる. 基底疑似入力の導入により基底ベクトルの数 m を減らすことができ, RGP の更なる高速化が期待できる. その詳細は 6.2 節で改めて述べる.

4.4 推定精度と学習時間の関係

Fig. 4.2, 4.4 では NARX-DNN モデルと NARX-GP モデルをエンジンシミュレータに適用した例を示す. 一方で, その予測・推定精度とモデル学習に必要な時間は学習データの点数に大きく依存する. この節では, エンジンシミュレータにおける, NARX-DNN モデルと NARX-GP モデルについて, データ点数と推定精度及び学習時間の比較を行う. 実際のエンジンベンチにおいては, 精度の良いデータを獲得するには, 多くの時間とコストが必要になるため学習データは少ないほど良い. 推定精度と必要となる学習データ点数の関係性を明らかにすることにより, 必要となる学習データ数の指標とすることができる. また, 学習モデルについては, 新しくデータを獲得した際のモデル更新が期待される場合が多く, 学習時間とデータ点数, あるいは推定精度との関係性を比較することは, モデル更新戦略を考える上でも重要である.

エンジンシミュレータを用いて, 第1気筒のトルク推定モデルの学習を例にとって比較する. エンジン回転速度をω =1500 [rpm], 入力列を機械的に取り得る範囲の一様乱数列とし, 検証データを300サイクルとする. 学習手法として, DNN, GP, FITC(M=500), DGP(M=200, 特徴量の次元:2), RGP(M=1000), RGP+FITC(M=500) の6つの手法を適用する. ただし, 計算コストを比較するため, 全ての手法でモデル次数を$p_o = p_i = 5$とし, 並列計算は行わず, シングルコアでの比較を行っている. また, RGPのハイパーパラメータは学習データ数N=1000のときにGPで求めたハイパーパラメータを用いており, その学習に必要なコストは含めていないことに注意する. 同様に, RGP+FITCのハイパーパラメータはN=1000のときにFITC(M=500) で求めたハイパーパラメータを用いており, その学習計算コストは含めない. 学習データ点数を50点から5000点まで変化させた場合の結果をFig. 4.5-4.7に示す. トルク推定モデルの学習結果は以下のようにまとめることができる.

- 1000点付近までの学習データ点数が少ない場合には, GPがDNNに比べ良い結果を示す. DNNは学習データに対しては非常に良くフィットするものの, 学習データが少ない場合には過学習が起こり汎化性能が落ちるため, 未知出力の推定精度は悪くなる. 一方で, GPは過学習が起こりにくいという性質から, データ数が少ない場合においても良い推定精度を示す.

- 学習点数が多くなると, GPの計算コストは飛躍的に増加するが, GPの高速計算手法ではその増加量を抑えることができる. 特に推定精度ではFITCが, 計算コストにおいてはRGPが良い結果を示す. FITCベースのRGPは更に高速な計算が可能であり, そのコストはDNNよりも少ない. しかしながら, ハイパーパラメータの更新を行わないため, 学習データに対して, 推定精度は頭打ちになってしまう.

- DGPは高速化を目的としたKISS-GPによる近似精度の問題と, 学習結果がハイパーパラメータの初期値に大きく依存してしまうことから, 良い推定精度を獲得することができなかった. 疑似入力mを増加させ, DNNとGPの初期値学習を十分行うことで精度は向上するが, その分学習時間は必要になってしまう.

- 学習データ点数が5000点以上となると, DNNが推定精度・学習時間の両面において, RGP+FITCを除くGPの高速化手法と比較して良い結果を示す. 学習データが多くなるにつれて, GPの推定精度の向上率が鈍化してくるのに対し, DNNベースの手法では推定精度の向上が見られることが分かる. ここでは示していないものの, 学習データが10000点以上になると, 推定精度はGPよりも良い結果となる.

以上より, まずトルクの学習データ点数が多く, 確率的予測が必要ない場合には NARX-DNN モデルがふさわしいといえる. そして, 学習点数が少ない場合には NARX-GP モデルを選択し, 学習データ点数が多く, かつ確率的予測が必要な場合には NARX-GP モデルの高速近似手法を選択することがふさわしい. また, モデル更新手法としては, 定期的な FITC によるハイパーパラメータ更新と, RGP+FITC を用いたオンライン更新を組み合わせる手法が効果的であると考えられる. RGP+FITC を用いて学習データを 1 点獲得する毎にモデルを更新することを考えると, 1 回あたりの平均計算時間は 5.0[ms] であり, 他のモデルを用いたオンライン更新に比べ非常に高速であることが分かる.

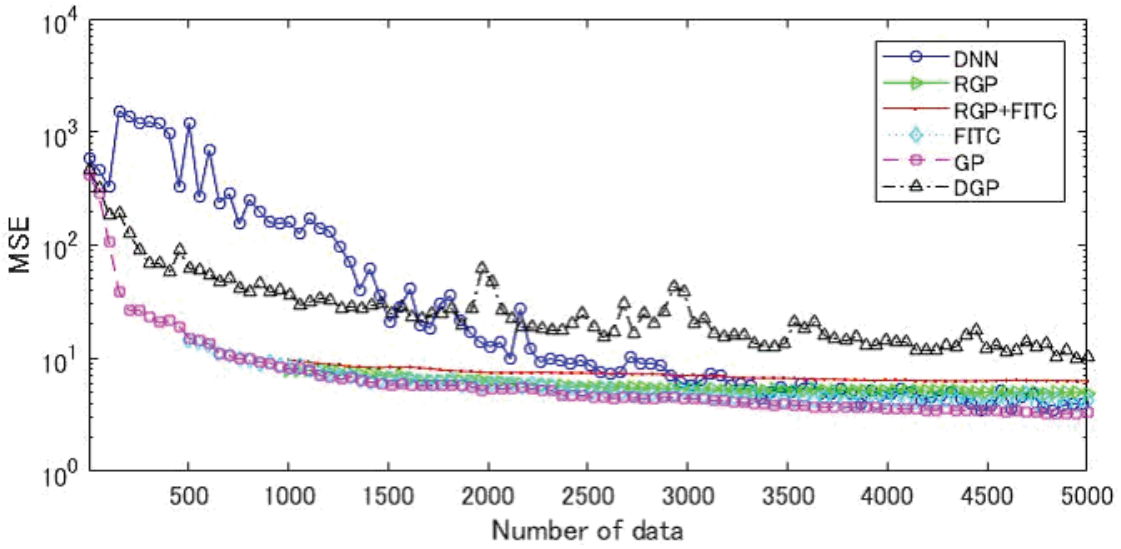


Fig 4.5: Relationship between the number of training data and estimation accuracy.

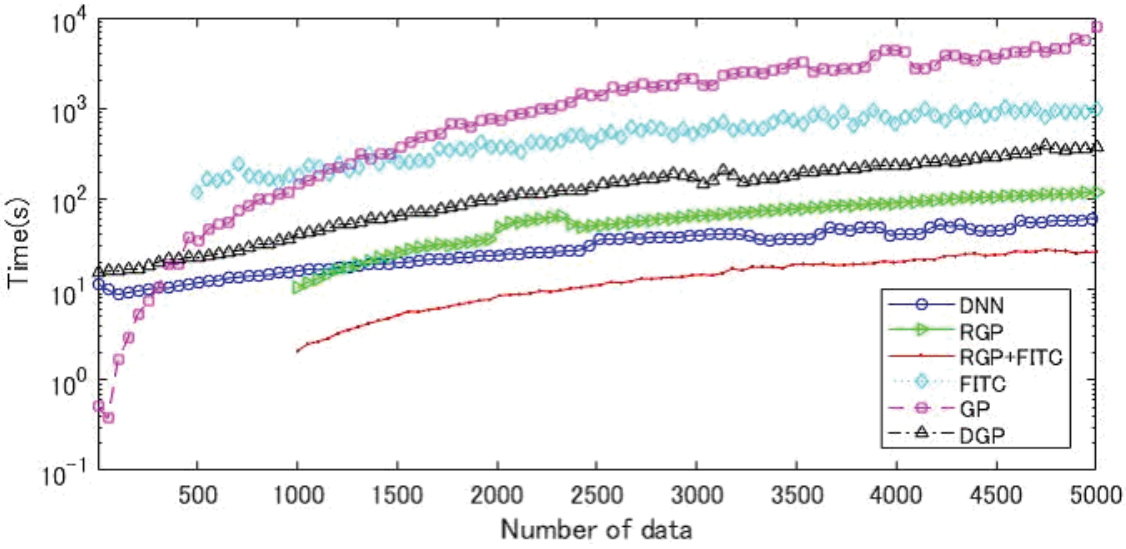


Fig 4.6: Relationship between the number of training data and learning time.

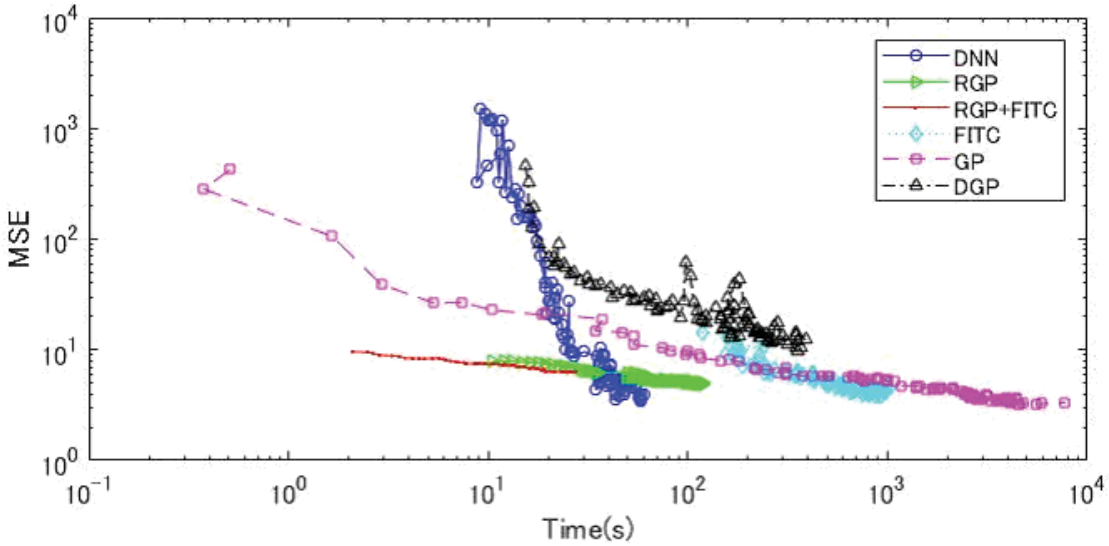
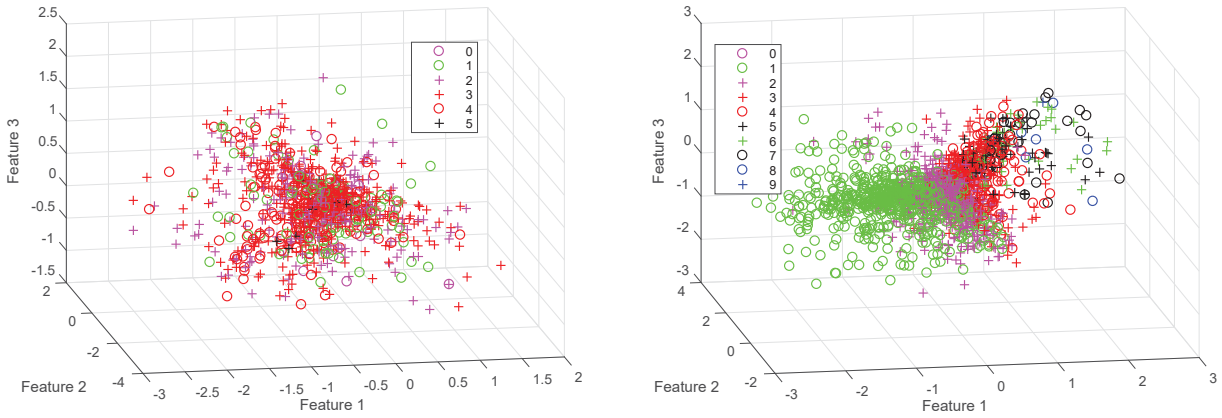


Fig 4.7: Relationship between learning time and estimation accuracy.



(a) Classification of ***Tor*** value (6 different classes) (b) Classification of m_t value (10 different classes)

Fig 4.8: Extracted features from a DBN on engine simulator dataset.

4.5 深層学習の応用に関する考察

4.5.1 エンジンモデルに対する事前学習の有効性

前節の結果より, DNN の低次元化と KISS-GP を組み合わせた DGP において良好な結果を得ることができず, ハイパーパラメータの初期値が上手く設定できていない可能性が示唆された. これは DNN における事前の教師無し学習が有効ではないことを示している. そこで事前学習のみに注目し, 深層信念ネットワーク (Deep Belief Network, DBN)[50][51] を用いて 3 次元空間に低次元化した結果を Fig. 4.8 に示す. ここでは MATLAB R2017a 上で動作する Toolbox[67] を用い, Fig. 4.2 で用いた学習データ 10000 点により事前学習を行っている. また, 対応する出力値はその大小により事前にラベル付けをしている. 結果から分かるようにトルク値 ***Tor*** は事前学習で得られた特徴量との関係が明確でない可能性がある. 他の出力に対しても比較した結果, 吸入空気量 m_t に対してのみその傾向を良く示していることが分かった. 事前学習が有効でない理由を以下のように考察する.

1. 事前学習に用いるネットワークが適切でない
2. 学習データの特徴量は適切に示しているが, 特徴と出力の関係が明確でない
3. 学習データはすでに十分低次元化された情報である
4. 開ループ同定をしているため, 制御系の特性が現われにくい

特に今回扱うエンジンシミュレータではセンサは経験的にエンジン制御に必要だと思われる最低限のものを予め選択されており, 学習に用いるセンサデータ自体がエンジ

ンの挙動を表現するために十分低次元なものとなっている可能性がある. また, 周波数応答など制御系の特徴として考えられるものを陽に扱えるように, 学習データへ前処理を適用することも必要になると考える.

ここでは事前学習について述べたが, 前節の DNN の結果から分かるように DNN の持つモデル表現の柔軟性は制御モデル構築に有効であり, その利点は制御系設計にも活用できると考えている.

4.5.2 非線形補助変数を用いた統計的線形近似

次に深層学習の柔軟性を応用した NARX モデル以外のアプローチの例として, DNN を応用した統計的線形近似手法について述べる. この手法ではデータを用いた統計的な線形化 [68] に DNN を応用することでその精度を高めることを提案している. 線形化を考える理由は線形動的システムの制御に関しては多くの完成した制御理論の手法が提案されていることと, その制御計算負荷が非線形モデルと比較すると非常に少なくなることから, ECU の制御計算負荷を軽減できる可能性があるためである. 特にノックモデルといった確率モデルを含むエンジンの制御を考える上では非線形性に起因する制御入力計算コストは大きく, 線形化の恩恵は大きい.

最初に補助変数を用いた統計的線形近似手法の概要を説明する. 次のような時不変離散非線形システムを考える.

$$\begin{aligned} \mathbf{x}(k+1) &= f(\mathbf{x}(k), \mathbf{u}(k)), \\ \mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k). \end{aligned} \tag{4.2}$$

ベクトル $\mathbf{x}(k) \in \mathbb{R}^n$, $\mathbf{u}(k) \in \mathbb{R}^m$, $\mathbf{y}(k) \in \mathbb{R}^l$ は k ステップでの状態ベクトル, 入力ベクトル, 出力ベクトルであり, $f : \mathbb{R}^n \to \mathbb{R}^n$ は非線形写像, $\mathbf{C} \in \mathbb{R}^{l \times n}$ は出力行列である. システムの非線形要素と関係する補助変数 $\xi \in \mathbb{R}^{l_a}$ を導入し, 出力ベクトルと補助変数を組み合わせた拡大出力ベクトルを以下で定義する.

$$\tilde{\mathbf{y}}(k) = \begin{bmatrix} \mathbf{y}(k) \\ \xi(k) \end{bmatrix} \in \mathbb{R}^{l_z}. \tag{4.3}$$

また, 学習入出力行列を以下で与える.

$$\begin{aligned} D_{\mathbf{u}} &= \begin{bmatrix} \mathbf{u}(0) & \mathbf{u}(1) & \cdots & \mathbf{u}(N) \end{bmatrix}, \\ D_{\tilde{\mathbf{y}}} &= \begin{bmatrix} \tilde{\mathbf{y}}(0) & \tilde{\mathbf{y}}(1) & \cdots & \tilde{\mathbf{y}}(N) \end{bmatrix}. \end{aligned} \tag{4.4}$$

統計的線形近では得られたデータに対して特異値分解を応用した次元削減手法を適用する. 制御理論の分野では入出力データから動的システムを同定する手法である部分

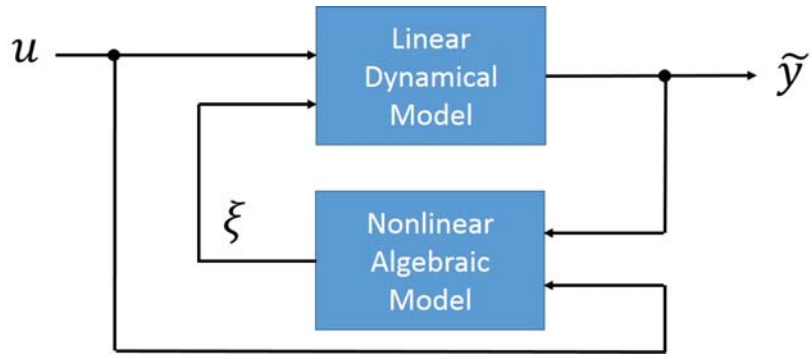


Fig 4.9: Statistical linear approximation using nonlinear auxiliary variables.

空間同定法 [69] がよく知られている. 例えば N4SID アルゴリズム [70] は次のような確率システムとして同定が可能である.

$$\begin{aligned} \mathbf{z}(k+1) &= \tilde{\mathbf{A}}\mathbf{z}(k) + \tilde{\mathbf{B}}\mathbf{u}(k) + \mathbf{K}\mathbf{e}(k), \\ \tilde{\mathbf{y}}(k) &= \tilde{\mathbf{C}}\mathbf{z}(k) + \mathbf{e}(k), \end{aligned} \tag{4.5}$$

ここで, $\mathbf{z} \in \mathbb{R}^{n_z}$ $(n_z \leq l_z)$ は潜在ベクトルである. $\tilde{\mathbf{A}} \in \mathbb{R}^{n_z \times n_z}$ は潜在空間でのシステム行列であり, $\tilde{\mathbf{B}} \in \mathbb{R}^{n_z \times m}$ は入力行列, $\tilde{\mathbf{C}} \in \mathbb{R}^{l_z \times n_z}$ は出力行列である. $\mathbf{K} \in \mathbb{R}^{n_z \times l_z}$ は外乱行列であり, 外乱 $\mathbf{e}(k) \in \mathbb{R}^{l_z}$ はガウスノイズ $p(\mathbf{e}(k)) = \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_e)$ であると仮定する. 元となったシステムは非線形であるにも関わらず, 式 (4.5) は潜在空間での近似線形動的システムとして表現される. つまり, 元のシステムはシステムの振る舞いが線形になる, より高次の潜在空間に作り直される (Fig. 4.9).

以上のような統計的モデルに対して, 深層学習の応用を考える. 即ち, 獲得した入出力データからの動的システム同定においてモデルの事前知識がないブラックボックスモデリングを想定し, 補助変数の選択手法として DNN を応用する. 潜在空間でのノイズフリーな離散線形動的システムとして以下を考える.

$$\begin{aligned} \mathbf{z}(k+1) &= \tilde{\mathbf{A}}\mathbf{z}(k) + \tilde{\mathbf{B}}\mathbf{u}(k), \\ \tilde{\mathbf{y}}(k) &= \tilde{\mathbf{C}}\mathbf{z}(k). \end{aligned} \tag{4.6}$$

ここで, $\tilde{\mathbf{C}}$ は列フルランクと仮定する. $\Delta\mathbf{y}(k) = \tilde{\mathbf{y}}(k+1) - \tilde{\mathbf{y}}(k)$, $\tilde{\mathbf{C}}^{\#} \in \mathbf{R}^{l_z \times n_z}$ を疑似逆行列 $\tilde{\mathbf{C}}^{\#} = (\tilde{\mathbf{C}}^T\tilde{\mathbf{C}})^{-1}\tilde{\mathbf{C}}^T$ と定義すると, もとの空間上での出力のダイナミクスは以下で表すことができる.

$$\Delta\mathbf{y}(k) = (\tilde{\mathbf{C}}\tilde{\mathbf{A}}\tilde{\mathbf{C}}^{\#} - \mathbf{I})\tilde{\mathbf{y}}(k) + \tilde{\mathbf{C}}\tilde{\mathbf{B}}\mathbf{u}(k) \tag{4.7}$$

即ち, システムが式 (4.6) で表現できる場合, 拡大出力ベクトルに関して, $\Delta\mathbf{y}(k)$ は $\tilde{\mathbf{y}}(k)$ と $\tilde{\mathbf{u}}(k)$ の線形和で表現できる必要がある. そこで, Fig. 4.10 の構造を持つニューラル

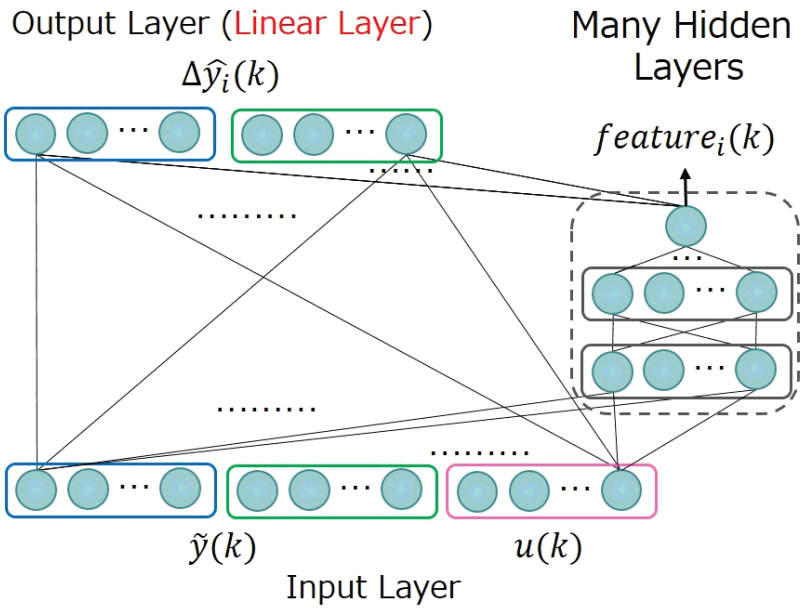


Fig 4.10: Illustration of the architecture of auxiliary variable training. The feature of last hidden layer generates a candidate for an auxiliary variable by training the neural network.

ネットワークを繰り返し用いて, 学習データから補助変数の候補を獲得することを考える.

