# MATLAB EXPO 2024
# MBD 04 データ駆動モデリングと制御

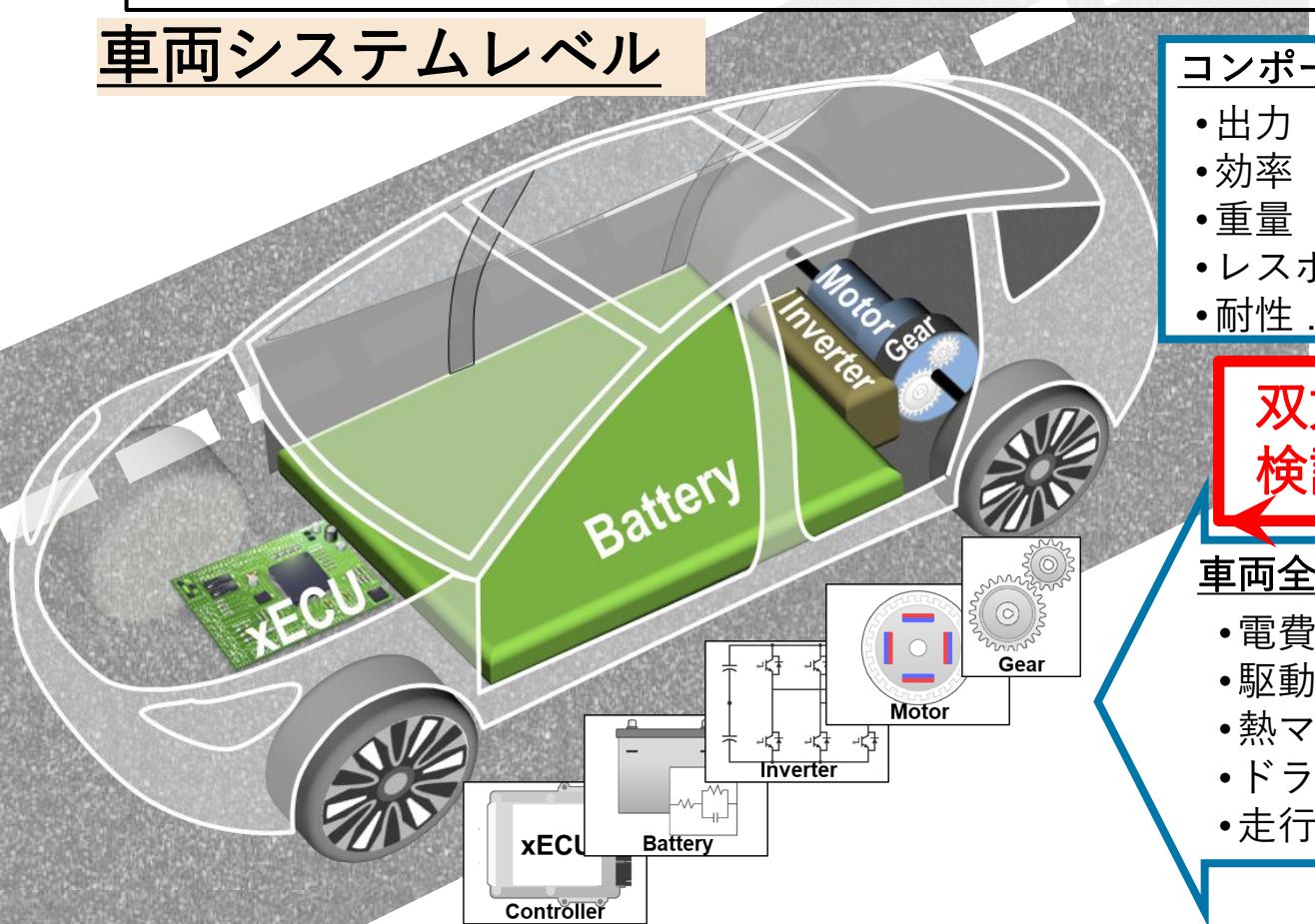
MathWorks Japan
アプリケーションエンジニアリング部

# AIを用いたサロゲートモデリング

# はじめに
# ますます求められるシステムレベルシミュレーション

**車両システムレベルシミュレーションにより、以下が可能となります。**
- コンポーネント性能向上がもたらす車体全体への効果検証
- 車体全体から見たコンポーネントへの要求の確認

## 車両システムレベル

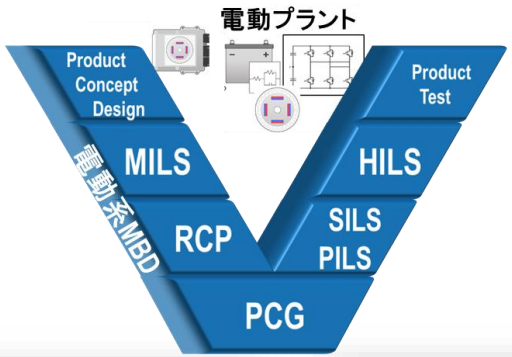


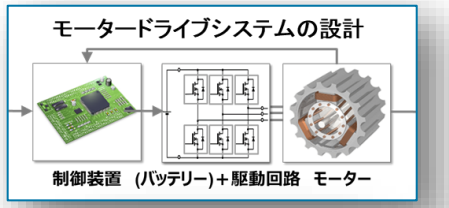

### コンポーネントへの要求
- 出力
- 効率
- 重量
- レスポンス
- 耐性 …etc.

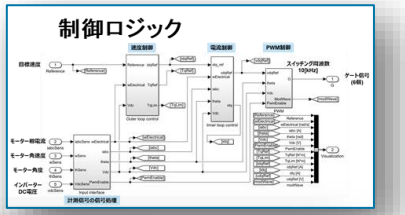
**双方向での検証が必要**

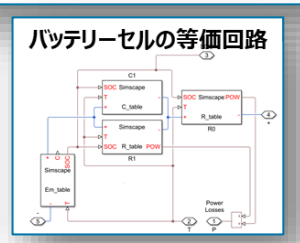
### 車両全体での効果検証
- 電費・エネマネ
- 駆動系振動
- 熱マネ
- ドライバビリティ
- 走行性能 …etc.