このニューラルネットワークは出力層が線形層となっており, $\Delta\mathbf{y}(k)$ との誤差を小さくするように隠れ層の学習を行う. 学習されたネットワークについて, 隠れ層の最終層の特徴量は補助変数の候補となる. ここで, 以下の学習データを定義し, 補助変数選択のアルゴリズムを Algorithm 2 にまとめる. ただし, 学習データ D は以下で定義される.

$$D = \left[\begin{array}{ccc} D_{\mathbf{u}}^T & D_{\tilde{\mathbf{y}}}^T & D_{\Delta\mathbf{y}}^T \end{array} \right]^T \tag{4.8}$$

$$D_{\Delta\mathbf{y}} = \left[\begin{array}{ccc} \Delta\mathbf{y}(0) & \Delta\mathbf{y}(1) & \cdots \Delta\mathbf{y}(N) \end{array} \right] \tag{4.9}$$

もし学習データ D に対し $\mathrm{rank}(D) = \mathrm{rank}([D_{\mathbf{u}}^T \ D_{\tilde{\mathbf{y}}}^T]^T)$ を満たす補助変数が発見できれば, 式 (4.6) のシステム表現が可能となるが, この手法は補助変数の発見を保証するものではないことに注意する. 統計的線形化においては, 変分線形回帰 [71] を応用することで, モデルの不確かさの表現も可能である. 例えば Fig. 4.11 では, $\Delta\mathbf{z}(k) = \mathbf{z}(k+1) - \mathbf{z}(k)$

Algorithm 2 Auxiliary variable selection using DNN

1: **初期化**: $\tilde{\mathbf{y}}_0(k) = \mathbf{y}(k) \in \mathbb{R}^l$ とし, 最大繰り返し数 i_{max} を設定する.
2: 学習データ $D_{\mathbf{u}}$, $D_{\tilde{\mathbf{y}}0}$, D_0 を獲得する.
3: 学習データに関して特異値分解による次元解析を行い, $\mathrm{rank}(D_0) = \mathrm{rank}([D_{\mathbf{u}}^T \ D_{\tilde{\mathbf{y}}0}^T]^T)$ ならば終了
4: **for** $i = 1 : i_{max}$
5: ニューラルネットワーク (Fig. 4.10) の学習を行う
6: 隠れ層最終層の特徴量を $\mathrm{feature}_i(k)$ とし, 拡大出力を以下のように設定する.

$$\tilde{\mathbf{y}}_i(k) = \begin{bmatrix} \tilde{\mathbf{y}}_{i-1}(k) \\ \mathrm{feature}_i(k) \end{bmatrix} \in \mathbb{R}^{l_{zi}=l+i} \tag{4.10}$$

7: 学習データ $D_{\tilde{\mathbf{y}}i}$, D_i を獲得する.
8: 学習データに関して特異値分解による次元解析を行い, $\mathrm{rank}(D_i) = \mathrm{rank}([D_{\mathbf{u}}^T \ D_{\tilde{\mathbf{y}}i}^T]^T)$ ならば終了
9: **end**

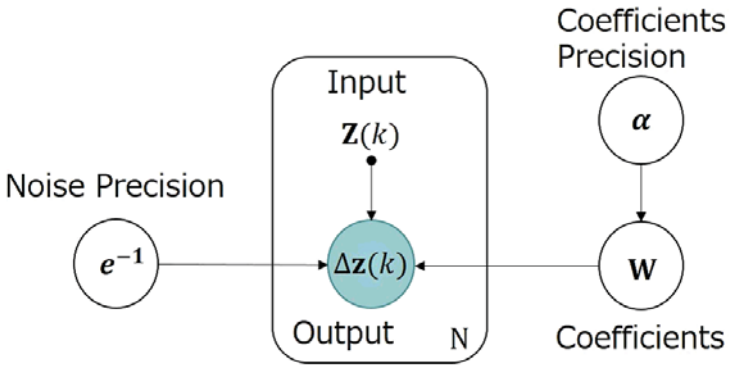


Fig 4.11: Graphical model representation of a Bayesian linear regression.

と定義した次の回帰問題を考えている.

$$\begin{aligned} \Delta\mathbf{z}(k) &= \mathbf{A}_V\mathbf{z}(k) + \mathbf{B}_V\mathbf{u}(k) + \mathbf{e} \\ &= [\mathbf{A}_V \ \mathbf{B}_V] \begin{bmatrix} \mathbf{z}(k) \\ \mathbf{u}(k) \end{bmatrix} + \mathbf{e} \\ &= \mathbf{W}\mathbf{Z}(k) + \mathbf{e}. \end{aligned} \tag{4.11}$$

$\mathbf{A}_V \in \mathbb{R}^{n_z \times n_z}$ は $(\tilde{\mathbf{A}} - \mathbf{I})$ に相当する行列, $\mathbf{B}_V \in \mathbb{R}^{n_z \times m}$ は入力行列に相当する. また, ベクトル $\mathbf{Z}(k) = [\mathbf{z}(k)^T \ \mathbf{u}(k)^T]^T \in \mathbb{R}^{n_z+m}$ は拡大された入力であり, 行列 $\mathbf{W} = [\mathbf{A}_V \ \mathbf{B}_V] \in \mathbb{R}^{n_z \times n_z+m}$ は線形回帰問題での重み行列に相当する.

Fig. 4.12 で示される振り子モデルを用いて数値シミュレーションを行い, 提案手法

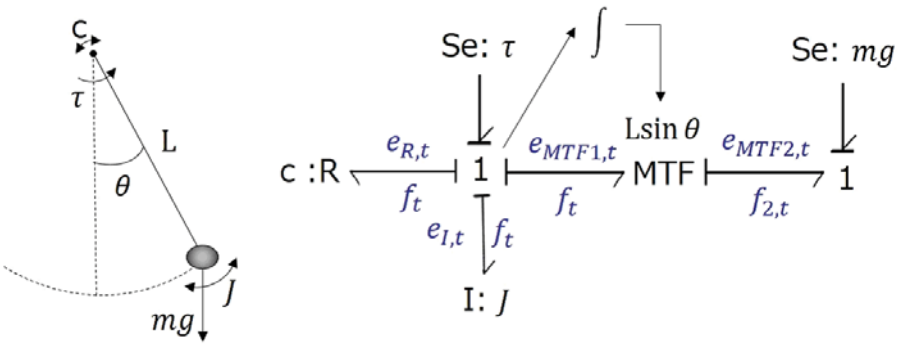


Fig 4.12: A pendulum system and the Bond Graph associated with the system.

の有効性を検証する. ここでは, 振り子モデルを以下の離散時間非線形システムとして近似している.

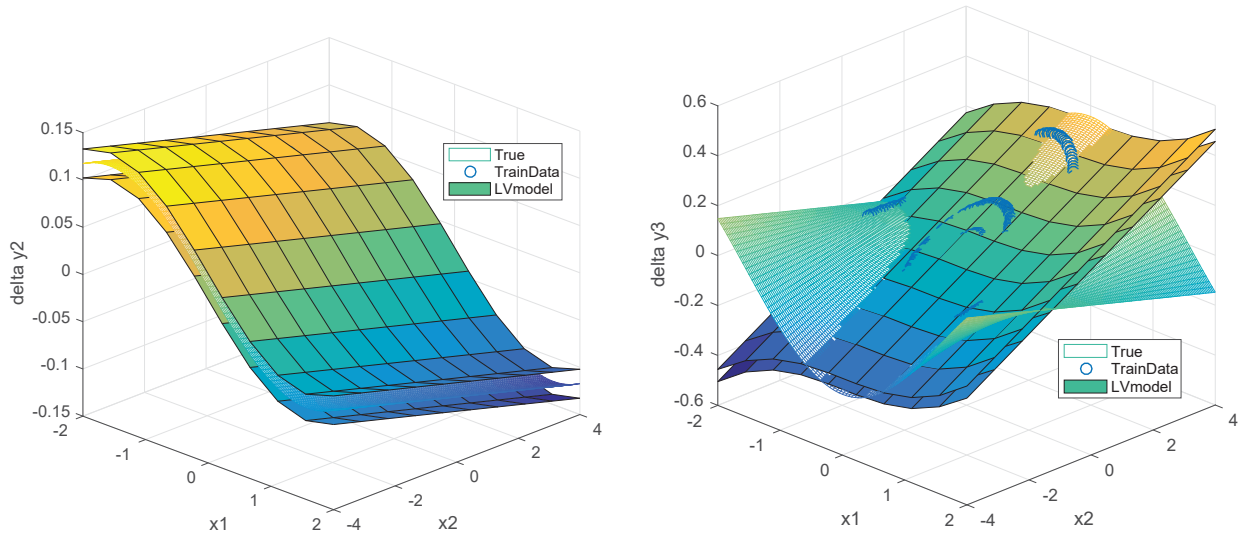
$$\begin{aligned}\begin{bmatrix}\theta(k+1)\\ \dot{\theta}(k+1)\end{bmatrix} &= \begin{bmatrix}\theta(k)\\ \dot{\theta}(k)\end{bmatrix} + \begin{bmatrix}\dot{\theta}(k)\\ -\frac{c}{J} - \frac{mLg}{J}\sin\theta(k)\end{bmatrix}\Delta t + \begin{bmatrix}0\\ \frac{1}{J}\end{bmatrix}\tau,\\ \mathbf{y}(k) &= \mathbf{x}(k) = [\theta(k),\ \dot{\theta}(k)]^T\end{aligned} \tag{4.12}$$

Algorithm 2 を適用した結果を Fig. 4.13 に示す. この例では 5 個の潜在変数を用いており, $\dot{\theta}(k)$ とその差分に対する真のダイナミクス (4.12) を学習データを用いて近似できているかを検証している. また, 視覚化のために入力を 0 とした場合を示している. 変分線形回帰を用いて各パラメータの近似事後分布を求めており, 信頼区間として 2σ の範囲を表示している. 学習データの周辺ではダイナミクスが良く近似できている一方で, 学習データがない箇所に関しては差異が大きなモデルになっていることが分かる.

しかしながら, 本節で扱ったモデルはエンジンシミュレータのような複雑な対象に対して精度良く近似することができていない. 提案手法は非線形変換による制御空間の拡張を考える点ではめ込み法 [72] と関係があると考えられるため, その関係性を明らかにすると共に, 複雑な非線形システムへの適用できるよう拡張していく必要がある.

4.6 おわりに

本章では, JSAE と SICE から提供されているエンジンベンチマーク問題を対象に, 学習データのみを用いた, 統計的制御用エンジンモデルの構築手法を提案した. 制御モデルとしては NARX モデルを考え, モデル学習のために DNN と GP ベースの手法を適用した. GP ベースのモデルに対しては, その高速化手法も扱い, 特に RGP と FITC を



(a) One step predictons of $\Delta \mathbf{y}_2$

(b) One step predictons of weighted difference value of $\Delta \mathbf{y}_2$

Fig 4.13: Comparison between true response value and one step predicted value of output with uncertainty models. True response can be approximated well near the training data.

組み合わせた手法を提案した. その後, エンジンシミュレータに対して, DNN, GP, 高速化 GP による NARX モデル学習手法を適用し, それらの学習データ点数と推定精度及び学習時間の関係を検証した. また, シミュレーションの結果を踏まえ, 深層学習の制御系応用に関して考察を行った.

学習モデル選定の指標として, 学習データ点数が少ない場合には, NARX-GP モデルが有効であることが分かった. また, 学習データ点数が多く確率的予測が必要ない場合には NARX-DNN モデルが, 学習データ点数が多くかつ確率的予測が必要な場合には NARX-GP モデルの高速近似手法を選択することがふさわしいと結論付けた. 更に, モデル更新手法としては, FITC によるハイパーパラメータ更新と, RGP+FITC を用いたオンライン更新を組み合わせたモデル学習手法が効果的であると考えられる. 深層学習の応用としては, 補助変数に DNN を応用した統計的線形近似手法を提案した. このモデルを用いた確率的モデル予想制御の有効性についても議論されている.

第5章　ガウス過程を用いた動的実験計画法

5.1　はじめに

本章では, ガウス過程を用いた動的実験計画法 (Dynamical Design of Experiment, DDoE) を提案する. 4 章においては, データ点数が少ない場合には NARX-GP 及びその近似手法が良い予測・推定性能を示すことを述べた. 一方で, 3 章で説明したように自動車エンジンのモデル同定では安全性を担保した上での効率的な入力列設計手法が求められる. 特に GP モデルの性能は学習データ点によって決定されるため, その性能は入力列設計法に従ったものとなる. ここでは, NARX-GP モデルでエンジンの動的制御モデルを表現し, 異常運転を発生を抑えた入力列設計手法について述べる. 3 章との違いは境界の動的モデルを扱う点と, 境界モデルだけではなく制御モデル自体を扱うことにある. 具体的には, 静的境界モデルを入力制約とした上で, 予測ホライゾン内の予測分散の重み付き和が最大となるような入力を計算しつつ, NARX-GP モデルをオンラインで更新していく手法となっている. その有効性は数値シミュレーションにより示される.

5.2　関連研究

実験計画法は獲得できるシステムの情報を最大とするような入力列設計を目的としているが, 本章では動的システムを対象とする. その手法はモデルフリーかモデルベースか, またオフラインかオンラインかで分類することができる [25][73][74]. ここでは関連研究についてカテゴリ毎に概要を説明する.

5.2.1　モデルフリー動的実験計画法 (オフライン)

モデルフリー DDoE では, 学習データ集合がシステム及び境界の同定において有効であろう指標を満たすように入力列を設計する [75]. 例えば以下の指標で学習データ集合を評価する.

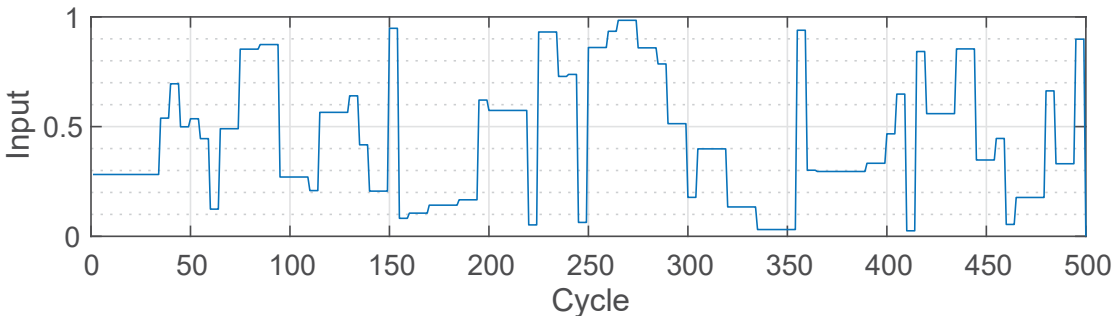


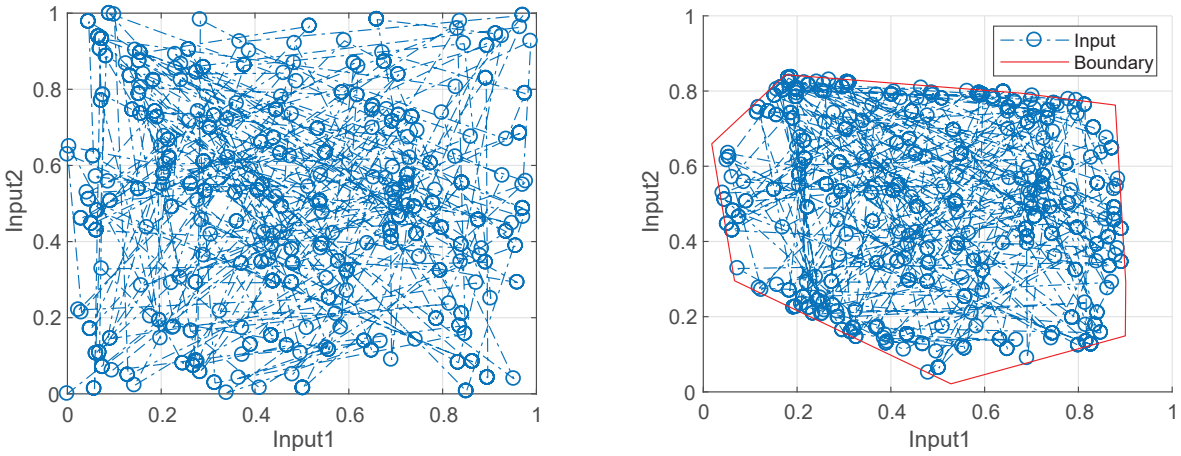
Fig 5.1: Example of APRBS signal.

- 周波数 : 入力周波数が十分な幅を持ち, 更に各入力の周波数の組み合わせが十分であるほど良いデータ集合である.
- 振幅 : 非線形システムの場合には周波数と振幅の組み合わせが十分であるほど良いデータ集合となる.
- 学習データ数 : 入力の次元に従い, 必要となる入力の数は指数的に増加するため, 獲得した学習データが多いほど良いデータ集合であるといえる.
- 入力空間の分布 : 境界内の入力空間において, 入力の分布が偏りなく一様であり, 入力空間を満たしているほど良いデータ集合である.

特に自動車エンジンに対しては Amplitude modulated Pseudo Random Binary Sequence (APRBS) 信号 [76](Fig. 5.1) をベースにしたものが広く使用されている [73][74]. 一般には入力境界を凸包として同定し, 入力空間の運転可能領域を満たすように入力の補正を行う (Fig. 5.2). モデルフリー DDoE は全ての非線形システムに適用できるため広く用いられているが, 入力の次元が多くなった場合, 必要となる学習データ点数が爆発的に増加してしまうという問題がある.

5.2.2 モデルフリー動的実験計画法 (オンライン)

モデルフリー DDoE による学習においての問題の 1 つは, 境界の扱いである. 通常オフラインでのモデルフリー DDoE においては凸包を仮定した静的境界をもとに入力列を設計するが, 3 章でも述べたように一般には静的境界は凸包にはならず, また動的境界の同定に必ずしも静的境界が有用かは分からない. 文献 [25] では, 静的境界近傍から周期的な入力値を与えることで, より広い入力空間で学習データを獲得する手法が提案されている (Fig. 5.3). この手法ではオンラインで異常運転を検知しながら入力列設計を行うため, 静的境界のみならず動的境界近傍のデータを獲得することが可能と



(a) Input space with APRBS　　(b) Feasible input space

Fig 5.2: Input singal for offline model-free DDoE (2D case). (a) Example of input space using APRBS signals. (b) Correction result to satisfy operable constraints.

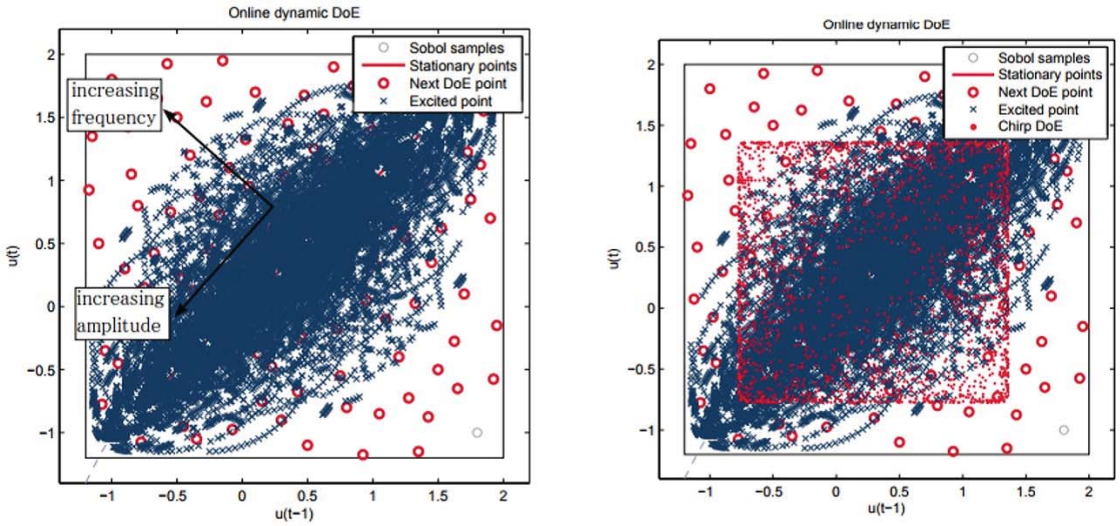


Fig 5.3: Comparison of dynamical input space, model-free offline DoE and online DDoE : N. Tietze (2015)[25]

なる. 一方でモデルフリー DDoE の課題である学習データ点数はオフラインの場合と比較しても更に膨大になってしまう.

5.2.3　モデルベース実験計画法 (オフライン)

モデルベース DoE ではモデルの構造を仮定し, モデルの設計パラメータを精度良く獲得する入力を求める. この手法はグレーボックスモデルと相性が良く, 自動車エンジンの分野においては, 特に D 最適計画法 [45] が広く用いられている. また, モデルに

関する知識が必要となるものの, モデルフリーの手法と比較して少ない学習点でモデルを獲得できる可能性がある. しかしながら, 動的モデルを考えた場合にその入力列を設計することは難しく, 次で紹介するオンライン手法が用いられる.

5.2.4 モデルベース動的実験計画法 (オンライン)

オンライン DDoE において基本的な考え方は, モデル予測制御のように予測ホライゾンを考え, その区間のモデル評価を異常運転を発生させずに最大とするような入力列を設計することである. 特に NARX モデルやニューラルネットワークといった入出力モデルに対し, 予測ホライゾン内のフィッシャー情報量を評価する D-最適化ベースのモデルベース DDoE が提案されている [73][74][77][78].

5.3 ガウス過程モデルの実験計画法

前節で述べたようにモデルベースのオンライン DDoE は効率的な学習データの取得が期待できる一方でモデルの構造を与える必要がある. 4 章では GP ベースモデルの有効性が示されたため, ここからは NARX-GP モデルに対するモデルベース DDoE を考える. そのためにまずは情報ゲイン [79] を用いた静的 GP モデルに対する入力列設計 [80] について述べた後, 長期予測と多出力系に拡張する.

2 章と同様に, 観測ノイズを含む入力の学習データを $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{n_x \times N}$, $\boldsymbol{x} \in H$ 出力の学習データを $\boldsymbol{y} = [y_1, y_2, \ldots, y_N]^T \in \mathbb{R}^{N \times 1}$ とする. 情報量ゲイン $F(\boldsymbol{X})$ を f と $\boldsymbol{y} = \boldsymbol{f} + \boldsymbol{\varepsilon}_N, \boldsymbol{\varepsilon}_N \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$ との相互情報量 $I(\boldsymbol{y}; f)$ でとるとすると, エントロピー関数 H を用いて以下で表される.

$$
\begin{aligned}
F(\boldsymbol{X}) &= I(\boldsymbol{y}; f) = H(\boldsymbol{y}) - H(\boldsymbol{y}|f) \\
H(\boldsymbol{A}) &= -\sum_{\boldsymbol{a} \in \mathrm{domA}} p(\boldsymbol{a}) \log p(\boldsymbol{a}) \\
H(\boldsymbol{A}|\boldsymbol{B}) &= -\sum_{\substack{\boldsymbol{a} \in \mathrm{domA} \\ \boldsymbol{b} \in \mathrm{domB}}} p(\boldsymbol{a}, \boldsymbol{b}) \log p(\boldsymbol{a}|\boldsymbol{b})
\end{aligned}
\tag{5.1}
$$

これは $\boldsymbol{y}$ を獲得した際の f の不確かさの減少量を示す. ここで, 正規分布のエントロピーは共分散行列を用いて次で算出される.

$$
H(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})) = \frac{1}{2} \log |2\pi e \boldsymbol{\Sigma}| \tag{5.2}
$$

したがって, 相対情報量は, 次で与えられる.

$$
I(\boldsymbol{y}; f) = I(\boldsymbol{y}; \boldsymbol{f}) = \frac{1}{2} \log |\boldsymbol{I} + \sigma^{-2} \boldsymbol{K}| \tag{5.3}
$$

f の不確かさの減少を最大とするような N 組の学習データについて解くことは NP 困難あるが, k サイクル目のデータとして $\boldsymbol{x}_k = \mathrm{argmax}_{\boldsymbol{x}_* \in H} F(\boldsymbol{X}_{k-1} \cup \{\boldsymbol{x}_*\})$ とすることで貪欲法により準最適解を求めることができる [81]. 更に計算をすすめると,

$$\begin{aligned}\boldsymbol{x}_k &= \underset{\boldsymbol{x}_* \in H}{\mathrm{argmax}}\, F(\boldsymbol{X}_{k-1} \cup \{\boldsymbol{x}_*\}) \\ &= \underset{\boldsymbol{x}_* \in H}{\mathrm{argmax}}\, \frac{1}{2} \log |\boldsymbol{I} + \sigma^{-2} \boldsymbol{K}_{\boldsymbol{X}_k - 1, \boldsymbol{X}_k - 1}| \\ &\qquad \times |\sigma^{-2} k(\boldsymbol{x}_*, \boldsymbol{x}_*) - k(\boldsymbol{x}_*, \boldsymbol{X}_{k-1})(\boldsymbol{I} + \sigma^{-2} \boldsymbol{K}_{\boldsymbol{X}_{k-1}, \boldsymbol{X}_{k-1}})^{-1} k(\boldsymbol{X}_{k-1}, \boldsymbol{x}_*)| \\ &= \underset{\boldsymbol{x}_* \in H}{\mathrm{argmax}}\, \sigma^2_{f*} | \boldsymbol{x}_*, \boldsymbol{X}_{k-1}, \boldsymbol{y}_{k-1}\end{aligned} \tag{5.4}$$

となる. ただし, $\boldsymbol{X}_k = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k]$ かつ $\boldsymbol{y}_k = [y_1, y_2, \ldots, y_k]^T$ である. 即ち, 静的ガウス過程モデルの入力列設計は逐次的に予測分散が最大となる入力を付加するモデルベースのオンライン DoE で求めることができる. 以後この情報量ゲインベースの入力列設計手法 (5.4) を GPDoE と呼ぶこととする. 式 (5.4) においてハイパーパラメータの更新は考慮していないが, 実際的には定期的にハイパーパラメータを更新していくことになる. 簡単な対象に GPDoE 適用した例を Fig. 5.4 に示す. ランダムな入力データを用いた場合と比較して, 不確定性の少ないモデルが獲得できていることが分かる.