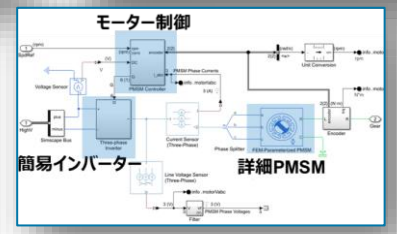
## コンポーネントレベル（例：電動化）



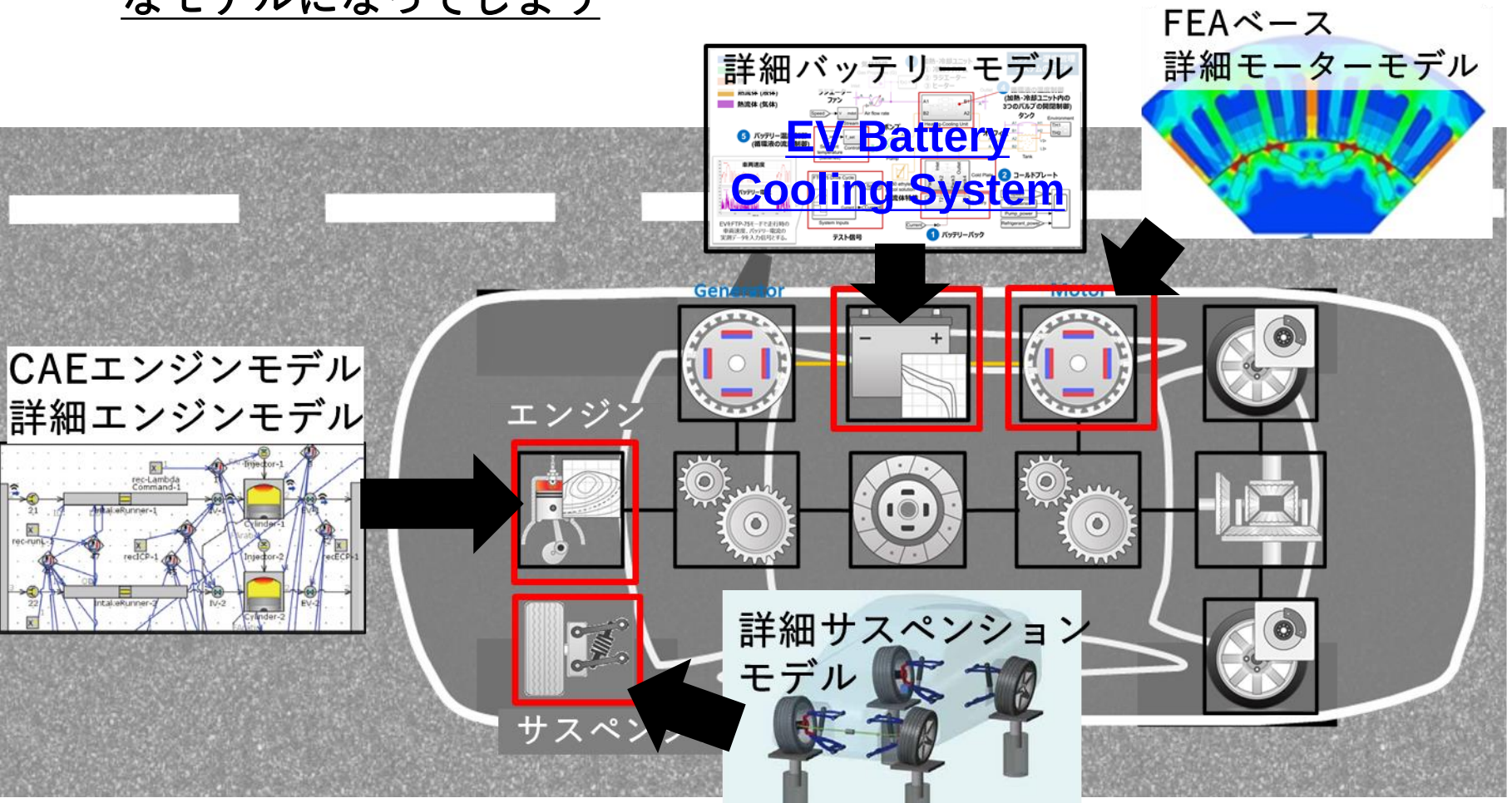