多出力システムの場合も情報ゲインベースの入力列設計を考えることができる. 特に本論文では出力は独立と仮定し, 出力毎に NARX-GP を学習することを想定している. $\boldsymbol{y}^m$ を出力の m 番目の要素に対する出力の学習データとすると, 出力の結合エントロピー $H(\boldsymbol{y}^1, \ldots, \boldsymbol{y}^n)$ は

$$H(\boldsymbol{y}^1, \ldots, \boldsymbol{y}^n) = \sum_{m=1}^{n} H(\boldsymbol{y}^m) \tag{5.5}$$

となるため, 単出力の場合と同様に計算をすすめると,

$$\boldsymbol{x}_k = \underset{\boldsymbol{x}_* \in H}{\mathrm{argmax}} \sum_{m=1}^{n} \sigma^2_{f*} | \boldsymbol{x}_*, \boldsymbol{X}_{k-1}, \boldsymbol{y}^m_{k-1} \tag{5.6}$$

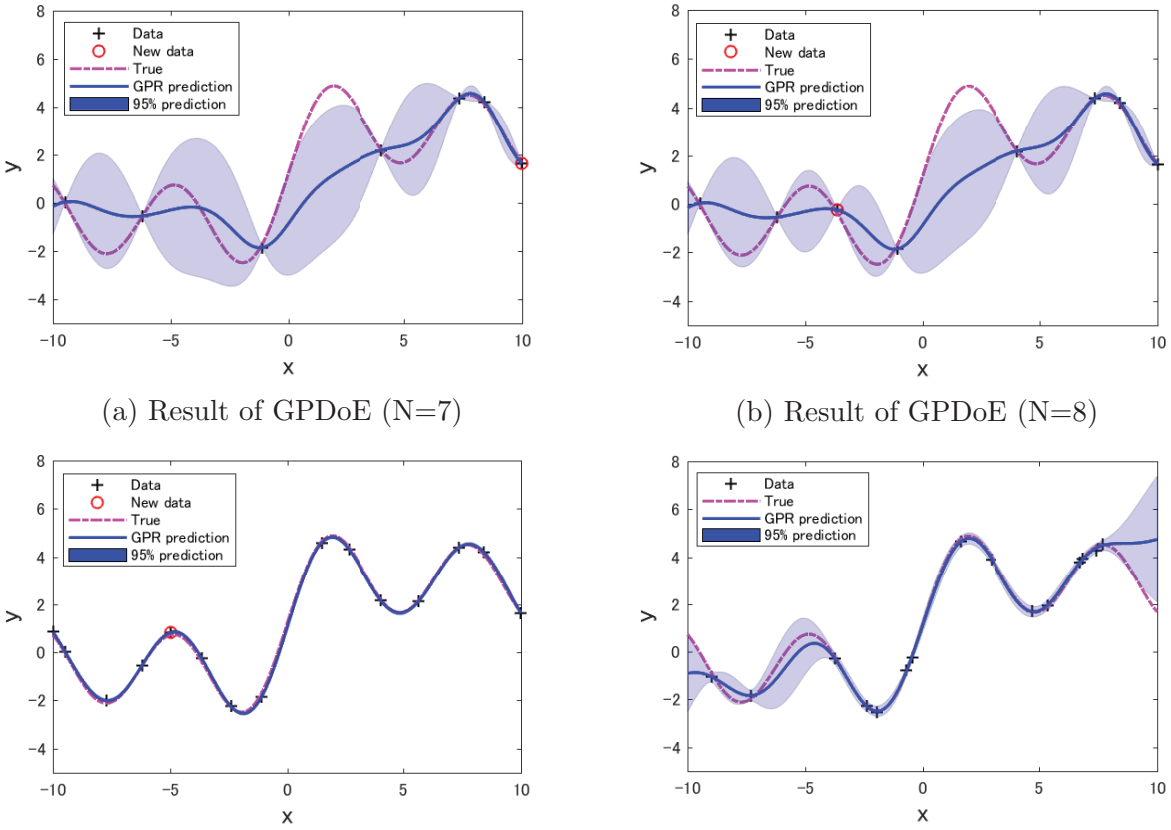
のように複数の GP モデルを用いた予測分散の和を用いて評価することができる. 一般に出力は要素毎にスケールが異なるため, データの正規化や重み付けで対応する必要があることに注意する.

また, $\boldsymbol{x}$ と出力 y が独立であるとすると, 予測ホライゾン T 内の情報ゲインは

$$H(\boldsymbol{y}_k) = H(\boldsymbol{y}_{k-1}) + H(y_k | \boldsymbol{y}_{k-1}) \tag{5.7}$$

の関係を用いると, 予測区間内でハイパーパラメータを更新しない場合

$$I(\boldsymbol{y}_{N+T}; \boldsymbol{y}_{N+T}) = \frac{1}{2} \sum_{k=1}^{T} \log(1 + \sigma^{-2} \sigma^2_{N+k}(\boldsymbol{x}_k)) \tag{5.8}$$



(a) Result of GPDoE (N=7)

(b) Result of GPDoE (N=8)

(c) Result of GPDoE (N=15)

(d) Random data (N=15)

Fig 5.4: Exmple of GPDoE (Initial data number $N_0 = 5$).

と表すことができる.

5.4 ガウス過程モデルの動的実験計画法

GPDoE は入力 $\boldsymbol{x}$ と出力 y が独立かつ, 入力を任意に取れる場合には効果的であるものの, NARX モデル (2.33) は入力にダイナミクスを含み, かつ入出力が独立でないため直接適用することができない. そこで本節では, 入出力関係を独立とするような新たなモデルを考え, そのモデルに対して情報ゲインベースの入力列設計法を適用する.

5.4.1 静的境界モデルを用いた入力制約

まず定常入力により静的境界モデル $h_s(\boldsymbol{u})$ を求め, 同定した静的境界モデルを入力制約として用いる (Fig. 5.5). 一般には凸包の境界を求め, 更に保守的にしたものを入力制約とするが, ここではより最適な制御を目指すために, Algorithm 1 の GPC によって同定した非凸を用いるとする. この手法は, 逐次的に GPC モデルを構築することで,

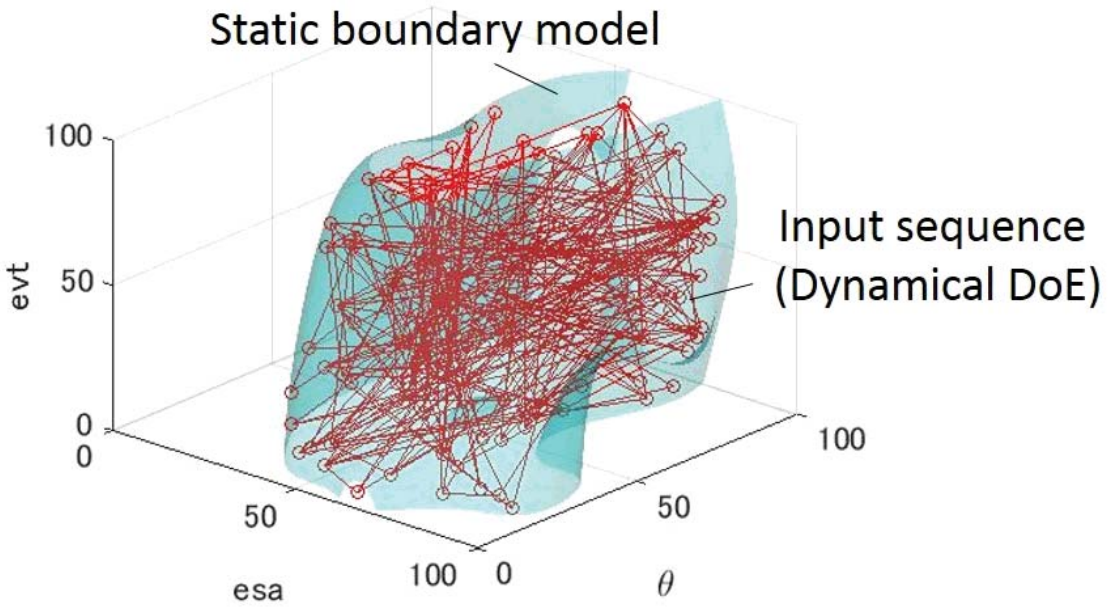


Fig 5.5: Example of the static boundary model and the input sequence generated by a DDoE. In this case the input dimensions are three ($\omega = 1500$[rpm] $egr = 0[mm]$). The engine operating conditions are divided into 100×100×100 grids.

異常運転領域で運転しない, 効率的な入力列設計を行うものであり, 入力の次元数が多い場合に特に効果的である. 一方で静的境界モデルによる入力制約のみでは異常運転に対応することができないため, 提案する DDoE では動的境界モデルの学習も同時に行っていく.

5.4.2 動的実験計画法

ここでは, 長期ホライゾンの考え方を応用し, 各出力要素に対する新たな評価指標を考える. まず次のような観測出力の NARX モデルを例に長期予測を考える.

$$\boldsymbol{y}_{k+1} = f(\boldsymbol{y}_k, \cdots, \boldsymbol{y}_{k-p_o}, \boldsymbol{u}_k, \cdots, \boldsymbol{u}_{k-p_i}) + \boldsymbol{w}_k \tag{5.9}$$

このとき t ステップ後の予測値はある確率モデル f_t を用いて

$$\boldsymbol{y}_{k+t} = f_t(\boldsymbol{y}_k, \cdots, \boldsymbol{y}_{k-p_o}, \boldsymbol{u}_{k+t-1}, \cdots, \boldsymbol{u}_{k-p_i}, \boldsymbol{w}_{k+t}, \ldots, \boldsymbol{w}_k) + \boldsymbol{w}_{k+t} \tag{5.10}$$

と書ける. 更に $\boldsymbol{w}_k$ がステップに依存しない分布を持つ変数 $\boldsymbol{w}$ で表せるとし,

$$\boldsymbol{c}_{k-1} = [\boldsymbol{y}_k^T, \cdots, \boldsymbol{y}_{k-p_o}^T, \boldsymbol{u}_{k-1}^T, \cdots, \boldsymbol{u}_{k-p_i}^T]$$

とおくと, 確率モデル $\tilde{f}_t$ により次のように表すことができる.

$$\boldsymbol{y}_{k+t}|\boldsymbol{c}_{k-1}, \boldsymbol{w} = \tilde{f}_t(\boldsymbol{u}_{k+t-1}, \cdots, \boldsymbol{u}_k) \tag{5.11}$$

この形式では出力は入力に対して独立となるが, 長期予測を考慮した (5.8) による情報ゲインベースの入力列設計は扱うことができないことに注意する. そこで本節では重みベクトル $\boldsymbol{r}^m \in \mathbb{R}^T$ を用いて,

$$\begin{aligned} \boldsymbol{s}_k^m &= \sum_{i=1}^{T} \boldsymbol{r}_i^m \boldsymbol{y}_{k+i}^m \\ \boldsymbol{s}_k|\boldsymbol{c}_{k-1}, \boldsymbol{w} &= F(\boldsymbol{u}_{k+T-1}, \cdots, \boldsymbol{u}_k) \end{aligned}$$

となるような新たな出力 $\boldsymbol{s} \in \mathcal{R}^{n_y}$ と関数 F を考える. ここで, T は設計パラメータで予測ホライズン長である. つまり. 直接 NARX モデル (2.33) の入力列設計を行うのではなく, その平均的な振る舞いを予測するような確率モデル F を GP で近似することを考え, その GP モデルを良く表すような入力列を設計する. この GP モデル学習には GPDoE を直接適用することができる. NARX-GP モデルでは出力への観測ノイズの伝播の影響を陽に考慮せずにガウスノイズとして扱った上で学習を行ったことに注意すると, DDoE で扱うモデルは境界モデルを含み以下のようにまとめられる.

$$\begin{aligned} \boldsymbol{s}_k^m &= \sum_{i=1}^{T} \boldsymbol{r}_i^m \boldsymbol{y}_{k+i}^m \\ \boldsymbol{b}_k^m &= \sum_{i=1}^{T} \boldsymbol{p}_i^m \boldsymbol{z}_{k+i-1}^m \\ \boldsymbol{s}_k &= F_{gp}(\boldsymbol{c}_{k-1}, \boldsymbol{u}_{k+T-1}, \cdots, \boldsymbol{u}_k) \\ \boldsymbol{b}_k &= G_{gp}(\boldsymbol{c}_{k-1}, \boldsymbol{u}_{k+T-1}, \cdots, \boldsymbol{u}_k) \end{aligned} \tag{5.12}$$

ここで, $T \geq p_o + 1$ かつ $T \geq p_i + 1$ を満たすとする. また, $\boldsymbol{p}^m \in \mathbb{R}^T$ は重みベクトルである. ステップ k の時点で獲得したデータにより学習された F_{gp} モデルを $\hat{F}_k$, G_{gp} モデル $\hat{G}_k$ と書くとする.

提案する動的実験計画法では, (5.12) のモデルに対して GPDoE を適用する. 即ち, ステップ k において次のような最適化問題を考え, 最適な入力列 $\boldsymbol{U}_k^*$, $\boldsymbol{U}_k = [\boldsymbol{u}_{k+T-1}^T, \cdots, \boldsymbol{u}_k^T]^T$

を求める.

$$
\begin{aligned}
\boldsymbol{U}_k{}^* &= \underset{\boldsymbol{U}_k}{\operatorname{argmax}} \left(\sum_{m=1}^{n_y} \sigma^2_{\boldsymbol{s}_k^m} + \sum_{m=1}^{n_z} \sigma^2_{\boldsymbol{b}_k^m} \right), \\
\text{s.t.} \quad \boldsymbol{s}_k^m &\sim N(\mu_{\boldsymbol{s}_k^m}, \sigma^2_{\boldsymbol{s}_k^m}), \\
\boldsymbol{s}_k &= \hat{F}_k(\boldsymbol{c}_{k-1}, \boldsymbol{U}_k), \\
\mu_{\boldsymbol{b}_k^\kappa} &\leq \epsilon_b, \quad 0 < \epsilon_b, \quad \kappa = 1, \ldots, 2l, \\
\boldsymbol{b}_k^m &\sim N(\mu_{b_k^m}, \sigma^2_{b_k^m}), \\
\boldsymbol{b}_k &= \hat{G}_k(\boldsymbol{c}_{k-1}, \boldsymbol{U}_k), \\
h_s(\boldsymbol{u}_i) &\leq -\epsilon_h, \quad i = k, \ldots, k+T-1, \quad 0 \leq \epsilon_h, \\
\Delta \boldsymbol{u}_{min} &\leq \Delta \boldsymbol{u}_j \leq \Delta \boldsymbol{u}_{max}, \\
\Delta \boldsymbol{u}_j &= \boldsymbol{u}_j - \boldsymbol{u}_{j-1}, \quad j = k, \ldots, k+T-1,
\end{aligned}
\tag{5.13}
$$

ここで, ϵ_b 及び ϵ_h は保守性に関わるパラメータ, $\Delta \boldsymbol{u}_{min}$ と $\Delta \boldsymbol{u}_{max}$ は入力変化速度に関係する制約条件である. 式 (5.13) によって計算された入力列のうち, $\boldsymbol{u}_k^*$ をステップ k での入力としてシステムに加え, 次のステップでも同様に入力列を計算していく. その際に獲得したデータをもとに逐次的に GP モデル $\hat{F}_k$ と $\hat{G}_k$ を更新していくが, ハイパーパラメータの学習は行わないものとする. GP の予測には式 (2.3) で示したように, $(\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1}$ を計算する必要があり, これがオンライン計算のボトルネックとなるが, ハイパーパラメータの更新がない場合は区分行列の逆行列計算により容易に計算できる (付録 A.3). ある程度のデータが獲得できた後は実験環境を初期化するのと同時にハイパーパラメータをオフラインで学習し, 更新したハイパーパラメータを用いて再度オンライン入力列設計を行う. 以上の行程を十分な学習データが得られるまで繰り返す. しかしながら, 一般に最適な入力列をオンラインで計算するのは困難であり, 更なる単純化が必要となる.

提案する動的実験計画法のアルゴリズムを Algorithm 3 にまとめる.

Algorithm 3 Dynamical DoE Based on GP

1: **Preparation**:
2: Stationary inputs are added to obtain a static boundary model $h_s(\boldsymbol{u})$ using Algorithm 1.
3: Acquire initial training data set $\mathcal{D}_{0,0}$ with the constraint $h_s(\boldsymbol{u}) \leq -\epsilon_h,\ 0 \leq \epsilon_h$.
4: **Dynamical DoE**:
5: **Outer Loop**: For $t = 0, \ldots, l_O$, perform:
6: Initialize the experiment environment
7: Calculate hyperparameters θ_t and GP model ($\hat{F}_{0,t}$, $\hat{G}_{0,t}$) with training data $\mathcal{D}_{0,t}$.
8: **Inner Loop**: For $k = 0, \ldots, l_I$, perform:
9: Determine a next input $\boldsymbol{u}^*_{k,t}$ in (5.13).
10: Apply the input point $\boldsymbol{u}^*_{k,t}$ to the system and obtain an new training data $\mathcal{D}_{k,t}$.
11: Update GP model to $\hat{F}_{k+1,t}$ and $\hat{G}_{k+1,t}$ with training data.
12: **End**
13: **End**
14: **Output**:
15: Generate GP-NARX models (2.33) with input and output data.

5.5 簡易エンジンシミュレータを用いた検証

提案手法に対して, まず簡単なエンジンシミュレータを用いた検証を行う [82][83]. 詳細は付録 C にまとめるが, 離散化した以下のシステムモデルを考える.

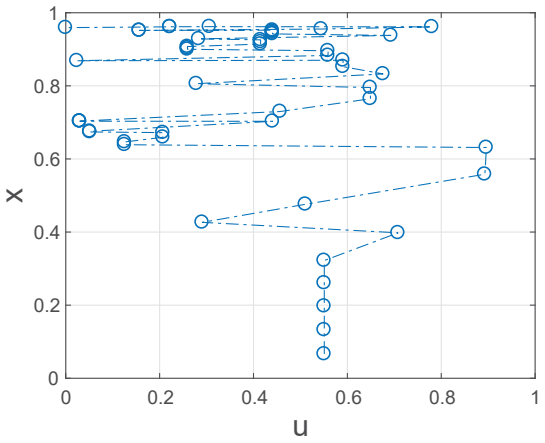
$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} \theta_1 x_1(k) + \theta_2(1 - x_2^2(k))u \\ \theta_3 x_2(k) + \theta_4(1 - x_2^2(k))u \end{bmatrix} \tag{5.14}$$

$$h(k) = \left(\frac{P_c(k)}{\alpha}\right)^2 + \left(\frac{T_c(k)}{\beta} - 1\right)^2 \tag{5.15}$$

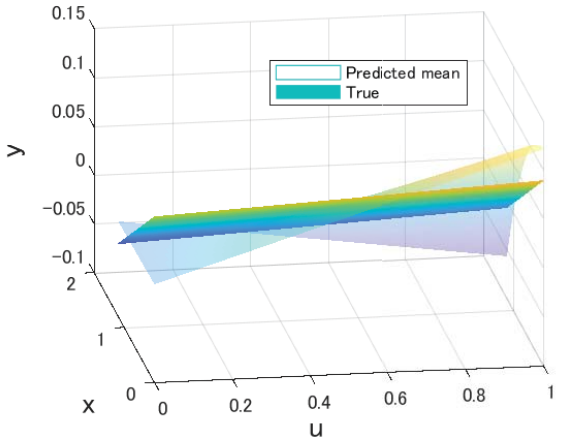
$$\text{s.t.} \qquad 0 \leq u(k) \leq 1 \tag{5.16}$$

ここで, k はステップ数, $\theta_1, \theta_2, \theta_3, \theta_4, \alpha, \beta$ は定数, $P_c(k) = x_2(k)$, $T_c(k) = x_2(k)/x_1(k)$ であり, $x_1(k)$ 及び $x_2(k)$ は観測できるとする. $h(k)$ はノック現象を模擬しており, $h(k) > 1$ となった場合にノックが発生するとする. 即ち, このシステムを同定するための入力列設計を考える場合, $h(k) \leq 1$ という制約条件を満たすようにする必要がある.

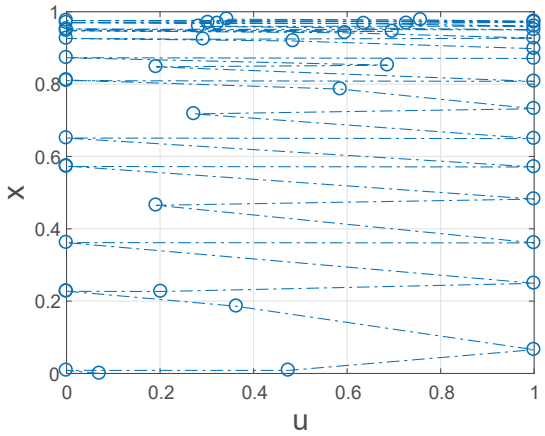
最初に制約条件 $h(k) \leq 1$ を考慮しない場合のシステムモデル同定について検証する. 可視化のために $x_2(k)$ の信号に注目し, $y(k) = x_2(k+1) - x_2(k)$ の同定を目的とした例を Fig. 5.6 に示す. この例では入力として APRBS 信号を加えた場合と, GPDDoE を適用した場合を比較している. 入力空間として $u - x_2$ 空間を考えると y を表現するのには十分であるが, 入力列として APRBS 信号を 50 ステップ付加した場合には入力に偏りがあり, $y(k) = f_{gp}(x_2(k), u(k))$ の GP モデルを考えるとそのモデルは真のシス



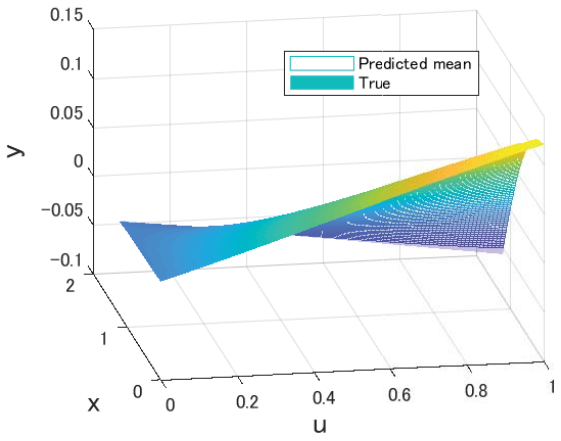
(a) Input space using APRBS signal



(b) GP model using APRBS signal



(c) Input space using GPDDoE

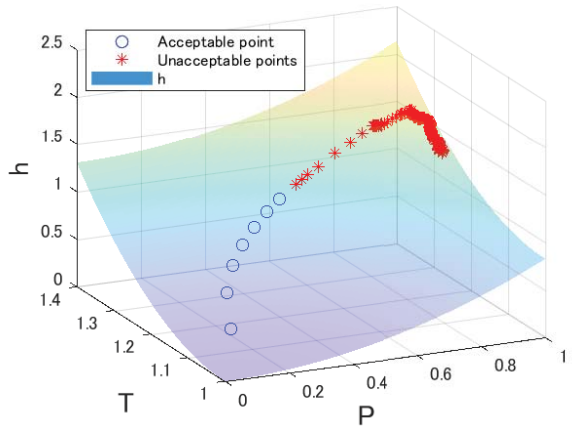


(d) GP model using GPDDoE

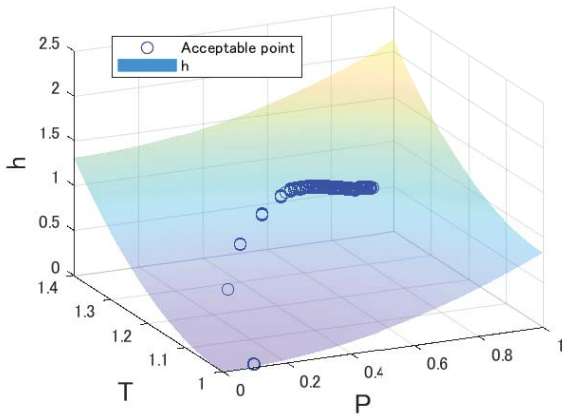
Fig 5.6: Comparison result between GPDDoE and APRBS without boundary constraints.

テムの応答と大きな差がでてしまう. 一方で, GPDDoE を用いた場合では入力空間での偏りが少なく, GP モデルで非常によく表現できていることがわかる. これは直接 $y(k) = f_{gp}(x_2(k), u(k))$ の実験計画法を考えずに, 長期予測を含むモデル (5.12) に対して GPDoE(5.4) を用いることで間接的に y のモデルをよく表現できるような入力が求まることを意味する. GPDDoE の初期点としては APRBS 信号から獲得した 10 点を用い, 予測ホライゾン $T = 3$ で最適化問題を解くことで 50 点の追加入力を獲得している. また予測分散和の最大値探索には MATLAB R2017a の粒子群最適化ソルバーである”particleswarm”を使用している.