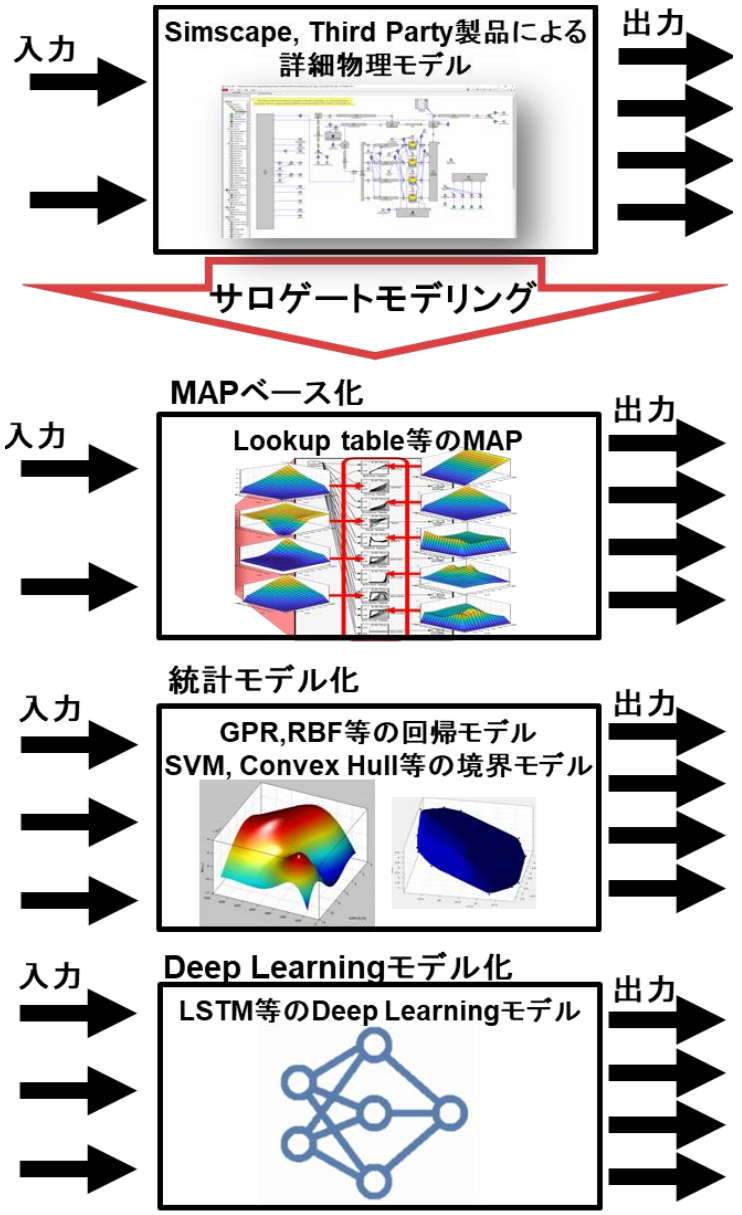



# 次数低減(Reduced Order Modeling)
## 機械学習・深層学習を利用したサロゲートモデリング