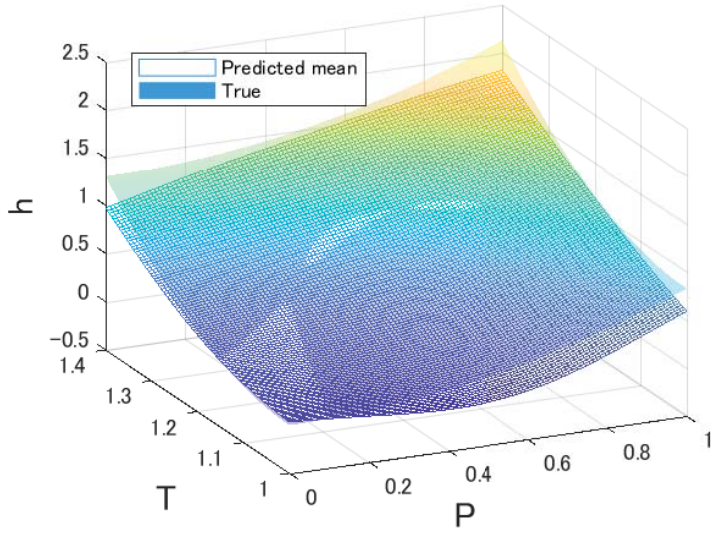
次にノックモデルを境界条件を含んだ入力列設計を考える. このノックモデルの定常状態を扱う静的境界モデルは入力制約 $0 \leq u \leq 1$ に含まれる. 一方で, 過渡応答においては入力制約内でもノックが発生するため, 動的境界モデルを考慮することが必要になる. Fig. 5.7 に APRBS 信号を付加した場合と, 境界制約を考慮した GPDDoE を用



(a) Abnormal operation signal excited with APRBS signal



(b) Abnormal operation signal in GPDDoE with constraint

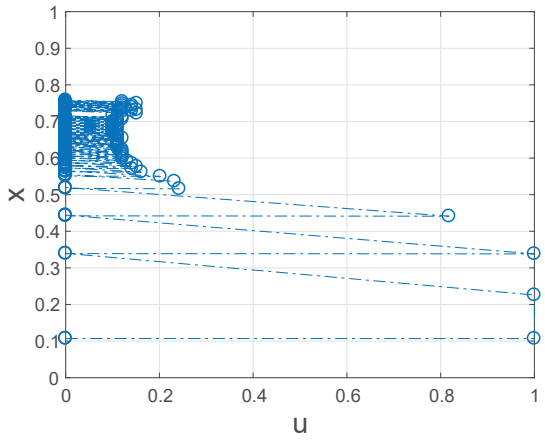


(c) GP model of the boundary function using GPDDoE.

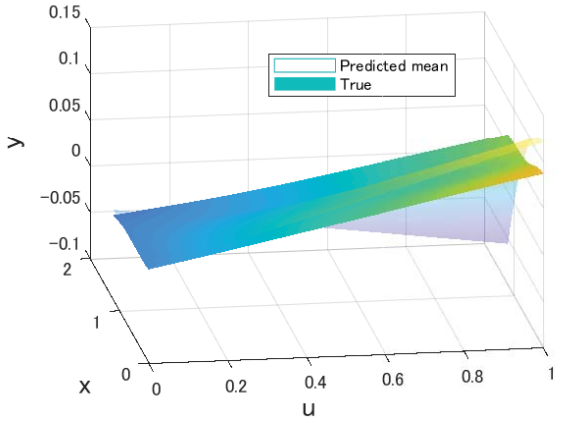
Fig 5.7: Result of boundary identidication using GPDDoE with boundary constraints.

いた場合を示す．APRBS 信号を付加した場合は入力制約内の信号であっても境界制約を違反しているのがわかる．一方で動的境界モデルを GP モデルで同定しながらその境界モデルを制約にしながら入力列を設計した場合には境界制約を違反していないことがわかる．GPDDoE の初期学習データとしては APRBS 信号で獲得したものを用いている．GPDDoE で獲得したデータを用いたノックの GP モデルは真のモデルの傾向を良く示していることがわかる．

Fig. 5.8 では境界制約を考慮した入力列設計で得られた y の入力空間とその入力により同定した GP モデルを示す．境界条件の影響で入力の偏りが発生していることがわかる．そのため GP モデルでは入力が不足している箇所の精度が境界制約がない場



(a) Input space using GPDDoE with constraint



(b) GP model using GPDDoE with constraint

Fig 5.8: System model identidication using GPDDoE with boundary constraints.

合と比較して悪化している.

今回の例では入力列の設計回数を $l_O = 1$ と設定しているが, l_O を増やすことで学習データは増加するもののモデルの精度は向上することが期待できる.

5.6 エンジンシミュレータへの適用

本節では, 提案手法をエンジンベンチマークシミュレータに適用した結果を述べる. 自動車のメカニカルモデルは与えられていないため, ここではエンジン回転速度を $\omega = 1500$[rpm] と固定した場合を考えている. また, エンジンの 1 サイクルを 1 ステップとし, 特に第一気筒に注目してモデルを構成する.

5.6.1 学習データの獲得

最初に Algorithm 3 により, GP モデル学習のための入力列設計を行う. ここでは初期データとして, 保守的な領域において一様乱数を用いて決定した入力を用いた 200 点のデータを使用する. その後 6 回のオンライン入力列設計 (Fig. 5.9) を行い, その都度 50 点のデータ獲得とハイパーパラメータの更新を行う. 最終的に得られた計 500 点のデータを学習データとして用いる. また, 初期データから GP-NARX を学習することで, モデル次数を $p_o = p_i = 5$ とし, 予測ホライズンは $T = 6$ と定めた. 更に計算コストを減らすため, 入力は $\boldsymbol{u}_k = \boldsymbol{u}_i, i = k+1, \ldots, k+T-1$, と一定値をとり続けるとして計算を行っている.

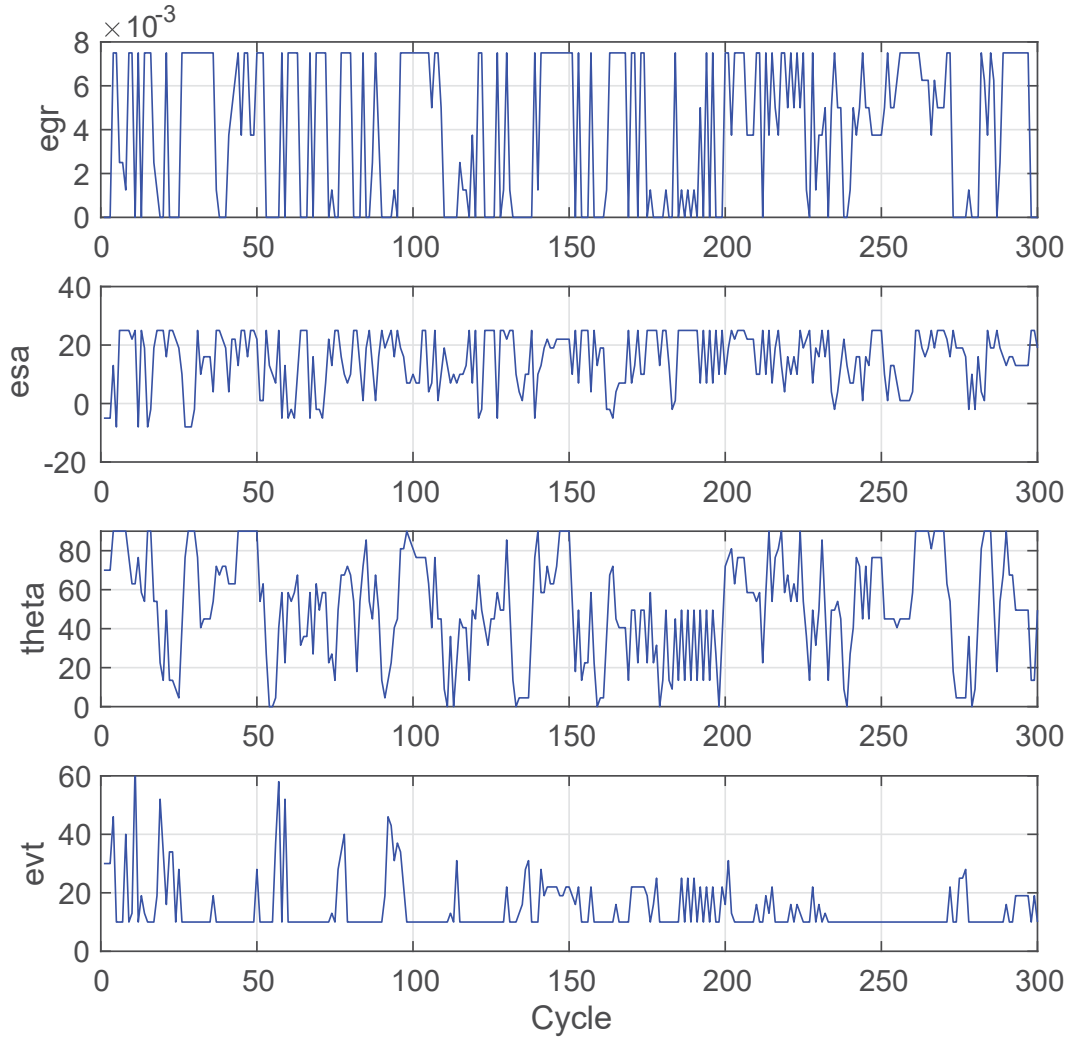


Fig 5.9: Example of the input sequence generated by the dynamical DoE based on GP-NARX model.

5.6.2 GP-NARX モデルの構築

次に GPDDoE によって得られた入力を用いて, GP-NARX モデルを学習, ハイパーパラメータを決定する. 学習データ点数が多い場合は疑似入力を導入し [31][34], GP 及び RRGP の計算コストを低減させることができる. Table. 5.1 に 500 点の学習データから同定した GP-NARX モデルの精度を示す. また, Fig. 5.10 に境界モデル及びトルクモデルの予測結果と実際の応答を示す. 検証データとしては 4 章と同様のデータを使用している. ここでは, Algorithm 3 に基づく GPDDoE を適用した場合と, 非凸境界内の入力を一様乱数入力を用いて選択した場合を比較しているが, 提案手法である GPDDoE の方が良い結果を示しているといえる. 次章で扱う吸気マニホルド内圧力 pm 予測検

証のために, pm の GP-NARX モデルの精度を向上させるよう式 (5.12) の入力探索重みを設定しているため, 特に pm の精度が良くなっている.

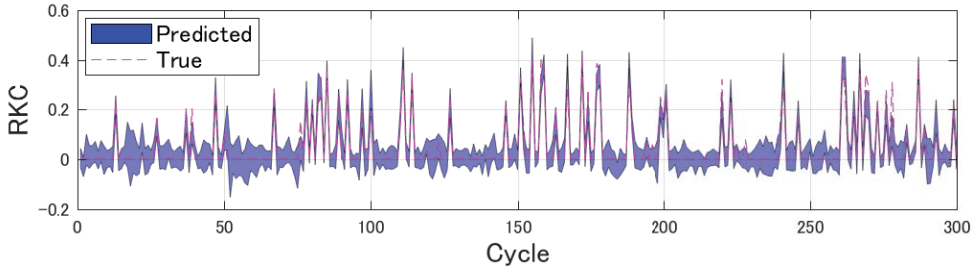
一方で, 異常運転モデルの精度は向上していないことが分かる. Fig. 5.11 に 6 ループ目 ($t = 6$) の GPDDoE における異常運転モデルの予測値と実際の応答を示すが, 入力列設計においては異常運転を予測しながら運転可能領域で動作していることがわかる. しかしながら, 運転可能領域のデータを取るが故に, 異常運転におけるデータが不足しており, 結果として異常運転モデルの精度が低下していると考えられる.

Table 5.1: Result of accuracy of the GP-NARX model (The number of training data points is 500)

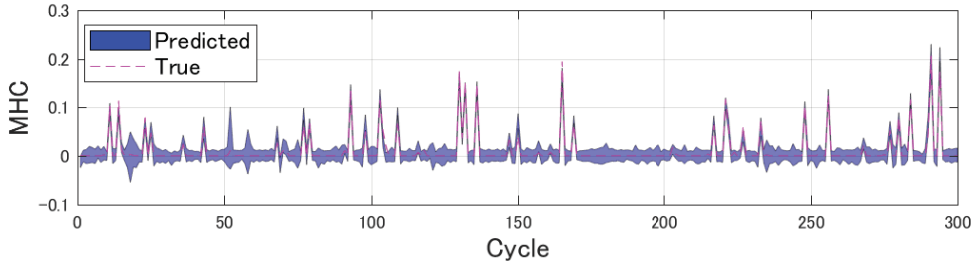
Predicted signal	Uniform random input (MSE)	Algorithm 3 (MSE)
m_t	2.515 e-05	1.922 e-05
α	1.642 e-01	6.833 e-02
pm	3.637 e-01	3.080 e-02
$\boldsymbol{R\_KC}_1$	7.019 e-04	9.200 e-04
$\boldsymbol{M\_HC}_1$	1.216 e-06	1.180 e-06
$\boldsymbol{Tor}_1$	3.208 e+01	3.163 e+01

5.7 おわりに

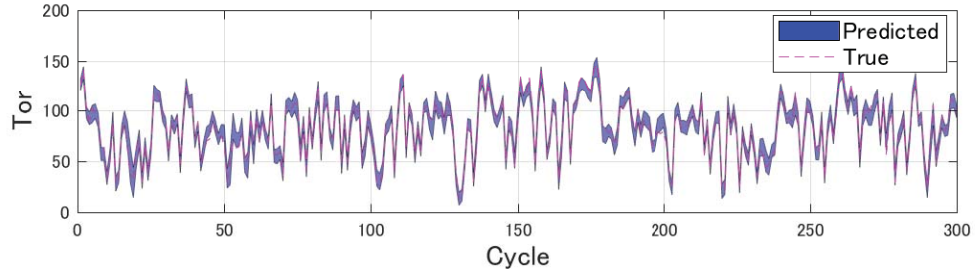
本章では自動車エンジンの動的モデルを構成する為のガウス過程を応用した動的実験計画法について述べた. 特に NARX-GP モデルを構成するためのデータを獲得する動的実験計画法として, ガウス過程の予測分散を応用した新たな入力列設計手法を提案した. 最後にエンジンシミュレータを用いて, 提案した動的実験計画法の有効性を検証した. 提案手法は制約を満たしつつ効率的な入力列を探索できるものの, 入力列設計に必要な計算コストは少なくなく, その高速化に課題が残っている. また, 多出力系に対しての重みや予測ホライゾン長の影響を解析し, その設計手法を提案する必要がある.



(a) Estimated ***R_KC***

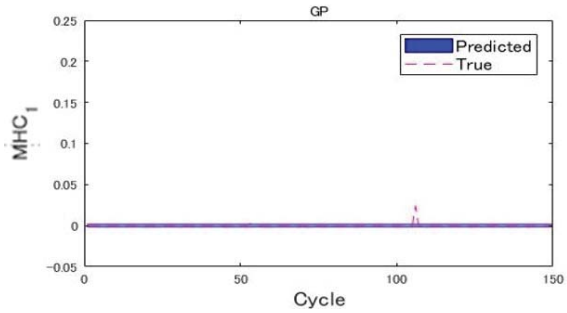


(b) Estimated ***M_HC***

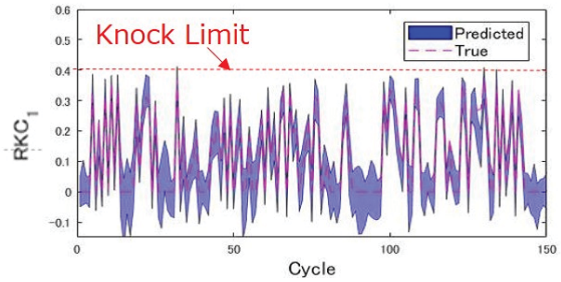


(c) Estimated ***Tor***

Fig 5.10: Modeling result using GPDDoE (training data: 500 cycles).



(a) Estimated ***M_HC***



(b) Estimated ***R_KC***

Fig 5.11: Result of abnormal operation signal using GPDDoE.

第6章　ロバスト逐次ガウス過程を用いたオンライン学習

6.1　はじめに

量産されたエンジンは個体差や経年劣化により性能にばらつきがあり, 更にその性能は燃料の種類や湿度, エンジン水温などの短期的に変化する運転条件によっても左右される. このようなエンジン性能の変化に素早く対応できるよう, 自動車エンジンではオンラインでのモデル学習手法が実装されている. この章では GP を応用した新たなオンライン学習手法について議論する.

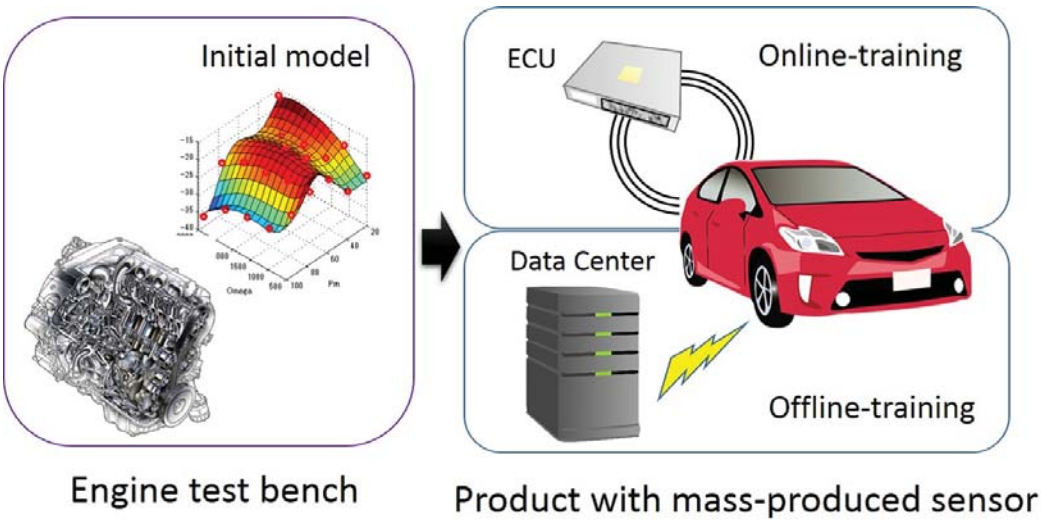


Fig 6.1: Concept of model update.

GP はカーネル関数としてガウスカーネルや ARD カーネルを選択した場合には, 一般に学習によるオーバーフィッティングが起こりにくい. また, 学習データが少ない場合には, 深層学習に比べ汎化性能に優れるという特徴を持つ. 一方で GP や深層学習といった統計的モデルを用いて学習を行う場合, 外れ値 (ここでは実用上, 異常値も外れ値として扱う) がモデルの学習に大きな影響を及ぼす. 開発の段階において, 自動車エンジンはエンジンテストベンチと呼ばれるエンジン適合用の施設においてそのモデル化や制御マップ適合が行われることとなる. 適合を行う際には専用の多種多様な

センサーを用いて, 良い環境のもとで実機を運転できるため, これまでの議論では外れ値を考慮してこなかった. しかしながら, 商品化後のエンジンではコスト面から安価なセンサーを最低限の種類で実装することが望まれる上, 悪条件での運転も想定される. したがって自動車エンジンのオンライン学習を考えた場合, 学習データへの外れ値の混合は避けては通れないものであり, 実用的にはロバストかつ高速なオンライン学習手法が要求される. この章では, 観測値に外れ値が混ざるような場合にも有効な GP のオンライン学習として, ロバスト逐次ガウス過程 (Robust Recursive Gaussian Process, RRGP)[84] の適用を提案する. この手法では GP のハイパーパラメータ学習といった計算コストの大きい学習計算を車外のサーバーなどによりオフラインで中長期的に行うことを想定した上で, 計算コストの少ないモデル更新を ECU を用いてオンラインで行うことで制御対象の短期的な変化に対応する (Fig. 6.1). 提案手法の有用性は Society of Automotive Engineers (JSAE) と Society of Instrument and Control Engineers (SICE) から提供されているノッキングと失火のモデルを含むエンジンベンチマーク問題 [11][12] にて検証する.

6.2 逐次ガウス過程

逐次ガウス過程 (Recursive Gaussian Process, RGP) はガウス過程回帰の近似手法の一つであり [65][66][84], オンラインで推定関数の更新を行う. 本来はガウス過程の計算量を減らすための近似手法として提案されているが, その逐次性を生かしオンライン学習への応用を考える. 4 章で示したように RGP は GP の近似手法と組み合わせることで, 予測精度を確保しながらの高速なモデル更新が期待できるためである.

まず, GP と同様に, 学習データ $\mathcal{D} = (\boldsymbol{X}, \boldsymbol{y})$ を用いて $\boldsymbol{f} = f(\boldsymbol{X})$ を定義する. $\boldsymbol{f}$ が初期分布 $p(\boldsymbol{f}) = \mathcal{N}(\boldsymbol{f}|\boldsymbol{\mu}_0^f, \boldsymbol{C}_0^f)$ を持つ GP であると仮定する. ここで $\boldsymbol{\mu}_0^f = \boldsymbol{0}, \boldsymbol{C}_0^f = k(\boldsymbol{x}, \boldsymbol{x})$ であり, これは GP における事前分布と等しい. GP では, 一度定義された事前分布は変わらないが, RGP における事前分布は, 新たな入力 $\boldsymbol{x}^t = [\boldsymbol{x}_1^t, \boldsymbol{x}_2^t, \ldots, \boldsymbol{x}_{n_t}^t] \in \mathbb{R}^{n_x \times n_t}$ とそれに対応する観測値 $\boldsymbol{y}^t = [y_1^t, y_2^t, \cdots, y_{n_t}^t] \in \mathbb{R}^{1 \times n_t}$ が与えられるとオンラインに更新される. 観測値 $\boldsymbol{y}^{1:t}$ が与えられたときの事後分布 $p(\boldsymbol{f}|\boldsymbol{y}^{1:t})$ を求めることについて考える. ここで $t-1$ までは更新されているとすると, t での事後分布は以下のように書くことができる.

$$
\begin{aligned}
p(\boldsymbol{f}|\boldsymbol{y}^{1:t}) &= \int p(\boldsymbol{f}, \boldsymbol{f}_t|\boldsymbol{y}^{1:t}) \mathrm{d}f_t \\
&= \int c_t \cdot \underbrace{p(\boldsymbol{y}^t|\boldsymbol{f}, \boldsymbol{f}_t) \cdot \overbrace{p(\boldsymbol{f}_t|\boldsymbol{f}) \cdot p(\boldsymbol{f}|\boldsymbol{y}^{1:t-1})}^{\text{予測ステップ}}}_{\text{更新ステップ}} \mathrm{d}f_t
\end{aligned}
\tag{6.1}
$$

ここで, $\boldsymbol{f}_t = f(\boldsymbol{x}^t)$ は新たな入力における潜在関数値であり, c_t は正規化定数である.

6.2.1 予測ステップ

式 (6.1) における予測ステップは以下の分布で表される.

$$\begin{aligned} p(\boldsymbol{f}, \boldsymbol{f}_t|\boldsymbol{y}^{1:t-1}) &= p(\boldsymbol{f}_t|\boldsymbol{f}) \cdot p(\boldsymbol{f}|\boldsymbol{y}^{1:t-1}) \\ &= \mathcal{N}(\boldsymbol{f}_t|\boldsymbol{\mu}_t^p, \boldsymbol{B}_t) \cdot \mathcal{N}(\boldsymbol{f}|\boldsymbol{\mu}_{t-1}^f, \boldsymbol{C}_{t-1}^f) \end{aligned} \tag{6.2}$$

ここで,

$$\boldsymbol{\mu}_t^p = \boldsymbol{J}_t \cdot \boldsymbol{\mu}_{t-1}^f \tag{6.3}$$

$$\boldsymbol{B}_t = k(\boldsymbol{x}^t, \boldsymbol{x}^t) - \boldsymbol{J}_t \cdot k(\boldsymbol{X}, \boldsymbol{x}^t) \tag{6.4}$$

$$\boldsymbol{J}_t = k(\boldsymbol{x}^t, \boldsymbol{X}) \cdot \boldsymbol{K}^{-1} \tag{6.5}$$

である. 2 つの分布はどちらもガウス分布であり, 同時分布を求めることができる. その同時分布を $p(\boldsymbol{f}, \boldsymbol{f}_t|\boldsymbol{y}^{1:t-1}) = \mathcal{N}(\boldsymbol{p}, \boldsymbol{P})$ とすると,

$$\boldsymbol{p} = \begin{bmatrix} \boldsymbol{\mu}_{t-1}^f \\ \boldsymbol{\mu}_t^p \end{bmatrix}, \ \boldsymbol{P} = \begin{bmatrix} \boldsymbol{C}_{t-1}^f & \boldsymbol{C}_{t-1}^f \boldsymbol{J}_t^{\mathrm{T}} \\ \boldsymbol{J}_t \boldsymbol{C}_{t-1}^f & \boldsymbol{C}_t^p \end{bmatrix} \tag{6.6}$$

となり, $p(\boldsymbol{f}_t|\boldsymbol{y}^{1:t-1})$ の平均値は $\boldsymbol{\mu}_t^p = \boldsymbol{J}_t \cdot \boldsymbol{\mu}_{t-1}^f$, 共分散は $\boldsymbol{C}_t^p = \boldsymbol{B}_t + \boldsymbol{J}_t \boldsymbol{C}_{t-1}^f \boldsymbol{J}_t^{\mathrm{T}}$ で表される. GP を用いた予測分布は式 (6.1) より、さらに $\boldsymbol{f}$ を周辺化することで得ることができる. t での出力の予測分布は、

$$p(\boldsymbol{y}^t|\boldsymbol{y}^{1:t-1}) = \int p(\boldsymbol{y}^t|\boldsymbol{f}_t) \cdot \int p(\boldsymbol{f}_t|\boldsymbol{f}) \cdot p(\boldsymbol{f}|\boldsymbol{y}^{1:t-1}) \mathrm{d}\boldsymbol{f} \mathrm{d}\boldsymbol{f}_t \tag{6.7}$$

より、$p(\boldsymbol{y}^t|\boldsymbol{y}^{1:t-1}) = \mathcal{N}(\boldsymbol{y}^t|\boldsymbol{\mu}_t^p, \boldsymbol{C}_t^p + \sigma^2 \mathrm{I})$ となることがわかる.

6.2.2 更新ステップ

$\boldsymbol{y}^t$ は $p(\boldsymbol{y}^t|\boldsymbol{f}_t) = \mathcal{N}(\boldsymbol{y}^t|\boldsymbol{f}_t, \sigma^2 \boldsymbol{I})$ であるから, $\boldsymbol{f}_t$ が与えられたときに $\boldsymbol{y}^t$ が $\boldsymbol{f}$ に独立である性質を用いることで $p(\boldsymbol{y}^t|\boldsymbol{f}, \boldsymbol{f}_t) = p(\boldsymbol{y}^t|\boldsymbol{f}_t)$ と表される. また, 予測ステップの同時分布は $p(\boldsymbol{f}, \boldsymbol{f}_t|\boldsymbol{y}^{1:t-1}) = p(\boldsymbol{f}_t|\boldsymbol{y}^{1:t-1}) \cdot p(\boldsymbol{f}|\boldsymbol{f}_t, \boldsymbol{y}^{1:t-1})$ と分割することができる. 更新ステップでは観測値 $\boldsymbol{y}^t$ が与えられているため, $p(\boldsymbol{y}^t|\boldsymbol{f}_t)$ と $p(\boldsymbol{f}_t|\boldsymbol{y}^{1:t-1})$ の同時分布はカルマンフィルターの更新則を用いることにより, 以下の平均値と共分散を持つガウス分布 $p(\boldsymbol{f}_t|\boldsymbol{y}^{1:t}) = \mathcal{N}(\boldsymbol{f}_t|\boldsymbol{\mu}_t^e, \boldsymbol{C}_t^e)$ に更新できる.

$$\begin{aligned} \boldsymbol{\mu}_t^e &= \boldsymbol{\mu}_t^p + \boldsymbol{G}_t(\boldsymbol{y}^t - \boldsymbol{\mu}_t^p) \\ \boldsymbol{C}_t^e &= \boldsymbol{C}_t^p - \boldsymbol{G}_t \boldsymbol{C}_t^p \end{aligned}$$

ここで, $\boldsymbol{G}_t = \boldsymbol{C}_t^p \cdot (\boldsymbol{C}_t^p + \sigma^2 \boldsymbol{I})^{-1}$ である.
最後に, $\boldsymbol{f}$ の事後分布は $p(\boldsymbol{f}_t|\boldsymbol{y}^{1:t})$ と $p(\boldsymbol{f}|\boldsymbol{f}_t, \boldsymbol{y}^{1:t-1})$ の同時分布を f_t について周辺化することより求められる. その同時分布は,

$$\boldsymbol{\mu}_t = \begin{bmatrix} \boldsymbol{\mu}_t^f \\ \boldsymbol{\mu}_t^e \end{bmatrix}, \ \boldsymbol{C}_t = \begin{bmatrix} \boldsymbol{C}_t^f & \boldsymbol{L}_t \boldsymbol{C}_t^e \\ \boldsymbol{C}_t^e \boldsymbol{L}_t^{\mathrm{T}} & \boldsymbol{C}_t^e \end{bmatrix} \tag{6.8}$$

となり, ここで

$$\begin{aligned}
\boldsymbol{\mu}_t^f &= \boldsymbol{\mu}_{t-1}^f + \tilde{\boldsymbol{G}}_t \cdot (\boldsymbol{y}^t - \boldsymbol{\mu}_t^p) && (6.9)\\
\boldsymbol{C}_t^f &= \boldsymbol{C}_{t-1}^f - \tilde{\boldsymbol{G}}_t \boldsymbol{J}_t \boldsymbol{C}_{t-1}^f && (6.10)\\
\tilde{\boldsymbol{G}}_t &= \boldsymbol{L}_t \cdot \boldsymbol{G}_t = \boldsymbol{C}_{t-1}^f \boldsymbol{J}_t^{\mathrm{T}} \cdot (\boldsymbol{C}_t^p + \sigma^2 \mathrm{I})^{-1} && (6.11)\\
\boldsymbol{L}_t &= \boldsymbol{C}_{t-1}^f \boldsymbol{J}_t^{\mathrm{T}} (\boldsymbol{C}_t^p)^{-1} && (6.12)
\end{aligned}$$

である. 以上をまとめると, 逐次ガウス過程のオンライン更新則は Algorithm 4 の 2 つの計算により行われる.

Algorithm 4 Recursive Gaussian Process

1: **Prediction**: ステップ $t-1$ の事後分布を用いて, ステップ t での予測分布 $p(\boldsymbol{y}^t|\boldsymbol{y}^{1:t-1}) = \mathcal{N}(\boldsymbol{y}^t|\boldsymbol{\mu}_t^p, \boldsymbol{C}_t^p + \sigma^2 \mathrm{I})$ を計算する.

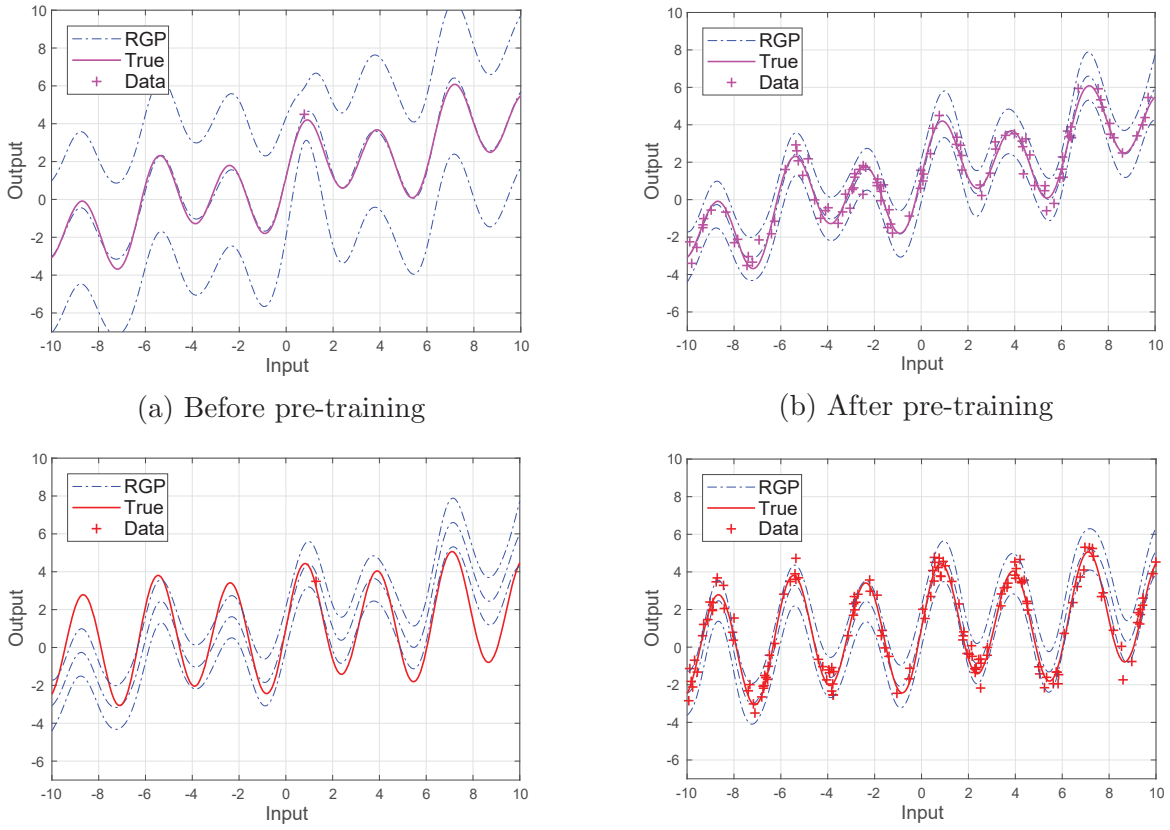
$$\begin{aligned}
\boldsymbol{\mu}_t^p &= \boldsymbol{J}_t \boldsymbol{\mu}_{t-1}^f \\
\boldsymbol{B}_t &= k(\boldsymbol{x}_t, \boldsymbol{x}_t) - \boldsymbol{J}_t k(\boldsymbol{X}, \boldsymbol{x}_t) \\
\boldsymbol{J}_t &= k(\boldsymbol{x}_t, \boldsymbol{X}) \cdot \boldsymbol{K}^{-1} \\
\boldsymbol{C}_t^p &= \boldsymbol{B}_t + \boldsymbol{J}_t \boldsymbol{C}_{t-1}^f \boldsymbol{J}_t^{\mathrm{T}}
\end{aligned}$$

2: **Update (RGP)**: 新たな観測値 $\boldsymbol{y}^t$ を用いて, f の事後分布を求める.

$$\begin{aligned}
\boldsymbol{\mu}_t^f &= \boldsymbol{\mu}_{t-1}^f + \boldsymbol{G}_t \cdot (\boldsymbol{y}^t - \boldsymbol{\mu}_t^p) \\
\boldsymbol{C}_t^f &= \boldsymbol{C}_{t-1}^f - \boldsymbol{G}_t \boldsymbol{J}_t \boldsymbol{C}_{t-1}^f \\
\boldsymbol{G}_t &= \boldsymbol{C}_{t-1}^f \boldsymbol{J}_t^{\mathrm{T}} \cdot (\boldsymbol{C}_t^p + \sigma^2 \mathrm{I})^{-1}
\end{aligned}$$

6.2.3 事前再学習

RGP は更新の中でハイパーパラメータの学習を行わないため, 事前に少量のデータから GP の学習に従ってハイパーパラメータを計算する. その後, RGP でオンライン



(a) Before pre-training

(b) After pre-training

(c) Before online-training

(d) After online-training

Fig 6.2: Examples of toy model training using RGP.

学習を行う前に, GP 学習に用いたデータを用いて事前に再学習を行うことで RGP の精度を向上させることができる. RGP は学習初期において分散値が大きくなり易い傾向があるため, この事前再学習は重要である.

Fig. 6.2 に RGP 学習の例を示す. 事前再学習を行うことで学習初期の精度が飛躍的に向上していることと, オンライン学習により真のモデルが変化した際においても学習モデルを更新できていることがわかる.

6.2.4 RGP 更新則とカルマンフィルターの関係

次のような状態 $\boldsymbol{\mu}^f$ を持つ状態方程式を考える.

$$\boldsymbol{\mu}_t^f = \boldsymbol{I}_n \boldsymbol{\mu}_{t-1}^f \tag{6.13}$$

$$\boldsymbol{y}_t = \boldsymbol{J}_t \boldsymbol{\mu}_t^f + \boldsymbol{w}_t \tag{6.14}$$

$$\boldsymbol{w}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{W}_t) \tag{6.15}$$

ただし $\boldsymbol{\mu}^f \in \mathbb{R}^{n_x}$, $\boldsymbol{y} \in \mathbb{R}^{n_t}$ であり、$\boldsymbol{W}_t = \boldsymbol{B} + \sigma^2 \boldsymbol{I}$ である.

上式の状態方程式は線形であり, カルマンフィルターを適用することができる．そのカルマンフィルターの更新式は以下のようになる．

$$\boldsymbol{\mu}_t^f = \boldsymbol{\mu}_{t-1}^f + \boldsymbol{L}_t \cdot (\boldsymbol{y}_t - \boldsymbol{J}_t \boldsymbol{\mu}_{t-1}^f) \tag{6.16}$$

$$\boldsymbol{C}_t^f = \boldsymbol{C}_{t-1}^f - \boldsymbol{L}_t \boldsymbol{J}_t \boldsymbol{C}_{t-1}^f \tag{6.17}$$

$$\boldsymbol{L}_t = \boldsymbol{C}_{t-1}^f \boldsymbol{J}_t^{\mathrm{T}} \cdot (\boldsymbol{J}_t \boldsymbol{C}_{t-1}^f \boldsymbol{J}_t^{\mathrm{T}} + \boldsymbol{W})^{-1} \tag{6.18}$$

ここで, $\boldsymbol{C}^f$ は状態の共分散行列である．このカルマンフィルター更新式は式 (6.9)-(6.12) と同様であることがわかる．また, カルマンフィルターの更新則は最適化問題として表すことができるため, 式 (6.15) に対する更新則は次のような最適化問題として表される．

$$\boldsymbol{\mu}_t^f = \underset{\boldsymbol{\mu}^f}{\operatorname{argmin}} \boldsymbol{w}_t^{\mathrm{T}} \boldsymbol{W}_t \boldsymbol{w}_t + (\boldsymbol{\mu}^f - \boldsymbol{\mu}_{t-1}^f)^{\mathrm{T}} \boldsymbol{C}_{t-1}^f (\boldsymbol{\mu}^f - \boldsymbol{\mu}_{t-1}^f) \tag{6.19}$$

$$\text{s.t. } \boldsymbol{y}_t = \boldsymbol{J}_t \boldsymbol{\mu}^f + \boldsymbol{w}_t \tag{6.20}$$

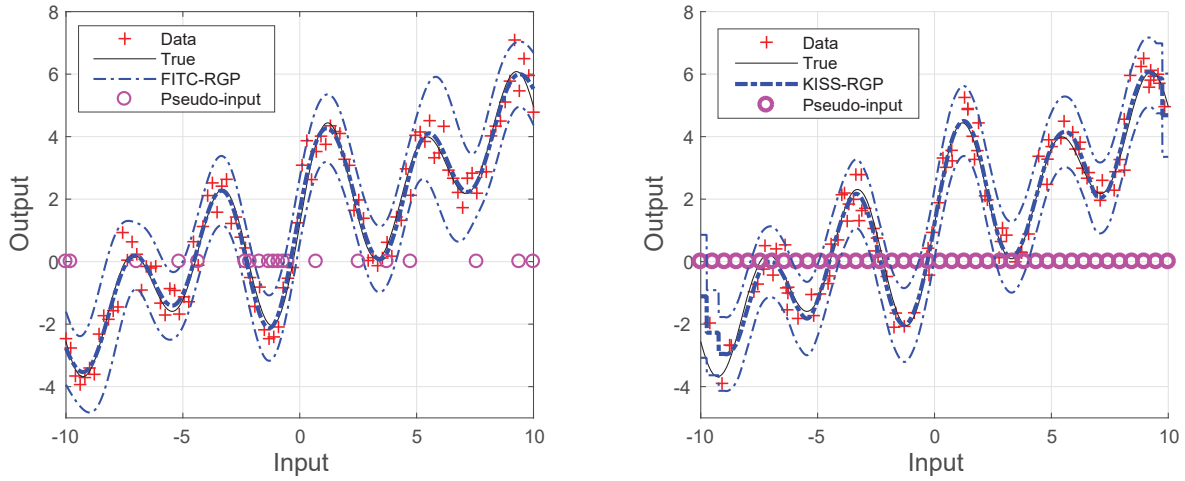
6.2.5 近似手法への応用

RGP は SoR や FITC といった疑似入力を用いた近似計算や SKI と容易に組み合わることができる. その場合, Algorithm 4 のカーネル関数を各々の近似手法で置き換えるだけである. RGP は基底データ点数を M, 学習データ点数を N とするとその計算コストは $O(M^2N)$ となるので, オンライン学習前の学習データ点数が十分にある場合, 疑似入力を用いることで基底ベクトル数を減らすことで学習や予測の計算コストを削減できる. RGP において基底データ点が重要になるが, 疑似入力をハイパーパラメータとして学習することで合理的で少数の基底データを決定したり, 学習データが少なかったり偏りがある場合には疑似入力を格子点上に配置するなど疑似入力の応用は特に有効である.

実際に Fig. 6.3 に近似手法に対して逐次学習アルゴリズムを適用した例を示す．この例は事前再学習の結果を示しているが, 通常の GP と同様に近似手法においても逐次学習が適用できることがわかる．また, FITC ベースの RGP では事前にハイパーパラメータを学習する際に, 疑似入力も最尤推定により求めており, 少量の基底データを用いた RGP を実現できている.

6.3 入力が確率分布を持つ逐次ガウス過程

本節では, モーメントマッチング (付録 D) を適用することで, 逐次ガウス過程を確率分布を持つ入力に対応させた予測・更新則を扱う．ここでは, 特に $n_t = 1$ の場合を考える.



(a) RGP+FITC　　(b) KISS-RGP

Fig 6.3: Examples of toy model training using RGP with approximation methods.

6.3.1　予測ステップ

予測分布を入力に対して期待値を取り, モーメントマッチングを用いる. ただし, $f_t = f(\boldsymbol{x}_t)$ とする.

$$\begin{aligned} p(y^t|\boldsymbol{y}^{1:t-1}) &= \int p(y^t|\boldsymbol{y}^{1:t-1}, \boldsymbol{x}_k^t) p(\boldsymbol{x}_t|\boldsymbol{y}^{1:t-1}) \mathrm{d}\boldsymbol{x}_t \\ &= \int p(y^t|f_t) \int \mathbb{E}_{\boldsymbol{x}_t|\boldsymbol{y}^{1:t-1}} \left[p(f_t|\boldsymbol{y}^{1:t-1}, \boldsymbol{x}_t) \right] \cdot p(\boldsymbol{f}|\boldsymbol{y}^{1:t-1}) \mathrm{d}\boldsymbol{f} \mathrm{d}f_t \end{aligned} \tag{6.21}$$

ここで, $\mathbb{E}_{\boldsymbol{x}_t|\boldsymbol{y}^{1:t-1}} \left[p(f_t|\boldsymbol{y}^{1:t-1}, \boldsymbol{x}_t) \right] \sim \mathcal{N}(\mu_t^p, B^t)$ とすると,

$$\mu_t^p = \boldsymbol{J}_t \boldsymbol{\mu}_{t-1}^f \tag{6.22}$$

$$B_t = \lambda^2 - \mathrm{tr}\left[(\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{Q} \right] + \sigma^2 + \boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{Q} \boldsymbol{\beta} - \mu_t^{p2} \tag{6.23}$$

$$\boldsymbol{J}_t = \boldsymbol{q}^{\mathrm{T}} \boldsymbol{K}^{-1} \tag{6.24}$$

$\boldsymbol{q}$ と $\boldsymbol{Q}$ は, それぞれ式 (D.6) と式 (D.9) より求められる. また, 式 (6.22)−(6.24) がそれぞれ式 (6.3)−(6.5) と同じ形式であることに注意すると, 直接式 (6.6) に適用することができる. したがって, 確率分布を持つ入力を考慮した RGP の予測分布は次式で得られる.

$$\begin{aligned} p(y^t|\boldsymbol{y}^{1:t-1}) &= \mathcal{N}(y^t|\mu_t^p, C^t + \sigma^2) \\ C^t &= B_t + \boldsymbol{J_t} C_{t-1}^f \boldsymbol{J_t}^{\mathrm{T}} \end{aligned} \tag{6.25}$$

6.3.2　更新ステップ

次に, 観測値 y_t が与えられたときの更新則を導出する. 式 (6.25) より $p(f_t|\boldsymbol{y}^{1:t-1}) = \mathcal{N}(\mu_t, C_t)$ であり, 4.1.2 節と同様に導出できる. ここで, $c_t = 1/p(y_t|\boldsymbol{y}^{1:t-1})$ は正規化定数

である.

$$
\begin{aligned}
p(\boldsymbol{f}|\boldsymbol{y}^{1:t-1}, y^t) &= \int p(\boldsymbol{f}, f_t|\boldsymbol{y}^{1:t-1}, y^t)\mathrm{d}f_t \\
&= \int c_t \cdot p(\boldsymbol{f}, f_t, y^t|\boldsymbol{y}^{1:t-1})\mathrm{d}f_t \\
&= \int c_t \cdot p(y^t|\boldsymbol{f}, f_t) \cdot p(\boldsymbol{f}, f_t|\boldsymbol{y}^{1:t-1})\mathrm{d}f_t \\
&= \int c_t \cdot p(y^t|f_t) \cdot \int p(f_t|\boldsymbol{f}, \boldsymbol{x}_t)\ p(\boldsymbol{x}^t|\boldsymbol{y}^{1:t-1})\mathrm{d}\boldsymbol{x}^t \cdot p(\boldsymbol{f}|\boldsymbol{y}^{1:t-1})\mathrm{d}f_t \\
&= \int c_t \cdot p(y^t|f_t) \cdot \underbrace{\mathbb{E}_{\boldsymbol{x}^t|\boldsymbol{y}^{1:t-1}}\left[p(f_t|\boldsymbol{y}^{1:t-1}, \boldsymbol{x}^t)\right]}_{\text{式 (6.22) } \sim \text{ (6.24)}} p(\boldsymbol{f}|\boldsymbol{y}^{1:t-1})\mathrm{d}f_t
\end{aligned}
$$

$p(\boldsymbol{f}|\boldsymbol{y}^{1:t-1}) \sim \mathcal{N}(\mu^f_{t-1}, C^f_{t-1})$ であるとすれば, 更新式は以下のようになる.

$$
\begin{aligned}
\boldsymbol{\mu}^f_t &= \boldsymbol{\mu}^f_{t-1} + \boldsymbol{G}_t \cdot (y^t - \mu^p_t) && (6.26)\\
\boldsymbol{C}^f_t &= \boldsymbol{C}^f_{t-1} - \boldsymbol{G}_t \boldsymbol{J}_t \boldsymbol{C}^f_{t-1} && (6.27)\\
\boldsymbol{G}_t &= \boldsymbol{C}^f_{t-1} {\boldsymbol{J}_t}^{\mathrm{T}} \cdot (C^t + \sigma^2)^{-1} && (6.28)
\end{aligned}
$$

以上をまとめると, 入力が確率分布を持つ場合の逐次ガウス過程更新則は Algorithm 5 となる.

6.4 ロバスト逐次ガウス過程

エンジンのセンサーによる観測値は外界の影響より外れ値と呼ばれる非ガウスノイズで観測値が汚されることが知られている. そこで, 外れ値に対してロバストな更新ができる RGP[84] を適用する. RRGP はロバストカルマンフィルター [85]-[87] と同様に RGP の更新を表す最適化問題へ $l1$ 正則化項を用いることにより外れ値を陽に推定するものである.