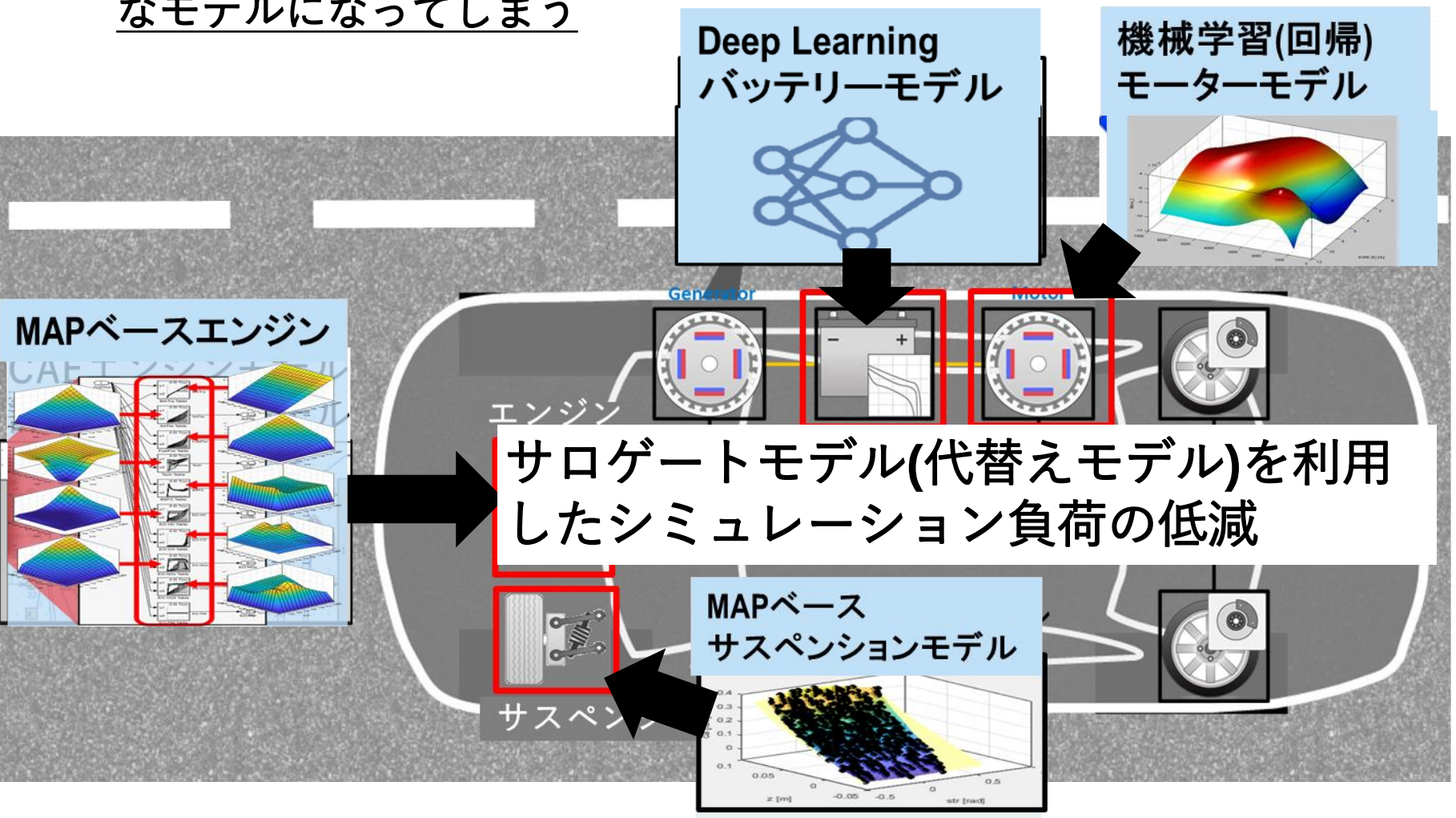
⇒ モデル構築時、既存の**各コンポーネントモデルを接続しただけでは高負荷なモデルになってしまう**