6.4.1 ロバストカルマンフィルター

外れ値を陽に扱うために, 式 (6.15) に外れ値として $\boldsymbol{z}_t$ を新たに定義する.

$$
\begin{aligned}
\boldsymbol{\mu}^f_t &= \boldsymbol{I}_n \boldsymbol{\mu}^f_{t-1} && (6.29)\\
\boldsymbol{y}_t &= \boldsymbol{J}_t \boldsymbol{\mu}^f_t + \boldsymbol{w}_t + \boldsymbol{z}_t && (6.30)\\
\boldsymbol{w}_t &\sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{W}_t) && (6.31)
\end{aligned}
$$

Algorithm 5 Recursive Gaussian Process with Uncertain Input

1: **Prediction**: ステップ $t-1$ の事後分布を用いて, ステップ t での予測分布 $p(\boldsymbol{y}^t|\boldsymbol{y}^{1:t-1}) = \mathcal{N}(\boldsymbol{y}^t|\mu_t^p, C^t + \sigma^2)$ を計算する.

$$
\begin{aligned}
\mu_t^p &= \boldsymbol{J}_t \boldsymbol{\mu}_{t-1}^f \\
B_t &= \lambda^2 - \mathrm{tr}\left[(\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1}\boldsymbol{Q}\right] + \sigma^2 + \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{\beta} - \mu_t^{p2} \\
\boldsymbol{J}_t &= \boldsymbol{q}^{\mathrm{T}}\boldsymbol{K}^{-1} \\
C^t &= B_t + \boldsymbol{J_t} C_{t-1}^f \boldsymbol{J_t}^{\mathrm{T}} \\
\boldsymbol{x}_t &= \mathcal{N}(\boldsymbol{x}_t|\tilde{\boldsymbol{\mu}}_t, \tilde{\boldsymbol{\Sigma}}_t) \\
\boldsymbol{\beta} &= (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1}\boldsymbol{y} \\
q_i &= \lambda^2 |\tilde{\boldsymbol{\Sigma}}_k \boldsymbol{\Lambda}^{-1} + \boldsymbol{I}|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}\boldsymbol{\nu}^{\mathrm{T}}(\tilde{\boldsymbol{\Sigma}}_t + \boldsymbol{\Lambda})^{-1}\boldsymbol{\nu}\right] \\
\boldsymbol{\nu}_i &= (\boldsymbol{x}_i - \tilde{\boldsymbol{\mu}}_t), \quad i = 1, \cdots, N \\
Q_{ij} &= |\boldsymbol{R}|^{-\frac{1}{2}} k(\boldsymbol{x}_i, \tilde{\boldsymbol{\mu}}_t) k(\boldsymbol{x}_j, \tilde{\boldsymbol{\mu}}_t) \exp\left[\frac{1}{2} \cdot z_{ij}^{\mathrm{T}}\, \boldsymbol{T}^{-1} z_{ij}\right] \\
\boldsymbol{R} &= 2\tilde{\boldsymbol{\Sigma}}_k \boldsymbol{\Lambda}^{-1} + \boldsymbol{I} \\
\boldsymbol{T} &= 2\boldsymbol{\Lambda}^{-1} + \tilde{\boldsymbol{\Sigma}}_t^{-1} \\
z_{ij} &= \boldsymbol{\Lambda}^{-1}(\boldsymbol{\nu}_i + \boldsymbol{\nu}_j)
\end{aligned}
$$

2: **Update (RGP)**: 新たな観測値 $\boldsymbol{y}^t$ を用いて, f の事後分布を求める.

$$
\begin{aligned}
\boldsymbol{\mu}_t^f &= \boldsymbol{\mu}_{t-1}^f + \boldsymbol{G}_t \cdot (y^t - \mu_t^p) \\
\boldsymbol{C}_t^f &= \boldsymbol{C}_{t-1}^f - \boldsymbol{G}_t \boldsymbol{J}_t \boldsymbol{C}_{t-1}^f \\
\boldsymbol{G}_t &= \boldsymbol{C}_{t-1}^f \boldsymbol{J}_t{}^{\mathrm{T}} \cdot (C^t + \sigma^2)^{-1}
\end{aligned}
$$

ロバストカルマンフィルターでは外れ値 $\boldsymbol{z}_t$ に対して l_1 正則化項を与えることにより, 外れ値を陽に推定することができる. 以下に式 (6.20) に外れ値の l_1 正則化項を導入した新たな最適化問題を示す.

$$\boldsymbol{\mu}_t^f, \hat{\boldsymbol{z}}_t = \underset{\boldsymbol{\mu}^f, \boldsymbol{z}_t}{\operatorname{argmin}} \boldsymbol{w}_t^{\mathrm{T}} \boldsymbol{W}_t \boldsymbol{w}_t + (\boldsymbol{\mu}^f - \boldsymbol{\mu}_{t-1}^f)^{\mathrm{T}} \boldsymbol{C}_{t-1}^f (\boldsymbol{\mu}^f - \boldsymbol{\mu}_{t-1}^f) + \boldsymbol{\lambda} ||\boldsymbol{z}_t||_1 \tag{6.32}$$

$$\text{s.t.} \quad \boldsymbol{y}_t = \boldsymbol{J}_t \boldsymbol{\mu}^f + \boldsymbol{w}_t + \boldsymbol{z}_t \tag{6.33}$$

ここで, $\boldsymbol{\lambda}$ は設計パラメータである. さらに, 上式は次のように変形でき, $\hat{\boldsymbol{z}}_t$ を以下の最適化問題で推定することができる.

$$\hat{\boldsymbol{z}}_t = \operatorname{argmin} (\boldsymbol{e}_t - \boldsymbol{z}_t)^{\mathrm{T}} \boldsymbol{\Sigma}_t (\boldsymbol{e}_t - \boldsymbol{z}_t) + \boldsymbol{\lambda} ||\boldsymbol{z}_t||_1 \tag{6.34}$$

ただし, $\boldsymbol{e}_t = \boldsymbol{y}_t - \boldsymbol{J}_t \boldsymbol{\mu}_{t-1}$, $\boldsymbol{\Sigma}_t = (\boldsymbol{J}_t \boldsymbol{C}_{t-1}^f \boldsymbol{J}_t^{\mathrm{T}} + \boldsymbol{B}_t + \sigma^2 \mathrm{I})^{-1}$ である. 特に単出力の場合, この最適解は解析的に求めることができ, 次で与えられる.

$$\hat{\boldsymbol{z}}_t = \begin{cases} \boldsymbol{e}_t - \frac{\boldsymbol{\lambda}}{2\boldsymbol{\Sigma}_t} & (\frac{\boldsymbol{\lambda}}{2\boldsymbol{\Sigma}_t} \leq \boldsymbol{e}_t) \\ \boldsymbol{0} & (-\frac{\boldsymbol{\lambda}}{2\boldsymbol{\Sigma}_t} \leq \boldsymbol{e}_t < \frac{\boldsymbol{\lambda}}{2\boldsymbol{\Sigma}_t}) \\ \boldsymbol{e}_t + \frac{\boldsymbol{\lambda}}{2\boldsymbol{\Sigma}_t} & (\boldsymbol{e}_t < -\frac{\boldsymbol{\lambda}}{2\boldsymbol{\Sigma}_t}) \end{cases} \tag{6.35}$$

6.4.2 外れ値を考慮した RGP 更新則

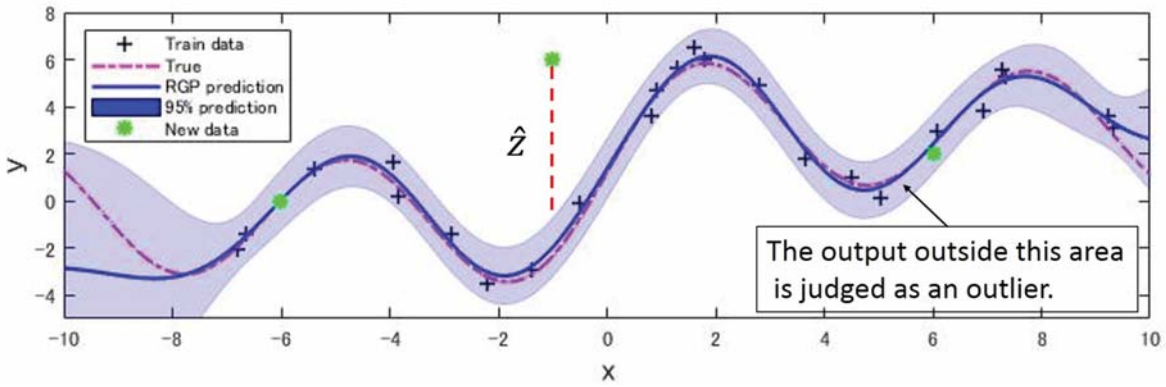


Fig 6.4: Outlier estimation using robust recursive Gaussian process regression.

ロバストカルマンフィルターでは設計パラメータ $\boldsymbol{\lambda}$ により外れ値の判定が行われるため, その設計法が問題となる. 提案する RRGP では RGP で予測される 95% 信頼区間から外れる場合には外れ値として判定するとし, $\boldsymbol{\lambda} = 4\sqrt{\boldsymbol{\Sigma}_t}$ としている (Fig 6.4). このとき RRGP の更新ステップは以下のようになる. ただし予測ステップは RGP と同様である.

Update (RRGP):

$$\boldsymbol{\mu}_t^f = \boldsymbol{\mu}_{t-1}^f + \boldsymbol{L}_t(\boldsymbol{e}_t - \hat{\boldsymbol{z}}_t) \tag{6.36}$$

$$\boldsymbol{e}_t = \boldsymbol{y}^t - \boldsymbol{J}_t\boldsymbol{\mu}_{t-1}^f \tag{6.37}$$

$$\boldsymbol{C}_t^f = (\boldsymbol{I} - \boldsymbol{L}_t\boldsymbol{J}_t)\boldsymbol{C}_{t-1}^f \tag{6.38}$$

$$\boldsymbol{L}_t = \boldsymbol{C}_{t-1}^f\boldsymbol{J}_t^{\mathrm{T}}(\boldsymbol{J}_t\boldsymbol{C}_{t-1}^f\boldsymbol{J}_t^{\mathrm{T}} + \boldsymbol{B}_t + \sigma^2\mathrm{I})^{-1} \tag{6.39}$$

$$\hat{\boldsymbol{z}}_t = \begin{cases} \boldsymbol{e}_t - \kappa_t & (\kappa_t \leq \boldsymbol{e}_t) \\ \boldsymbol{0} & (-\kappa_t \leq \boldsymbol{e}_t < \kappa_t) \\ \boldsymbol{e}_t + \kappa_t & (\boldsymbol{e}_t < -\kappa_t) \end{cases} \tag{6.40}$$

$$\kappa_t = 2(\boldsymbol{C}_t^p + \sigma^2\mathrm{I})^{\frac{1}{2}} \tag{6.41}$$

6.4.3 ハイパーパラメータの更新

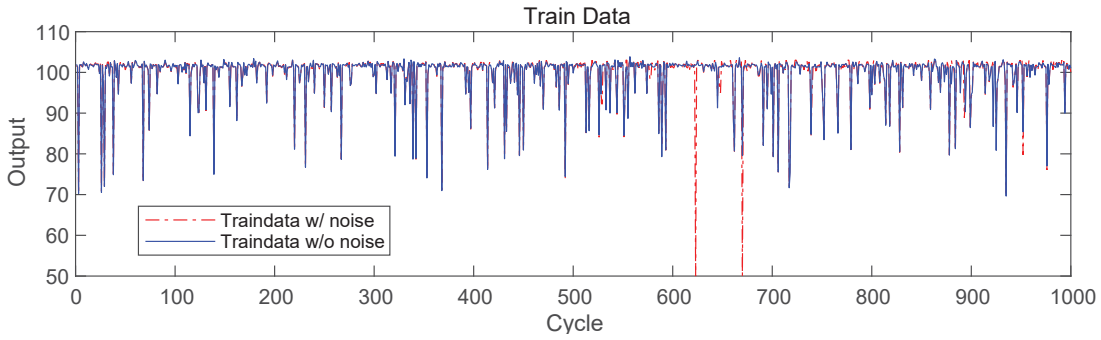
GP の性質はハイパーパラメータにより定まるが, 提案した RRGP はハイパーパラメータの更新は行っていないことに注意する. ハイパーパラメータも同時に更新する逐次アルゴリズム [84] も提案されているものの, 計算コストが大きくエンジンシステムのような更新周期が早いオンライン学習には適さないためである. つまり, この逐次学習の利点は, 計算コストが大きいハイパーパラエータの更新を避けながらも, 学習データを補正することにより, 予測精度を向上できる点にある.

一方で, ハイパーパラメータ更新を行なわない逐次学習には限度があるため, より正確なモデル学習のためにはハイパーパラメータ及び学習データの更新をオフラインで定期的に行うことが望ましい. この際, RRGP により外れ値を推定, 除去したデータを獲得することができるため, オフラインでの GP モデル学習に対する外れ値の影響を軽減できる.

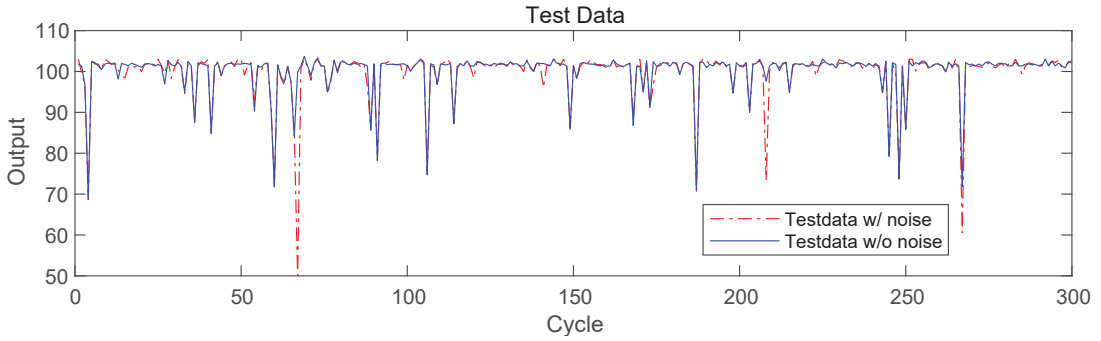
6.4.4 RRGP の適用例

最後に RRGP によるオンライン学習の有効性をエンジンシミュレータを用いて検証する. ここでは, 特に外れ値の影響が大きかった pm について扱う. センサー信号に Fig. 6.5 のように, 観測外乱としてガウス性のノイズと外れ値としてコーシー分布に従うノイズを加えた場合の学習を行う. ここで, 初期の学習データはエンジンテストベンチといった計測環境の良い場所で獲得するとして, ガウスノイズである観測外乱のみを加え上で, 5 章の Algorithm 3 を用いて獲得していることに注意する.

Fig. 6.5 の信号値を用いて学習を行った結果を Fig. 6.6 に示す. RRGP-NARX は 1 ステップ毎にオンライン更新を行い, GP-NARX は得られた 1000 点の学習データを用



(a) Train data (including initial training data)



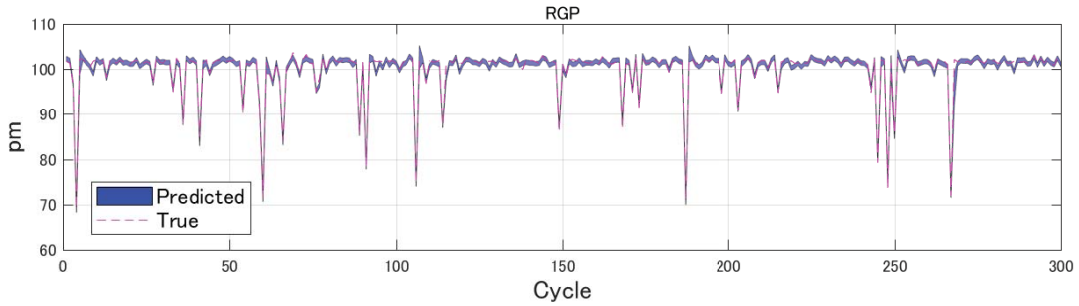
(b) Test data

Fig 6.5: Training and test data.

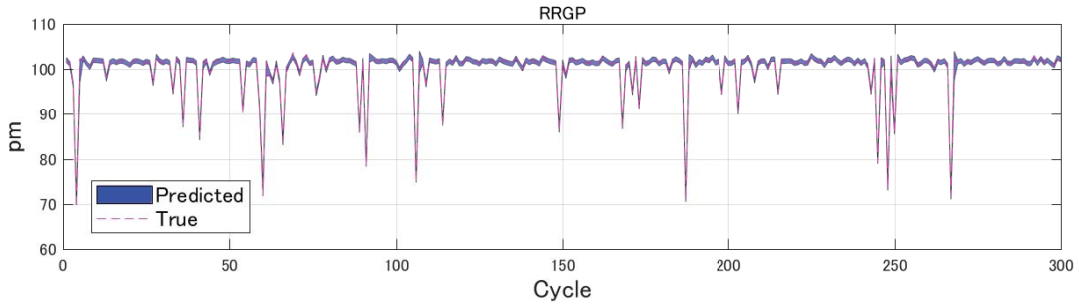
いて一度に学習を行っている．この結果より, GP の学習は外れ値の影響を受けやすく, 正確な予測ができなくなり, 予測分散も大きくなっていることが分かる．一方で, RRGP-NARX は外れ値に対して非常にロバストなモデルが構成できている．

Fig. 6.7 では学習データ数に対する精度の関係を示す．RGP 及び RRGP は 1 ステップ毎に, GP は 50 点毎に学習を行っている．まず, 外れ値を含む GP は予測精度が悪く, 学習データが増加しても精度が向上していない．対して, RRGP は学習データが増えるにつれ, 予測精度が向上する傾向が見られ, MSE では学習データ及び検証データに外れ値を含まない GP には及ばないものの, それに近い精度を実現できていることが分かる．入力の不確定性を考慮した RRGP-NARX の精度が悪くなっている理由は, 学習した GP のハイパーパラメータが観測ノイズからの入力 $\boldsymbol{x}$ への不確定性の伝播を陽に考慮したものでないためであると考えられる．別の言い方をするならば, ハイパーパラメータを学習した時点で入力の不確定性の影響を観測ノイズとして学習してしまっているため, 入力の不確定性を陽に考慮しながら予測を行うと, 2 重に不確定性を考慮していることになるためである．一方で, 入力の不確定性を考慮する手法は確率的モデル予測制御などで用いる長期予測には必須であり, ハイパーパラメータ学習時の工夫が必要となる [88].

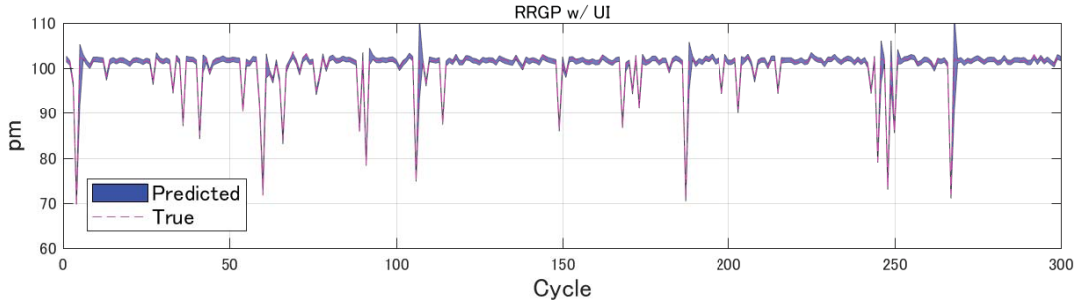
また, 初期学習データが 500 点の場合, 1 ステップ当たりの更新計算時間は Simulink



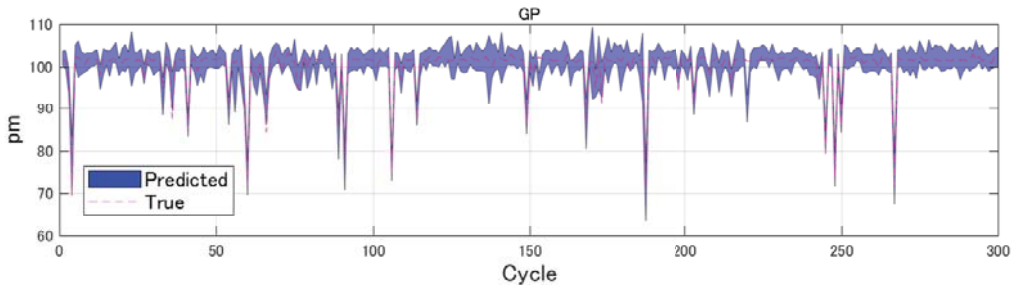
(a) RGP-NARX



(b) RRGP-NARX

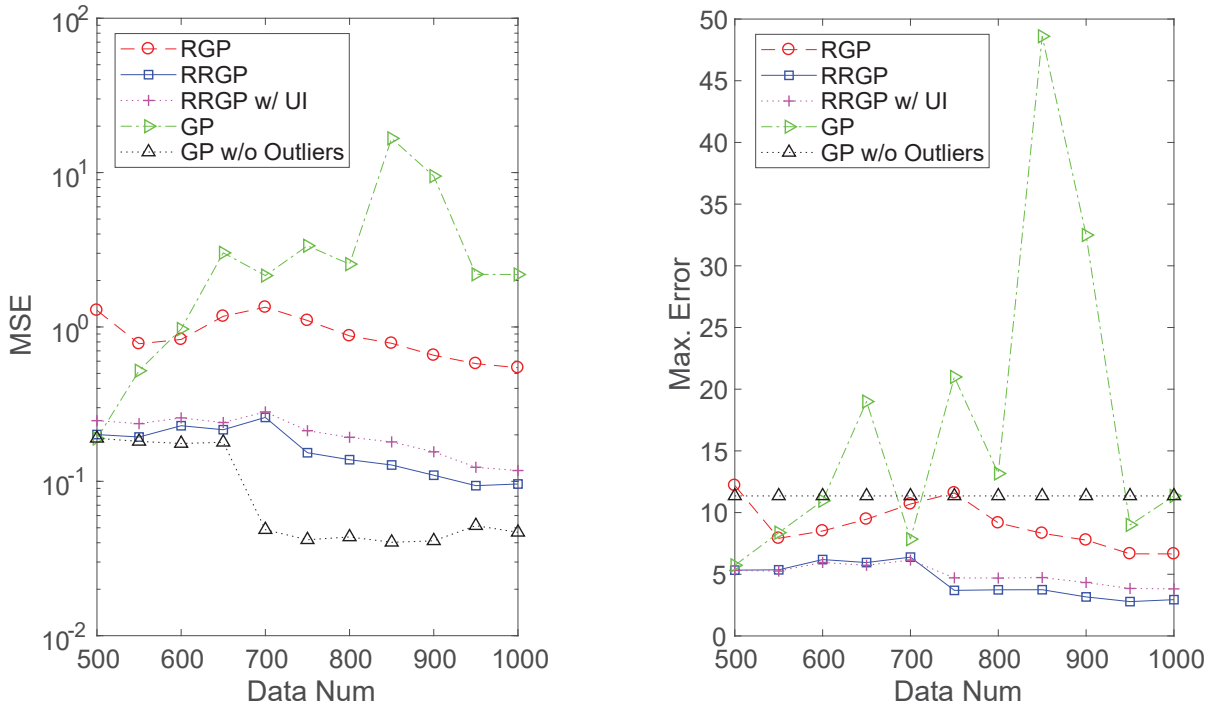


(c) RRGP with Uncertain Input-NARX



(d) GP-NARX

Fig 6.6: One-step ahead prediction value of *pm* when the output includes outliers (The number of training data points is 1000).



(a) MSE (b) Max. Error

Fig 6.7: Relationship between number of training data points and *pm* prediction accuracy.

R2017a (Core i7-4770S, RAM:16.0GB) において約 1[ms] となった. 更新に必要な計算時間は初期学習データ点数を m として, $O(m^2)$ のオーダーとなるが, 疑似入力を用いて計算コストを減らすことができ, 十分オンライン学習が可能であると言える.

6.5 おわりに

本章では自動車エンジンの動的モデルを構成する NARX-GP モデルのオンライン学習手法について述べた. オンライン学習として外れ値を陽に推定するロバストカルマンフィルタの考え方を RGP に応用した RRGP を提案し, その学習法として事前学習の必要性や近似計算手法の応用について述べた. その後エンジンシミュレータを用いて, 提案手法の有効性を検証した. 結果として, RRGP により外れ値の影響を受けにくい, オンライン学習モデルの構築が期待できることが分かった.

今後の課題としては基底ベクトル選定指標の提案と, 観測御ノイズから入力への不確定性の伝播を陽に考慮した GP 学習法の提案がある.

第7章　結論

本研究では自動車ガソリンエンジンシステムを対象とした入出力データを用いた統計モデルの学習法について議論を行った. 特に静的実験計画法, 動的実験計画法, 制御モデルの同定とオンライン学習という一連の制御モデル同定手法に対し, ガウス過程などを応用した新たな手法の提案とエンジンシミュレータを用いた有効性検証を行った. 本研究で得られた成果を以下にまとめる.

2 章では最初に本研究で扱うガウス過程及びその近似手法の概要をまとめた. その後, Society of Automotive Engineers と Society of Instrument and Control Engineers のジョイントベンチャーから提供されている境界近傍制御ベンチマーク問題について説明を行い, 本研究で学習を行う Nonlinear Auto-Regressive with eXogenous inputs (NARX) モデルをベースとした, 観測出力の動的モデルとトルク及び境界推定モデルを統合した制御モデルについて述べた.

3 章ではガウス過程を用いた新たな静的実験計画法を提案した. この手法は定常状態における非凸な境界モデルを学習するための入力列を効率的に獲得することができる. 具体的には, まず確率的モデルである期待値伝播法を用いたガウス過程分類器を応用することで生成モデルを生成し, そのモデルを用いて逐次的にシステムへ付加する入力を設計する. 提案手法の有用性はエンジンシミュレータにて検証した.

4 章では提案したエンジンシミュレータの制御モデルに対して, 深層学習とガウス過程及びその近似手法を用いた学習を行った. その中で, 学習データ数と予測・推定精度及び学習時間の関係を数値シミュレーションを用いて明確にすることで, モデル構築手法選択の指標を示した. 特にガウス過程の高速近似計算手法である完全独立学習条件近似とオンライン更新アルゴリズムである逐次ガウス過程を併用する手法は学習データ点数に対して柔軟であり, かつ学習コストが少なくなることを示した. また, 数値シミュレーションの結果を踏まえ, 深層学習の制御系応用に関しての考察を行った.

5 章では NARX モデルでエンジンの動的制御モデルを表現するとした上で, 異常運転の発生をおさえた入力列設計手法について議論した. 特にガウス過程モデルベースのオンライン動的実験計画法を扱い, 情報ゲインベースの入力列設計手法を提案した. エンジンシミュレータにおいて, 提案した入力列設計法により得られた学習データを用いることで, 精度の良い制御モデルを獲得できることを示した.

6 章では観測値に外れ値が混ざるような場合にも有効なガウス過程のオンライン学

習として, ロバスト逐次ガウス過程の適用を提案した. 統計的モデルにおいては外れ値がモデルの学習に大きな影響を及ぼすが, ロバスト逐次ガウス過程では逐次ガウス過程にロバストカルマンフィルターの考え方を適用しすることで外れ値を陽に推定し, 外れ値がモデル学習に与える影響を減らすことができる. その有効性はエンジンシミュレータにて検証した.

以上の研究結果より, ガウス過程をベースとした統計的制御モデルを用いた一連の同定手法を提案し, エンジンシミュレータによりその有効性を示した. 提案手法により獲得した統計的モデルは十分な正確性と簡易性を持ち, MILS/SILS/PILS/HILS といった開発工程において用いられるリアルタイムシミュレーションモデルとしての使用が期待できる. また, 統計的モデル予測制御をはじめとしたモデルベース制御への応用も期待できる.

数値シミュレータを用いた検証での課題としては, まず制御モデル構造の選択がある. 今回は NARX ベースのモデルを選択したが, モデルベース制御への適用という観点からはガウス過程状態空間モデルが実用的であるかもしれない. しかしながら, ガウス過程状態空間モデルは制御対象に対する事前知識がない場合, その正確な学習が容易ではなく, また誤学習が制御結果に与える影響も大きいという課題がある. そこで今後はガウス過程状態空間モデルの正確な同定手法を提案し, NARX ベースのモデルとの比較を行っていきたい. また, モデルベース制御においてガウス過程モデルを用いる場合では, 予測計算に必要なコストが実用化のボトルネックとなるため, 制御計算の並列高速化も課題となる. 更に, 深層学習を用いた制御モデルの簡単化や制御器への応用を考えていきたい. データ点数が多くない場合には深層学習によるモデルは良い結果を示さなかったものの, 情報の低次元化を行える点とその表現力は非常に魅力的である. 特に MILS などで制御モデルが与えられた場合には仮想環境で正確なデータが大量に獲得できるため, 深層学習の応用が望めると考えている.

本研究では一般に公開されているエンジンシミュレータを用いた検証結果のみを示しているが, 応用としては新たなデバイスを追加した次世代型エンジンへの適用を考えており, 境界近傍での同定はより難しい問題となる. 実機での検証もすすめているが, そのノイズや不確定性は必ずしも一定とならないため, 異分散ガウス過程をはじめとしたより複雑なモデルの導入など, 統計モデルの改良が望まれている. そして同時に, 統計モデルの学習及び予測高速化や入力列設計の効率化技術の発展も期待されている.

第8章　付録

A　数学的補足

A.1　ガウス分布の性質

ガウス分布は正規分布とも呼ばれ, 平均値 $\boldsymbol{\mu} \in \mathbb{R}^D$, 共分散行列 $\boldsymbol{\Sigma} \in \mathbb{R}^{D \times D}$ のガウス分布は次のように表される.

$$p(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})\right\} \tag{A.1}$$

ただし行列 $\boldsymbol{\Sigma}$ は正定かつ対称である. 表記を省略するために, この分布は $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ または $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ と記述される.

変数 $\boldsymbol{x}$ に対して, それを 2 つのグループ $\boldsymbol{x}_A, \boldsymbol{x}_B$ に分割したとき, $p(\boldsymbol{x}) = p(\boldsymbol{x}_A, \boldsymbol{x}_B)$ は同時確率と呼ばれる. このとき, $\boldsymbol{x}_A$ の周辺確率 $p(\boldsymbol{x}_A)$ は次で与えられる.

$$p(\boldsymbol{x}_A) = \int p(\boldsymbol{x}_A, \boldsymbol{x}_B) d\boldsymbol{x}_B \tag{A.2}$$

また, $p(\boldsymbol{x}_B) > 0$ のときの条件付き確率は次で定義され, 特に $p(\boldsymbol{x}_A|\boldsymbol{x}_B)$ と $p(\boldsymbol{x}_B|\boldsymbol{x}_A)$ の関係式はベイズの定理と呼ばれる.

$$p(\boldsymbol{x}_A|\boldsymbol{x}_B) = \frac{p(\boldsymbol{x}_A, \boldsymbol{x}_B)}{p(\boldsymbol{x}_B)} = \frac{p(\boldsymbol{x}_A) p(\boldsymbol{x}_B|\boldsymbol{x}_A)}{p(\boldsymbol{x}_B)} \tag{A.3}$$

次のような同時分布がガウス分布となるような変数 $\boldsymbol{x}_A, \boldsymbol{x}_B$ を考える.

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_A \\ \boldsymbol{x}_B \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_A \\ \boldsymbol{\mu}_B \end{bmatrix}, \begin{bmatrix} \boldsymbol{A} & \boldsymbol{C} \\ \boldsymbol{C}^T & \boldsymbol{B} \end{bmatrix} \right) \tag{A.4}$$

このとき $\boldsymbol{x}_A$ の周辺分布は $\boldsymbol{x}_A \sim \mathcal{N}(\boldsymbol{\mu}_A, \boldsymbol{A})$ となり, また $\boldsymbol{x}_B$ が与えられたときの $\boldsymbol{x}_A$ の条件分布は次で与えられる [20].

$$\boldsymbol{x}_A|\boldsymbol{x}_B \sim \mathcal{N}(\boldsymbol{\mu}_A + \boldsymbol{C}\boldsymbol{B}^{-1}(\boldsymbol{x}_B - \boldsymbol{\mu}_B), \boldsymbol{A} - \boldsymbol{C}\boldsymbol{B}^{-1}\boldsymbol{C}^T) \tag{A.5}$$

A.2　共分散行列の微分計算

本論文で用いた共分散行列 $\boldsymbol{K}(\theta)$ に関する演算をいくつか紹介する [89]. ここで θ はハイパーパラメータである.

$$\begin{aligned}
\frac{\partial \log |\boldsymbol{K}|}{\partial \boldsymbol{K}} &= \boldsymbol{K}^{-1} \\
\frac{\partial \log |\boldsymbol{K}|}{\partial \theta} &= \mathrm{tr}\left(\boldsymbol{K}^{-1}\frac{\partial \boldsymbol{K}}{\partial \theta}\right) \\
\frac{\partial \boldsymbol{a}^T\boldsymbol{K}^{-1}\boldsymbol{b}}{\partial \boldsymbol{K}} &= -\boldsymbol{K}^{-1}\boldsymbol{a}\boldsymbol{b}\boldsymbol{K}^{-1} \\
\frac{\partial \boldsymbol{K}^{-1}}{\partial \theta} &= -\boldsymbol{K}^{-1}\frac{\partial \boldsymbol{K}}{\partial \theta}\boldsymbol{K}^{-1}
\end{aligned} \tag{A.6}$$

A.3　区分行列の逆行列計算

次のような可逆区分行列 $\boldsymbol{A} \in \mathbb{R}^{D\times D}$ とその逆行列 $\boldsymbol{A}^{-1}$ を考える.

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{P} & \boldsymbol{Q} \\ \boldsymbol{R} & \boldsymbol{S} \end{pmatrix}, \quad \boldsymbol{A}^{-1} = \begin{pmatrix} \tilde{\boldsymbol{P}} & \tilde{\boldsymbol{Q}} \\ \tilde{\boldsymbol{R}} & \tilde{\boldsymbol{S}} \end{pmatrix} \tag{A.7}$$

このとき, $\boldsymbol{A}^{-1}$ の各ブロックは次で与えられる [20][90].

$$\begin{aligned}
\tilde{\boldsymbol{P}} &= \boldsymbol{P}^{-1} + \boldsymbol{P}^{-1}\boldsymbol{Q}\boldsymbol{M}\boldsymbol{R}\boldsymbol{P}^{-1} \\
\tilde{\boldsymbol{Q}} &= -\boldsymbol{P}^{-1}\boldsymbol{Q}\boldsymbol{M} \\
\tilde{\boldsymbol{R}} &= -\boldsymbol{M}\boldsymbol{R}\boldsymbol{P}^{-1} \\
\tilde{\boldsymbol{S}} &= \boldsymbol{M} \\
\boldsymbol{M} &= (\boldsymbol{S} - \boldsymbol{R}\boldsymbol{P}^{-1}\boldsymbol{Q})^{-1}
\end{aligned} \tag{A.8}$$

または,

$$\begin{aligned}
\tilde{\boldsymbol{P}} &= \boldsymbol{N} \\
\tilde{\boldsymbol{Q}} &= -\boldsymbol{N}\boldsymbol{Q}\boldsymbol{S}^{-1} \\
\tilde{\boldsymbol{R}} &= -\boldsymbol{S}^{-1}\boldsymbol{R}\boldsymbol{N} \\
\tilde{\boldsymbol{S}} &= \boldsymbol{S}^{-1} + \boldsymbol{S}^{-1}\boldsymbol{R}\boldsymbol{N}\boldsymbol{Q}\boldsymbol{S}^{-1} \\
\boldsymbol{N} &= (\boldsymbol{P} - \boldsymbol{Q}\boldsymbol{S}^{-1}\boldsymbol{R})^{-1}
\end{aligned} \tag{A.9}$$

B テプリッツ法

N 個の学習データが 1 次元に間隔 d で等間隔に配置されており, カーネル関数が式 (2.21) の性質を持つとする. このとき, テプリッツ法 [35][36] により $\boldsymbol{K}^{-1}\boldsymbol{y}$ の高速な計算が可能になる. ここではその概要を述べる.

カーネル関数を $k(x,x') = k(x-x') = k(d\tau) = k_\tau$ で表すとする. このとき共分散行列 $\boldsymbol{K} \in \mathbb{R}^{N\times N}$ は次の対称なテプリッツ行列で記述できる.

$$\boldsymbol{K} = \begin{bmatrix} k_0 & k_1 & \cdots & k_{N-2} & k_{N-1} \\ k_1 & k_0 & \cdots & k_{N-3} & k_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ k_{N-2} & k_{N-3} & \cdots & k_0 & k_1 \\ k_{N-1} & k_{N-2} & \cdots & k_1 & k_0 \end{bmatrix} \tag{B.1}$$

ここで次の巡回行列 $\boldsymbol{C} \in \mathbb{R}^{2(N-1)\times 2(N-1)}$ を考える.

$$\begin{aligned} \boldsymbol{C} &= \left[\begin{array}{ccccc|ccccc} k_0 & k_1 & \cdots & k_{N-2} & k_{N-1} & k_{N-2} & k_{N-3} & \cdots & k_2 & k_1 \\ k_1 & k_0 & \cdots & k_{N-3} & k_{N-2} & k_{N-1} & k_{N-2} & \cdots & k_3 & k_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ k_{N-2} & k_{N-3} & \cdots & k_0 & k_1 & k_2 & k_3 & \cdots & k_{N-2} & k_{N-1} \\ k_{N-1} & k_{N-2} & \cdots & k_1 & k_0 & k_1 & k_3 & \cdots & k_{N-3} & k_{N-2} \\ \hline k_{N-2} & k_{N-1} & \cdots & k_2 & k_1 & k_0 & k_1 & \cdots & k_{N-4} & k_{N-3} \\ k_{N-3} & k_{N-2} & \cdots & k_3 & k_2 & k_1 & k_0 & \cdots & k_{N-3} & k_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ k_2 & k_3 & \cdots & k_{N-2} & k_{N-3} & k_{N-4} & k_{N-3} & \cdots & k_0 & k_1 \\ k_1 & k_2 & \cdots & k_{N-1} & k_{N-2} & k_{N-3} & k_{N-2} & \cdots & k_1 & k_0 \end{array}\right] \\ &= \begin{bmatrix} \boldsymbol{K} & \boldsymbol{S} \\ \boldsymbol{S}^T & \boldsymbol{A} \end{bmatrix} \end{aligned} \tag{B.2}$$

以後 $\boldsymbol{C}$ の最初の列を $\boldsymbol{c} \in \mathbb{R}^{2(N-1)}$ とおく. 巡回行列の性質から $\boldsymbol{C}$ の固有値 λ 及び固有ベクトル $\boldsymbol{v}$ を求めることで, $\boldsymbol{K}^{-1}\boldsymbol{y}$ 及び $|\boldsymbol{C}|$ を効率的に求めることができる. このとき, ω_m は 1 の m 乗根として, 対応する固有値と固有ベクトルは次のように書ける [91].

$$\begin{aligned} \lambda_m &= \sum_{t=0}^{N-1} \boldsymbol{c}_t \omega_m^t = \mathrm{DFT}[\boldsymbol{c}_t] = \tilde{\boldsymbol{c}}_m \\ \boldsymbol{v}_m &= N^{-1/2}[1, \omega_m, \ldots, \omega_m^{N-1}]^T \\ \omega_m &= \exp\left(-\frac{2\pi i m}{N}\right) \\ m &= 0, \ldots, N-1 \end{aligned} \tag{B.3}$$

すなわち, 固有値は $\boldsymbol{c}$ の離散フーリエ変換 (Discrete Fourier Transform, DFT) を用いて表すことができ, 高速フーリエ変換により簡単に計算できる. また, 任意のベクトル $\boldsymbol{z}$ と $\mathrm{DFT}[\boldsymbol{z}_t] = \tilde{\boldsymbol{z}}_m$ に対して, 逆離散フーリエ変換 (Inverse DFT, IDFT) により

$$\begin{aligned}\boldsymbol{C}\boldsymbol{z} &= \frac{1}{N}\sum_m \exp\left(\frac{2\pi imt}{N}\right)\tilde{\boldsymbol{c}}_m\tilde{\boldsymbol{z}}_m \\ &= \mathrm{DFT}^{-1}[\tilde{\boldsymbol{c}}_m\tilde{\boldsymbol{z}}_m]\end{aligned} \tag{B.4}$$

で計算できる. 任意の $\boldsymbol{r} \in \mathbb{R}^{(N-1)}$ に対して $\boldsymbol{z} = [\boldsymbol{r}, \boldsymbol{0}]^T$ とすると,

$$\boldsymbol{C}\boldsymbol{z} = \boldsymbol{C}\begin{bmatrix}\boldsymbol{r}\\ \boldsymbol{0}\end{bmatrix} = \begin{bmatrix}\boldsymbol{K}\boldsymbol{r}\\ \boldsymbol{0}\end{bmatrix} \tag{B.5}$$

となる. したがって, $\boldsymbol{C}\boldsymbol{z}$ を扱うことで $\boldsymbol{K}\boldsymbol{r}$ を求めることができ, 結果として共役勾配法により $\boldsymbol{K}^{-1}\boldsymbol{y}$ を効率的に計算することができる. ここで, DFT と IDFT による演算により, $\boldsymbol{K}^{-1}\boldsymbol{y}$ は $O(N\log N)$ の計算コストで算出できる.

C　簡易エンジンシミュレータ

本論文で扱うエンジンシミュレータのようなエンジンモデルは ICEM(In-Cycle Engine Model) と呼ばれ, 各サイクルで気筒毎の筒内圧特性を予測するため, その予測計算が複雑になり制御問題を考えることが難しい. そこで簡易エンジンシミュレータ (Fig. C.1) では, エンジンの平均的な振る舞いを考える平均値エンジンモデル (Mean Value Engine Model, MVEL) を扱うことで制御系設計の検討を容易にしている.

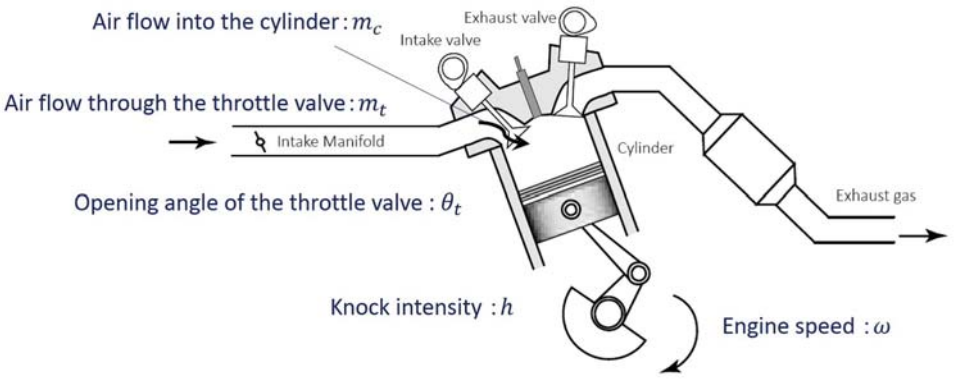


Fig C.1: Overview of simple engine model.

C.1 状態方程式

自然吸気の 4 気筒火花点火エンジンを仮定する. 吸気室の容積を V_m, 圧力を p_m, 密度を ρ_m, 温度を T_m とし, 同様に大気の圧力を p_a, 密度を ρ_a, 温度を T_a とする. またスロットル弁通過空気量を m_t, 筒内吸入空気量を m_c, 大気の自由度を f_m, 空気のガス定数を R_m とする. 導出は省略するが, 保存則と気体の状態方程式から次の状態方程式を得ることができる.

$$\frac{d}{dt}\begin{bmatrix} p_m/T_m \\ p_m \end{bmatrix} = \begin{bmatrix} \frac{R_m}{V_m}m_t - \frac{R_m}{V_m}m_c \\ \frac{f_m+2}{f_m}\frac{R_m}{V_m}T_a m_t - \frac{f_m+2}{f_m}\frac{R_m}{V_m}T_m m_c \end{bmatrix} \tag{C.1}$$

$$\begin{aligned} m_t &= d\{1-\cos(\theta_t)\}\left\{1-\left(\frac{p_m}{p_a}\right)^2\right\} \\ m_c &= \frac{p_m}{R_m T_m}V_h\frac{\omega}{\pi} \end{aligned} \tag{C.2}$$

ここで d はスリットルボア径, 大気圧力と密度で決まる定数であり, V_h は各気筒の排気量, ω はエンジン回転速度, θ_t はスロットル開度を示す. このモデルの特徴は m_t を単純な多項式計算で近似していることであり, これにより解析が行いやすくなると共に制御し易くなっている. さらに単純化することにより, 次のモデルを得る.

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \theta_1 x_1 + \theta_2(1-x_2^2)u \\ \theta_3 x_2 + \theta_4(1-x_2^2)u \end{bmatrix} \tag{C.3}$$

$$u = 1-\cos(\theta_t) \tag{C.4}$$

ここで $\theta_1, \theta_2, \theta_3, \theta_4$ は定数であり, $x_1 = p/T$, $x_2 = p$ としている. ただし $p = p_m/p_a$, $T = T_m/T_a$ とおく.

C.2 ノックモデル

ノック現象は物理モデルによる表現が難しいため, ここでは次のモデルを仮定する.

$$h = \left(\frac{p}{\alpha}\right)^2 + \left(\frac{T}{\beta}-1\right)^2 - 1 \tag{C.5}$$

ここで α, β は定数である. これはノックが温度と密接な関係があることから導出しているが, 物理的根拠はないことに注意する. ノックモデルでは $h > 0$ の場合にノックが発生しているとしているため, $h \leq 0$ を制約とした実験計画法及び制御を行う必要がある (C.2).

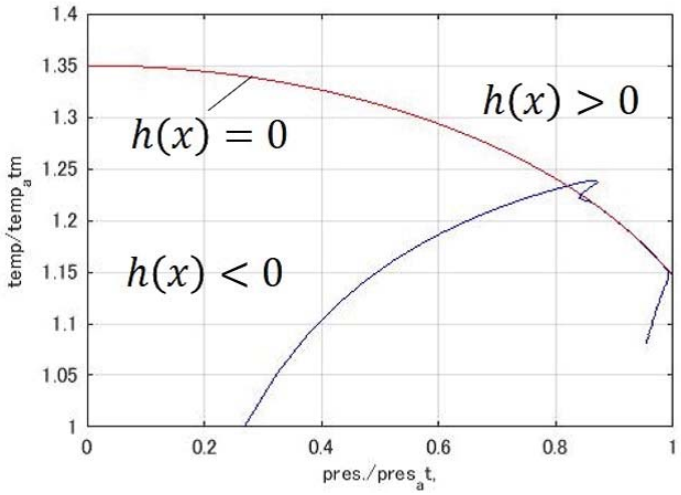


Fig C.2: Example of control result using discrete model predictive control. This indicates that knocking has occurred.

D　ガウス過程を用いた長期予測

長期予測 (Long-Term Prediction) では, 真値や観測値が得られない状態で予測値を用いて次の予測を行う問題である. 本節では, 期待値のみを用いて確定的な入力として次の予測値を計算するのではなく, 分布を持つ不確かな入力を扱う.

確定的な入力に対しては, ガウス過程による予測値はガウス分布として与えられるため, 次の予測ステップにおける入力も確率分布となる. 一般にガウス分布の入力を用いた出力予測値の確率分布はガウス分布にならないため, 解析的に予測分布を得ることができない. このような問題に対する解決法はサンプリング法, 線形近似法などが考えられる. しかし、サンプリング法はサンプリングの数が多くなるにつれて計算時間が膨大になる問題があり、線形近似法は正確な分散が得られない問題がある. そのため本論文ではモーメントマッチング [63][64] を用いて長期予測における予測分布を解析的にガウス分布へ近似することについて考える.