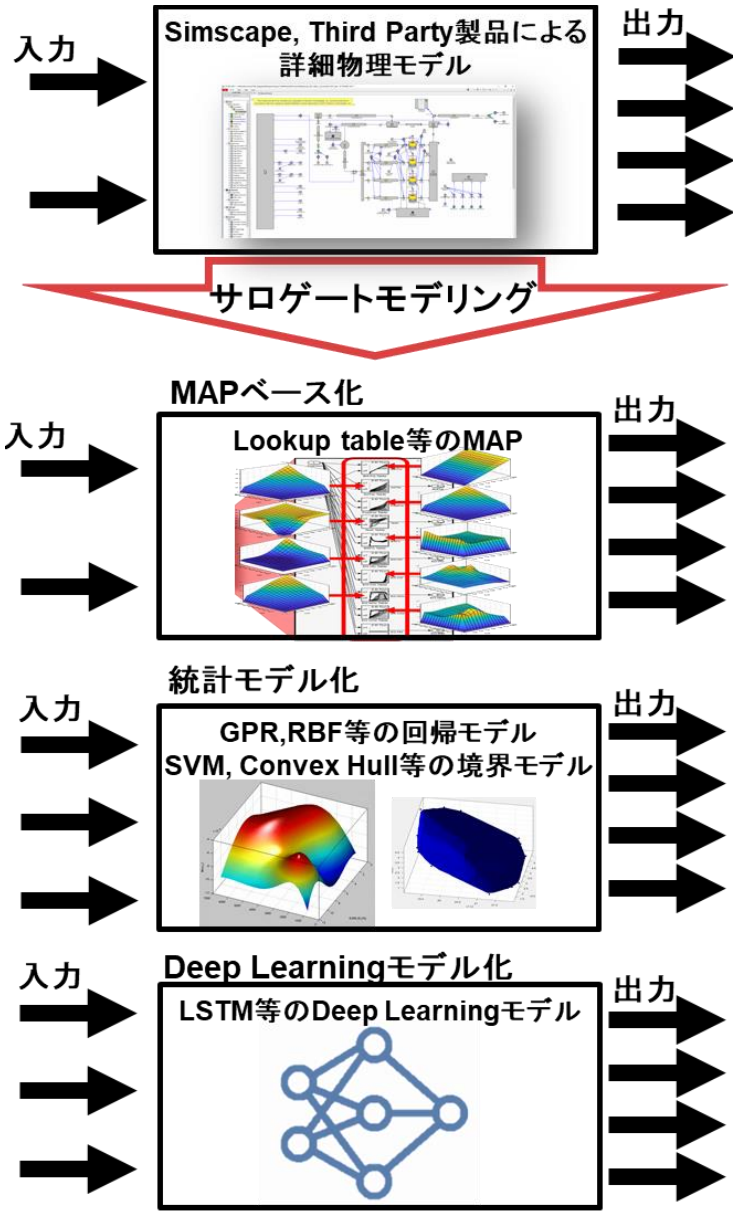


# 次数低減(Reduced Order Modeling)
## 機械学習・深層学習を利用したサロゲートモデリング

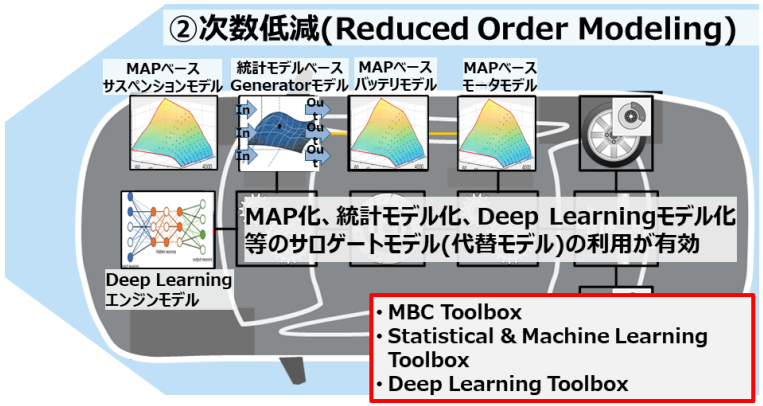
⇒ モデル構築時、既存の<u>各コンポーモデルを接続しただけでは高負荷なモデルになってしまう</u>



# 詳細物理モデルのサロゲートモデリング 次数低減(Reduced Order Modeling)

- 詳細物理モデルに対して以下のサロゲートモデル(代替モデル)を作成しシミュレーション負荷低減へ




**<u>モデルタイプ</u>**

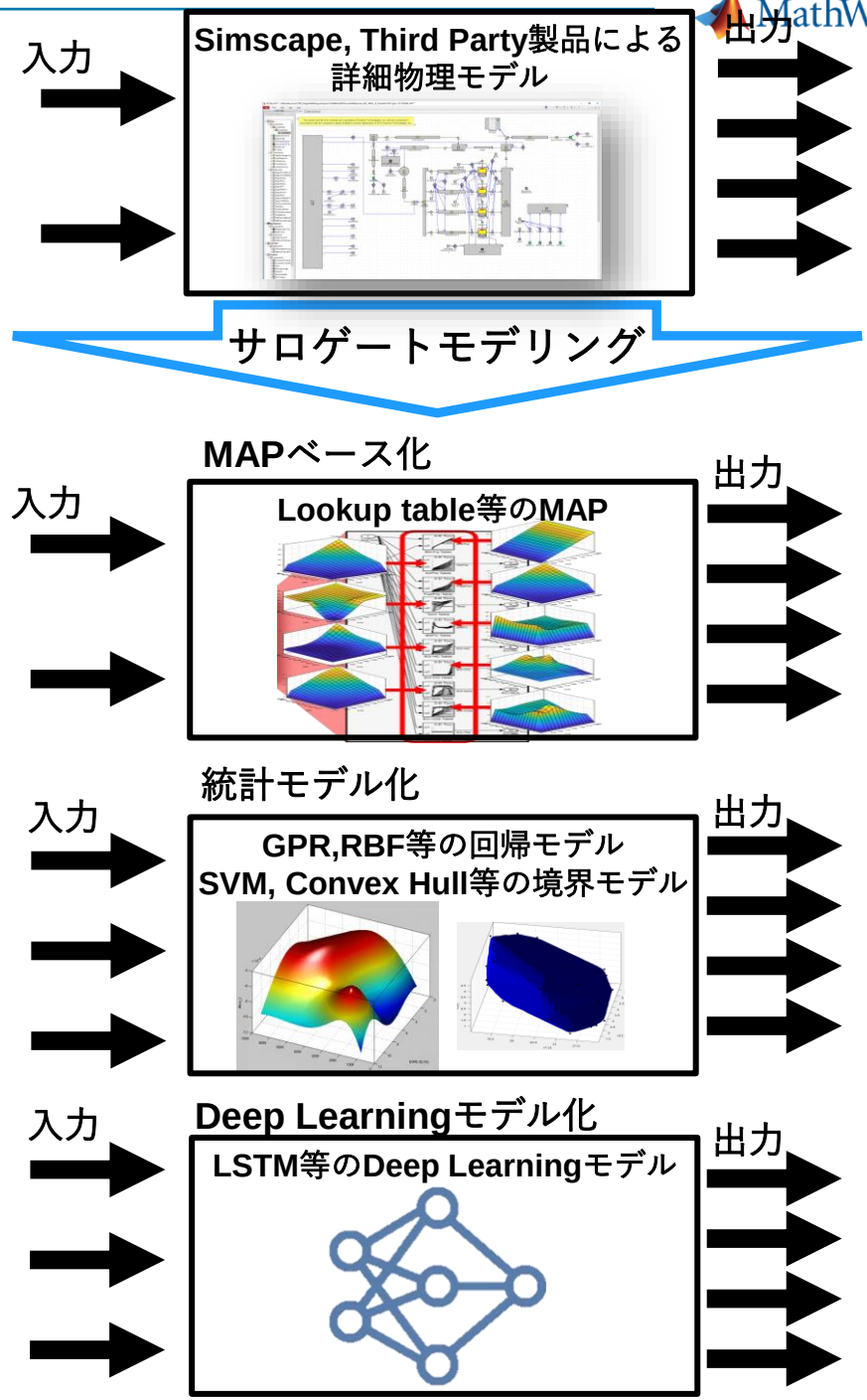
⇒定常モデル

- **MAPベース化**
- **統計モデル化**

⇒過渡モデル

- **Deep Learningモデル化(例：LSTMの利用)**

- Simulink上で既存の詳細モデルに対して<u>仮想試験</u>を行い上記モデルに必要なデータを取得。



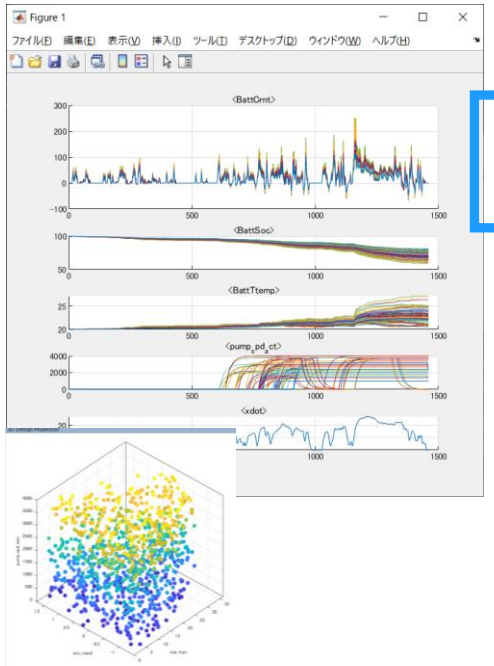
# Deep Learning（深層学習) モデル化 ワークフロー(概念図)

①仮想試験の実施
⇒学習データの取得

②DNNの検討
⇒ネットワークを学習

③学習状況の評価
⇒学習済みネットワークでの推論と評価

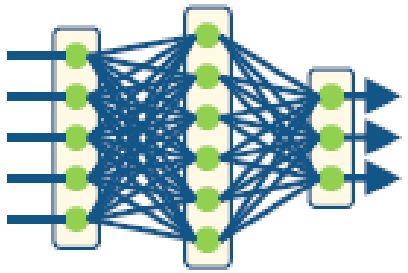
④学習済みのネットワークをSimulinkブロックとして配置
例)学習データ時間-RMSE