はじめに, 入力が $\boldsymbol{x}_k = \mathcal{N}(\boldsymbol{x}_k|\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k)$ に従うとする. このときの予測分布は

$$p(y_k) = \iint p(f(\boldsymbol{x}_k)|\boldsymbol{x}_k)p(\boldsymbol{x}_k)df d\boldsymbol{x}_k \tag{D.1}$$

と表される. モーメントマッチングを用いると, この予測分布の期待値 μ_k と分散 C_k を以下のように書くことができる.

$$\mu_k = \int f(\boldsymbol{x}_k) \cdot \int p(f(\boldsymbol{x}_k)|\boldsymbol{x}_k)p(\boldsymbol{x}_k)\mathrm{d}\boldsymbol{x}_k\mathrm{d}f \tag{D.2}$$

$$C_k = \int (f(\boldsymbol{x}_k) - \mu_k)^2 \int p(f(\boldsymbol{x}_k)|\boldsymbol{x}_k)p(\boldsymbol{x}_k)\mathrm{d}\boldsymbol{x}_k\mathrm{d}f \tag{D.3}$$

式 (D.2) は以下のように整理できる.

$$\begin{aligned} \mu_k &= \mathbb{E}_{\boldsymbol{x}_k}\left[\mathbb{E}_f\left[f(\boldsymbol{x}_k)|\boldsymbol{x}_k\right]\right] = \mathbb{E}_{\boldsymbol{x}_k}\left[\mu(\boldsymbol{x}_k)\right] \\ &= \boldsymbol{\beta}^{\mathrm{T}} \int k(\boldsymbol{X}, \boldsymbol{x}_k) p(\boldsymbol{x}_k) d\boldsymbol{x}_k = \boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{q} \qquad \text{(D.4)} \\ \boldsymbol{\beta} &= (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{y} \qquad \text{(D.5)} \end{aligned}$$

ここで, $\boldsymbol{q} = [q_1, \ldots, q_N]^{\mathrm{T}}$ である. 積分項について, カーネル関数が式 (4.1) であることに注意すると, $\mathcal{N}(\boldsymbol{x}_k|\boldsymbol{X}_i, \Lambda)$ と $\mathcal{N}(\boldsymbol{x}_k|\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k)$ の 2 つのガウス分布の積として計算できる.

$$\begin{aligned} q_i &= \lambda^2 |\tilde{\boldsymbol{\Sigma}}_k \boldsymbol{\Lambda}^{-1} + \boldsymbol{I}|^{-\frac{1}{2}} \exp\left[-\frac{1}{2} \boldsymbol{\nu}^{\mathrm{T}} (\tilde{\boldsymbol{\Sigma}}_k + \boldsymbol{\Lambda})^{-1} \boldsymbol{\nu}\right] \\ \boldsymbol{\nu}_i &= (\boldsymbol{x}_i - \tilde{\boldsymbol{\mu}}_k), \quad i = 1, \cdots, N \end{aligned} \qquad \text{(D.6)}$$

また, 式 (D.3) は次のように表せる.

$$C_k = \mathbb{E}_{\boldsymbol{x}_k}\left[\mathrm{var}_f\left[f(\boldsymbol{x}_k)|\boldsymbol{x}_k\right]\right] + \mathbb{E}_{f,\boldsymbol{x}_k}\left[f^2(\boldsymbol{x}_k)|\boldsymbol{x}_k\right] - \mu_k^2 \qquad \text{(D.7)}$$

$\mathbb{E}_{f,\boldsymbol{x}_k}\left[f^2(\boldsymbol{x}_k)|\boldsymbol{x}_k\right]$ について整理すると, 次のようになる.

$$\begin{aligned} \mathbb{E}_{f,\boldsymbol{x}_k}\left[f^2(\boldsymbol{x}_k)|\boldsymbol{x}_k\right] &= \mathbb{E}_{\boldsymbol{x}_k}\left[\mathbb{E}_f\left[f(\boldsymbol{x}_k)|\boldsymbol{x}_k\right] \cdot \mathbb{E}_f\left[f(\boldsymbol{x}_k)|\boldsymbol{x}_k\right]\right] \\ &= \int \mu(\boldsymbol{x}_k) \cdot \mu(\boldsymbol{x}_k) \cdot p(\boldsymbol{x}_k) d\boldsymbol{x}_k = \boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{Q} \boldsymbol{\beta} \qquad \text{(D.8)} \\ \boldsymbol{Q} &= \int k(\boldsymbol{X}, \boldsymbol{x}_k) k(\boldsymbol{x}_k, \boldsymbol{X}) p(\boldsymbol{x}_k) d\boldsymbol{x}_k \end{aligned}$$

ここで,

$$\begin{aligned} Q_{ij} &= |\boldsymbol{R}|^{-\frac{1}{2}} k(\boldsymbol{x}_i, \tilde{\boldsymbol{\mu}}_k) k(\boldsymbol{x}_j, \tilde{\boldsymbol{\mu}}_k) \exp\left[\frac{1}{2} \cdot z_{ij}^{\mathrm{T}}\, \boldsymbol{T}^{-1} z_{ij}\right] \\ \boldsymbol{R} &= 2\tilde{\boldsymbol{\Sigma}}_k \boldsymbol{\Lambda}^{-1} + \boldsymbol{I}, \quad \boldsymbol{T} = 2\boldsymbol{\Lambda}^{-1} + \tilde{\boldsymbol{\Sigma}}_k^{-1} \\ z_{ij} &= \boldsymbol{\Lambda}^{-1} (\boldsymbol{\nu}_i + \boldsymbol{\nu}_j) \end{aligned} \qquad \text{(D.9)}$$

また, $\mathbb{E}_{\boldsymbol{x}_k}\left[\mathrm{var}_f\left[f(\boldsymbol{x}_k)|\boldsymbol{x}_k\right]\right] = \lambda^2 - \mathrm{trace}\left[(\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{Q}\right] + \sigma^2$ となるため, 予測分散は次の式で求めることができる.

$$C_k = \lambda^2 - \mathrm{tr}\left[(\boldsymbol{K} + \sigma^2 \mathrm{I})^{-1} \boldsymbol{Q}\right] + \sigma^2 + \boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{Q} \boldsymbol{\beta} - \mu_k^2 \qquad \text{(D.10)}$$

参考文献

[1] A. Ohata : Control design in Model-Based Development and new SICE benchmark control design problem, ICROS-SICE International Joint Conference 2009 (2009)

[2] 申 鉄龍, 大畠 明 : 自動車エンジンのモデリングと制御, コロナ社 (2011)

[3] 大畠 明, 古田 勝久 : モデルベース開発のための複合物理領域モデリング, TechShare 社 (2012)

[4] 尾形 永 : 電子制御システム開発を加速する MBD, 計測と制御 47(3), 175/179 (2008)

[5] 嶋田 敏 : モデルベース開発に向けた JMAAB 活動紹介, 計測と制御 47(3), 198/201 (2008)

[6] 今田 道宏, 小森 賢 : エンジン制御システム開発技術, 計測と制御 53(8), 702/709 (2014)

[7] ETSS-JMAAB フォローアップ WG : ETSS-JMAAB 解説書 2015 年度版 Ver1.0, JMAAB ホームページ, available at http://jmaab.mathworks.jp/ (2015)

[8] K. Sata, S. Azuma and A. Ohata : A Parallelized Method for Continuous-Time Models with Dependence on Calculation Order, Proc. 8th IFAC International Symposium on Advances in Automotive Control, 697/702 (2016)

[9] 安井 裕司 : エンジン制御の最新動向, 計測と制御 53(8), 702/709 (2014)

[10] 川口 正浩, 山北 昌毅 : エンジンモデル同定における誤差モデルのカーネル同定法, 計測自動制御学会論文集 50(3), 311/317（2014）

[11] S. Watanabe and A. Ohata : Benchmark Problem for Near Boundary Operation Control for Automotive Engine, Proc. 6th Conference on Simulation and Testing for Automotive Electronics, 101/110 (2014)

[12] S. Watanabe and A. Ohata : Benchmark Problem for Near Boundary Operation Control for Automotive Engine, available online at $http://cig.ees.kyushu-u.ac.jp/benchmark\_JSAE\_SICE/$ (2015)

[13] J. M. Spelina, J. C. Peyton Jones and J. Frey : Characterization of knock intensity distributions: Part 1: Statistical independence and scalar measures, Proceeds of the

Institution of Mechanical Engineers Part D Journal of Automobile Engineering 228(2), 117/128 (2013)

[14] J. M. Spelina, J. C. Peyton Jones and J. Frey : Characterization of knock intensity distributions: Part 2: parametric models. Proceeds of the Institution of Mechanical Engineers Part D Journal of Automobile Engineering 227(12), 1650/1660 (2013)

[15] U. Kiencke and L. Nielsen : Automotive Control Systems for Engine, Driveline, and Vehicle. Springer (2005)

[16] J. C. Peyton Jones, J. M. Spelina and J. Frey : Likelihood-Based Control of Engine Knock, IEEE Ttransactions on Control Systems Technology, 21(6), 2169/2180 (2013)

[17] A. Thomasson, L. Eriksson, T. Lindell, J. C. Peyton Jones, J. Spelina and J. Frey : Tuning and experimental evaluation of a likelihood-based engine knock controller, Proc. 52nd IEEE Conference on Decision and Control, 6849/6854 (2013)

[18] J. Frey, J. C. Peyton Jones and S. Shayestehmanesh : Stochastic Simulation of a Cum-Sum Knock Controller, Proc. the 8th IFAC International Symposium on Advances in Automotive Control, 210/216 (2016)

[19] X. Shen and T. Shen : Knock Limit Controller based on Exponential Moving Average of Knock Intensity, Proc. the 8th IFAC International Symposium on Advances in Automotive Control, 691/695 (2016)

[20] C. E. Rasmussen and C. K. I. Williams : Gaussian Processes for Machine Learning, The MIT Press (2006)

[21] G. Gregor and G. Lightbody, Gaussian Processes for Modelling of Dynamic Nonlinear Systems, Proc. Irish Signals and System Conference, 141/147 (2002)

[22] J. Kocijan, A. Girard, B. Banko and R. Murray-Smith : Dynamic systems identification with Gaussian processes, Mathematical and Computer Modelling of Dynamical Systems 11(4), 411/424 (2005)

[23] B. Berger, F. Rauscher and B. Lohmann : Analysing Gaussian Processes for Stationary Black-Box Combustion Engine Modelling, Proc. the 18th IFAC World Congress, 10633/10640 (2011)

[24] T. Makowicki, M. Bitzer, P. Kotman and K. Graichen : A Combustion Cycle Model for Stationary and Transient Engine Operation, Proc. the 8th IFAC International Symposium on Advances in Automotive Control, 479/485 (2016)

[25] N. Tietze : Model-based Calibration of Engine Control Units Using Gaussian Process Regression, Ph.D. thesis, Technische Universitat Darmstadt (2015)

[26] C. E. Rasmussen and C. K. I. Williams ：Gaussian Processes for Machine Learning Software, *Online*, available at http://www.gaussianpro cess.org/gpml/code/matlab/doc/

[27] P.W. Goldberg, C.K.I. Williams, and C.M. Bishop : Regression with Input-dependent Noise: A Gaussian Process Treatment, Advances in Neural Information Processing Systems, 493/499 (1997)

[28] V. Tolvanen, P. Jylanki and A. Vehtari : Expectation propagation for nonstationary heteroscedastic Gaussian process regression, IEEE International Workshop on Machine Learning for Signal Processing (2014)

[29] Q. V. Le, A. J. Smola and S. Canu : Heteroscedastic Gaussian process regression, Proc. International Conference on Machine Learning, 489/496 (2005)

[30] K. Kersting, C. Plagemann, P. Pfaff and W. Burgard : Most likely heteroscedastic Gaussian process regression, Proc. International Conference on Machine Learning, 393/400 (2007)

[31] J. Quinonero-Candela and C. E. Rasmussen : A Unifying View of Sparse Approximate Gaussian Process Regression, Journal of Machine Learning Research (JMLR), 6 1939/1959 (2005)

[32] B. W. Silverman : Some Aspects of the Spline Smoothing Approach to Non-Parametric Regression Curve Fitting, Journal of the Royal Statistical Society. Series B (Methodological), 47(1), 1/52 (1985)

[33] M. Seeger, C. K. I. Williams and N. D. Lawrence : Fast forward selection to speed up sparse Gaussian process regression, Proc. the Ninth International Workshop on Artificial Intelligence and Statistics (2003)

[34] E. Snelson and G. Zoubin : Sparse Gaussian Processes using Pseudo-inputs, Proc. Advances in Neural Information Processing Systems (NIPS), 18, The MIT Press (2006)

[35] R. E. Turner : Statistical models for natural sounds, PhD thesis, University of London (2003)

[36] A. G. Wilson : Covariance Kernels for Fast Automatic Pattern Discovery and Extrapolation with Gaussian Processes, PhD thesis, University of Cambridge (2014)

[37] Y. Saatchi : Scalable Inference for Structured Gaussian Process Models, PhD thesis, University of Cambridge (2011)

[38] A. G. Wilson and H. Nickisch : Kernel interpolation for scalable structured Gaussian processes (KISS-GP), Proc. International Conference on Machine Learning (ICML) (2015)

[39] R. Keys : Cubic convolution interpolation for digital image processing, IEEE Transactions on Acoustics, Speech and Signal Processing, 29(6), 1153/1160 (1981)

[40] R. Salakhutdinov and G. E. Hinton : Using deep belief nets to learn covariance kernels for Gaussian processes, Proc. Advances in Neural Information Processing Systems (NIPS), (2007).

[41] A. G. Wilson, Z. Hu, R. Salakhutdinov and E. P. Xing : Deep Kernel Learning, Proc. Artificial Intelligence and Statistics (AISTATS) (2016)

[42] W. Huang, D. Zhao, F. Sun, H. Liu and E. Chang : Scalable gaussian process regression using deep neural networks, Proc. the 24th International Joint Conference on Artificial Intelligence (IJCAI), 3576/3582 (2015)

[43] N. Kieft : Evaluation of different design space description methods for analysing combustion engine operation limits, Leiden Institute of Advanced Computer Science (LIACS), Faculty of Science, Leiden University (2014)

[44] P. Renniger and M. Aleksandrov : Rapid hull determination: a new method to determine the design space for model based approaches, Design of Experiments (DoE) in Engine Development II. Expert-Verlag, Renningen (2005)

[45] F. Pukelsheim : Optimal Design of Experiments. Classics in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM) (2006)

[46] G. Kampmann, N. Kieft, and O. Nelles : Support Vector Machines for Design Space Exploration, Proc. the World Congress on Engineering and Computer Science 2012, Vol.2, 1116/1121 (2012)

[47] C. M. Bishop : Pattern recognition and machine learning, Springer-Verlag New York, LLC (2006)

[48] T. P. Minka : A family of algorithms for approximate Bayesian inference, Ph. D. thesis, MIT (2001)

[49] T. P. Minka and J. Lafferty : Expectation-Propagation for the Generative Aspect Model, Proc. 18th Conf. Uncertainty in Artificial Intelligence (UAI), 352/359 (2002)

[50] G. E. Hinton and R. Salakhutdinov : Reducing the Dimensionality of Data with Neural Networks, Science, 313, 504/507 (2006).

[51] G. E. Hinton, S. Osindero, and Y.W. Teh : A Fast Learning Algorithm for Deep Belief Nets, Neural Computation, 18(7), 1527/1554 (2006)

[52] Y. Bengio, P. Lamblin, D. Popovici and H. Larochelle : Greedy Layer-Wise Training of Deep Networks, Proc. Advances in Neural Information Processing Systems 19 (NIPS), 153/160 (2006).

[53] Y. Bengio : Learning deep architectures for AI, Foundations and Trends in Machine Learning, 2(1), 1/127 (2009)

[54] S. Hochreiter and J. Schmidhuber : Long Short-Term Memory, Neural computation, 9(8), 1735/1780 (1997)

[55] T. Lin, B. G. Horne, P. Tino and C. L. Giles : Learning Long-Term Dependencies in NARX Recurrent Neural Networks, IEEE Transactions on Neural Networks, 7(6), 1329/1338, (1996)

[56] D. Yua and L. Deng : Automatic Speech Recognition - A Deep Learning Approach, Springer (2014)

[57] N. D. Lawrence : Gaussian process latent variable models for visualisation of high dimensional data, Proc. Advances in Neural Information Processing Systems (NIPS), 16, 329/336 (2003)

[58] J. M. Wang, D. J. Fleet and A. Hertzmann : Gaussian Process Dynamical Models, Proc. Advances in Neural Information Processing Systems (NIPS), 18 (2005)

[59] J. M. Wang, D. J. Fleet and A. Hertzmann : Gaussian Process Dynamical Models for Human Motion, IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(2), 283/298 (2008)

[60] C. L. C. Mattos, Z. Dai, A. Damianou, J. Forth, G. A. Barreto and Neil D. Lawrence : Recurrent Gaussian Processes, Proc. International Conference on Learning Representations (ICLR) (2016)

[61] A. Damianou and N. D. Lawrence : Semi-described and semi-supervised learning with Gaussian processes, Proc. The 31st Conference on Uncertainty in Artificial Intelligence (UAI) (2015)

[62] R. Frigola, Y. Chen and C. E. Rasmussen : Variational Gaussian process state-space models, Proc. Advances in Neural Information Processing Systems (NIPS), 27, 3680/3688, (2014)

[63] M. P. Deisenroth, D. Fox and C. E. Rasmussen : Gaussian Processes for Data-Efficient Learning in Robotics and Control, IEEE Transactions on Pattern Analysis and Machine Intelligence 37(2) (2015)

[64] A. Girard and R. Murray-Smith : Gaussian Processes: Prediction at a Noisy Input and Application to Iterative Multiple-Step Ahead Forecasting of Time-Series, Switching and Learning, Lecture Notes in Computer Science (LNCS), 3355, 158/184 (2005)

[65] M. F. Huber : Recursive Gaussian process regression, Proc. the 38th International Conference on Acoustics, Sound, and Signal Processing (ICASSP), 3362/3366 (2013)

[66] M. F. Huber : Recursive Gaussian Process: On-line regression and learning, Pattern Recognition Letters 45, 85/91 (2014)

[67] M. A. Keyvanrad and M. M. Homayounpour : A brief survey on deep belief networks and introducing a new object oriented toolbox (DeeBNet), arXiv:1408.3264, (2014)

[68] H. H. Asada, F. Wu, A. Girard, and M. Mayalu : A Data-Driven Approach to Precise Linearization of Nonlinear Dynamical Systems in Augmented Latent Space, Proc. American Control Conference (ACC) 2016, 1838/1844 (2016)

[69] P. V, Overschee and B. D. Moor : Subspace Identification of Linear Systems: Theory, Implementation, Applications, Springer Publishing (1996)

[70] P. V. Overschee and B. D. Moor : N4SID: Subspace Algorithms for the Identification of Combined Deterministic Stochastic Systems, Automatica, 31(1), 75/93 (1994)

[71] M. J. Beal : Variational Algorithms for Approximate Bayesian Inference, PhD. Thesis, University College London (2003)

[72] T. Ohtsuka : Model structure simplification of Nonlinear Systems via immersion, IEEE Transactions on Automatic Control, 50(5) (2005)

[73] M. Deflorian, K. Florian and J. Ruckert : Online dynamic black box modelling and adaptive experiment design in combustion engine calibration, Proc. 6th IFAC International Symposium on Advances in Automotive Control, 703/708 (2010)

[74] M. Deflorian and S. Zaglauer : Design of Experiments for nonlinear dynamic system identification, Proc. the 18th IFAC World Congress, 13179/13184 (2011)

[75] R. Pintelon and J. Schoukens : System Identification A Frequency Domain Approach, Wiley-IEEE Press (2012)

[76] O. Nelles : Nonlinear System Identification : From Classical Approaches to Neural Networks and Fuzzy Models, Springer-Verlag Berlin Heidelberg (2001)

[77] M. Kawaguchi and M Yamakita : Identification with Physical and Error Models for Internal Combustion Engine, Proc. 7th IFAC Symposium on Advances in Automotive Control 780/785 (2013)

[78] H. Ishiyama and M. Yamakita : A new D-optimal input sequence design using PI^2 algorithm, Proc. 8th IFAC International Symposium on Advances in Automotive Control 795/800 (2016)

[79] T. M. Cover and J. A. Thomas : Elements of Information Theory, Wiley-Interscience (1991)

[80] N. Srinivas, A. Krause, S. Kakade, and M. Seeger : Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design, Proc. the International Conference on Machine Learning (2010)

[81] A. Krause and C. Guestrin : Near-optimal nonmyopic value of information in graphical models, Proc. Uncertainty in Artificial Intelligence (UAI) (2005)

[82] A. Ohata : Automotive Engine Control with Rational Function Satisfying inequality Constraint, Proc. 8th IFAC International Symposium on Advances in Automotive Control, 685/690 (2016)

[83] A. Ohata : Benchmark Problems on Automotive Engine Controls, 第 59 回 自動制御連合講演会 予稿集, 801/806, (2016)

[84] K. Seo and M. Yamakita : Nonlinear Time-Varying System Identification with Recursive Gaussian Process, Proc. 2017 American Control Conference, 825/830 (2017)

[85] J. Mattingley and S. Boyd : Real-Time Convex Optimization in Signal Processing, IEEE Signal Processing Magazine, 27(3), 50/61 (2010)

[86] D. Y. Kim, S. G. Lee and M. Jeon : Outlier Rejection Methods for Robust Kalman Filtering, Proc. 6th International Conference on Future Information Technology, 316/322 (2011)

[87] Y. Kaneda, Y. Irizuki and M. Yamakita : Fast Algorithm of Robust Kalman Filter via l1 regression by a Closed Form Solution, Proc. 19th World Congress The International Federation of Automatic Control (IFAC), 6153/6158 (2014)

[88] A. McHutchon and C. E. Rasmussen : Gaussian Process Training with Input Noise, Proc. Advances in Neural Information Processing Systems (NIPS), 24 (2011)

[89] K. B. Petersen and M. S. Pedersen : The Matrix Cookbook, Version: November 15 (2012)

[90] H. V. Henderson and S. R. Searle : On Deriving the Inverse of a Sum of Matrices, SIAM Review, 23(1), 53/60 (1981)

[91] R. M. Gray : Toeplitz and Circulant Matrices: A Review, Foundations and Trends in Communications and Information Theorym, 2(3), 155/239 (2006)

研究業績

研究業績 (発表順に掲載)

学術雑誌論文

- 大山 博之，山北 昌毅：エンジンベンチマークシミュレータを用いた制御モデル学習手法の検証，日本シミュレーション学会論文誌，9(3)，47/55 (2017)（2 章と 4.1-4.4 節に関連）

国際会議発表

- K. Odachi, H. Oyama and M. Yamakita : Nonlinear Model Predictive Control of SI Engine using Discrete Time Identification Model, Chinese Control Conference and SICE Annual Conference (2015)（関連なし）
- H. Oyama and M. Yamakita : A Nonlinear MPC Based on Application of a Sequential Distributed Accelerated Gradient Method to the Continuation Method, Proc. the 10th Asian Control Conference (2015)（関連なし）
- H. Oyama and M. Yamakita : Speed Control of Vehicles with Variable Valve Lift Engine by NMPC Based on Application of SDAGM with Continuation Method, Proc. 2015 IEEE Conference on Control Applications Part of 2015 IEEE Multi-Conference on Systems and Control (2015)（関連なし）
- H. Oyama, M. Yamakita, K. Sata and A. Ohata : Identification of Static Boundary Model Based on Gaussian Process Classification, Proc. the 8th IFAC Symposium on Advances in Automotive Control 801/806 (2016)（3 章に関連）
- H. Oyama, M. Yamakita and H. H. Asada : Stochastic Model Predictive Control Based on Data-Driven Linearization of Nonlinear System in Augmented Latent Space, Proc. the SICE Annual Conference (2016) (Position paper)（4.5 節に関連）
- H. Oyama, M. Yamakita and H. H. Asada : Approximated Stochastic Model Predictive Control using Statistical Linearization of Nonlinear Dynamical System in Latent Space, Proc. the 55th Conference on Decision and Control 7153/7158 (2016)（4.5 節に関連）

- H. Oyama and M. Yamakita : Online Learning of Automotive Gasoline Engine Model Using Robust Recursive Gaussian Process, The 2018 American Control Conference (査読済)（5 章と 6 章に関連）

国際会議ポスター発表

- H. Oyama, M. Yamakita and H. H. Asada : Stochastic Model Predictive Control Based on Data-Driven Linearization of Nonlinear System in Augmented Latent Space, The SICE Annual Conference (2016) （4.5 節に関連）

査読なし国内会議発表

- 大山 博之, 山北 昌毅 : 連続変形法へ逐次分散加速勾配法を応用した非線形モデル予測制御, 第 2 回 制御部門マルチシンポジウム 予稿集 (2015) （関連なし）
- 大山 博之, 山北 昌毅, 佐多 宏太, 大畠 明 : ガウス過程を応用したシステムの境界モデリング, 第 3 回 制御部門マルチシンポジウム 予稿集 (2016) （3 章に関連）
- 大山 博之, 山北 昌毅 : ロバスト逐次ガウス過程を用いた自動車エンジンモデルの学習, 第 60 回 自動制御連合講演会 予稿集 (2017) （5 章と 6 章に関連）

謝辞

本論文は東京工業大学理工学研究科機械制御システム専攻 山北研究室とトヨタ自動車株式会社との共同研究の成果もとに作成されており, 本研究を遂行するにあたり，様々な方々からのご助言やご指導などを賜りました．

特に指導教員である山北昌毅准教授には，学部時代から博士課程まで長い間お世話になりました. その間, 数学や制御，論文の執筆方法, 研究の進め方など研究者としての基盤となる様々なご指導を賜りました．山北准教授のもとで学び, そして研究を行えたことを心から嬉しく, そして光栄に思います. 誠にありがとうございました．

トヨタ自動車株式会社東富士研究所の皆様には共同研究に加え, 研究生として長期インターンシップの受け入れもしていただき, 大変お世話になりました. 特に大畠様, 佐多様, 松永様, 加古様にはエンジンへの応用研究を進める上で様々な面でご指導・御鞭撻を賜りました．上智大学の申鉄龍教授からはエンジン実機のデータやノウハウをいただき, 大変感謝しております.

また, マサチューセッツ工科大学浅田春比古教授とは, 共同研究という形で研究成果を発表できたことを嬉しく思います. ありがとうございました．

タイ国立科学技術開発庁 Manida Tongroon 博士のもとでは短期留学の間, ディーゼルエンジンについて学ばせていただきました. 受け入れて下さったこと, またご指導いただいたことを感謝しております.

カンボジア工科大学 Srang Sarot 博士, Lychek Keo 博士とは UAV 制御の共同研究ができたことを嬉しく思っております. また, カンボジアにもお招きいただきありがとうございました.

山北研究室でのエンジン制御研究を進めるにあたり, 大立さん, 石山さん, 徐さん, 遠藤さん, 高野さん, 井川さんとは共同研究者として互いに研究方針や内容を相談することができ, 結果として研究を順調に進めることができたと思います. ありがとうございました. 山北研究室の秘書である小林さんには，事務手続き関連だけではなく，プライベートでのアドバイスも多くいただきました. また, 山北准教授の奥様には人生の先輩として色々なご指導していただき, お世話になりました，

最後に，博士進学を後押してくれた両親と妻に深く感謝いたします．