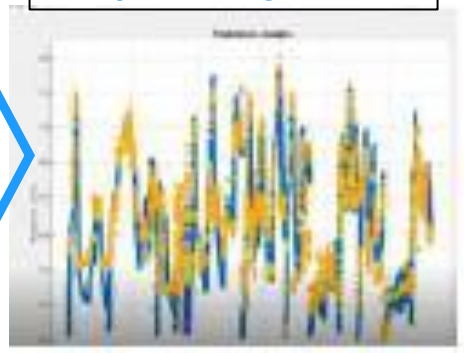

例) 実験計画法(DoE)を利用し、入力変数に関する、パラメータの設計

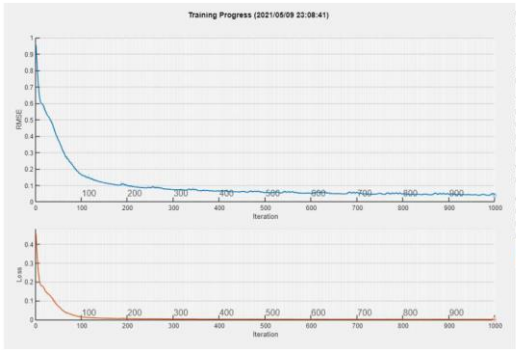
Deep Learning Toolbox


ディープネットワーク デザイナーでDNNを作成



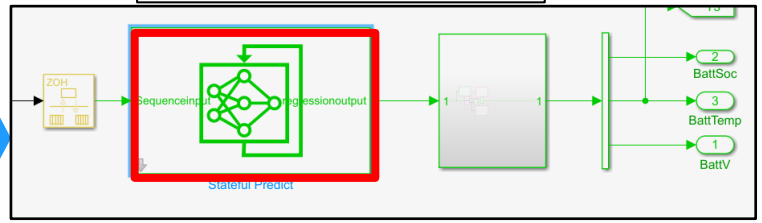
Deep Learning Toolbox





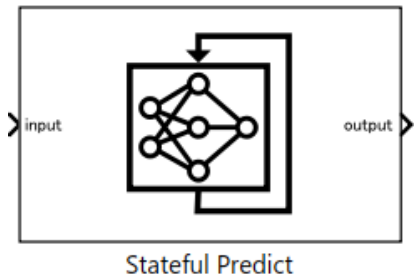

※推論：学習済みのネットワークで予測を行うこと

Deep Learning Toolbox




Stateful Predict block

R2021a

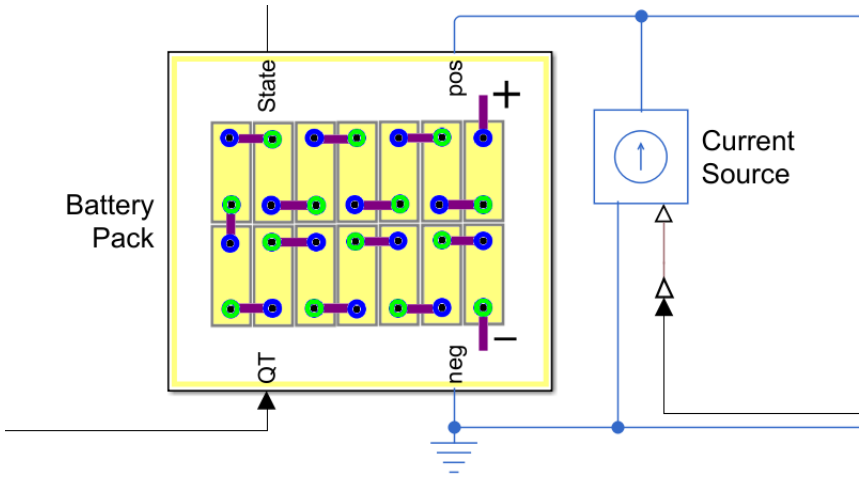



学習済みの再帰型ニューラル ネットワークを使用してInputに対する応答を予測

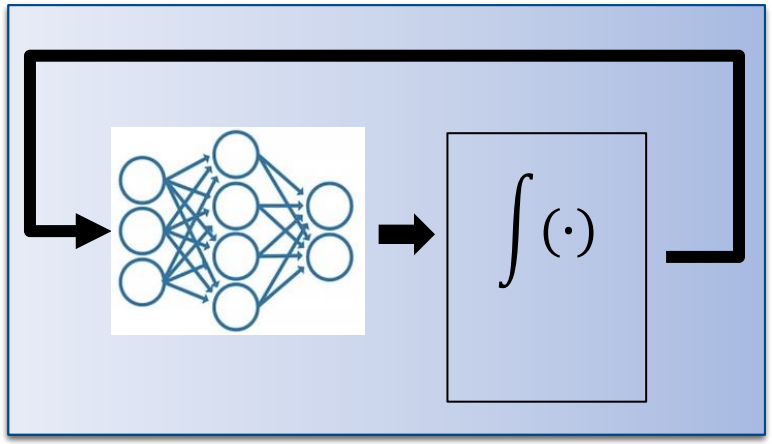
学習済みのネットワークは、MAT ファイルまたは MATLAB 関数からインポート

# サロゲートモデリングの例

## ◆バッテリーマネジメントシステム




$$Temperature = f(Current, SOC)$$



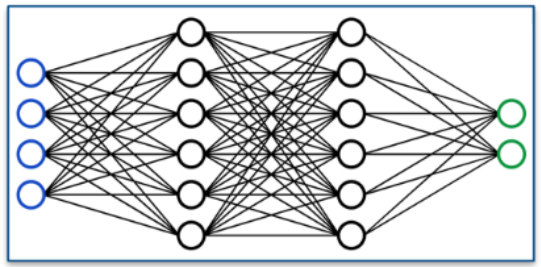

**Neural State Space**

＝ニューラルODE

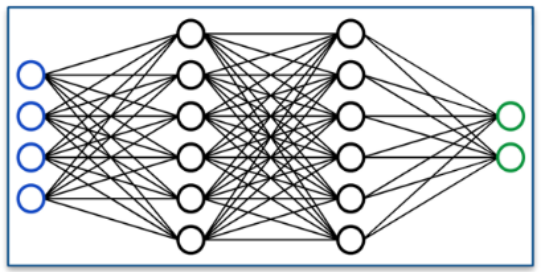
# Neural State Space とは？

$$\dot{x} = f(x,u)$$
$$y = g(x,u)$$

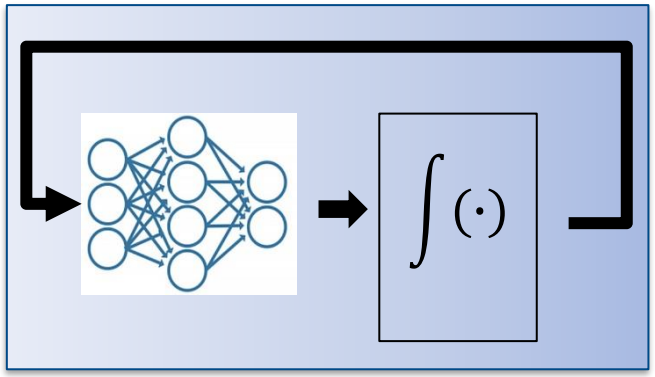
State Space (ODE)



State Network ($f$)



Output Network ($g$)




R2022b

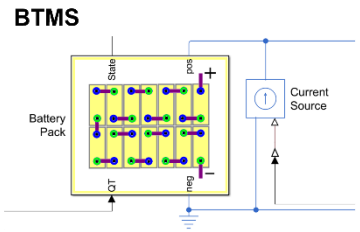
- Model: idNeuralStateSpace object
- Estimator: nlssest

Live Task R2023b

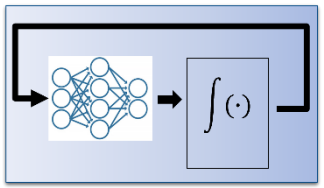
必要なライセンス：

- System Identification Toolbox
- Deep Learning Toolbox

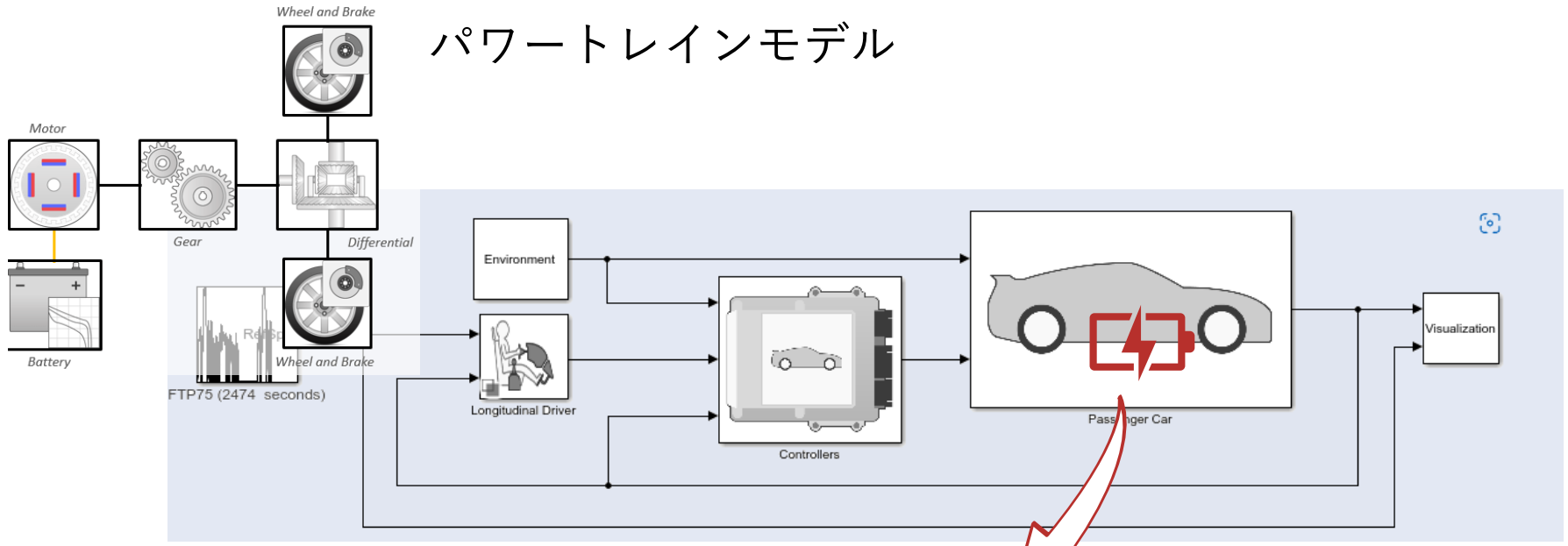
# EVバッテリーシステムに対するNeural State Spaceの適用例



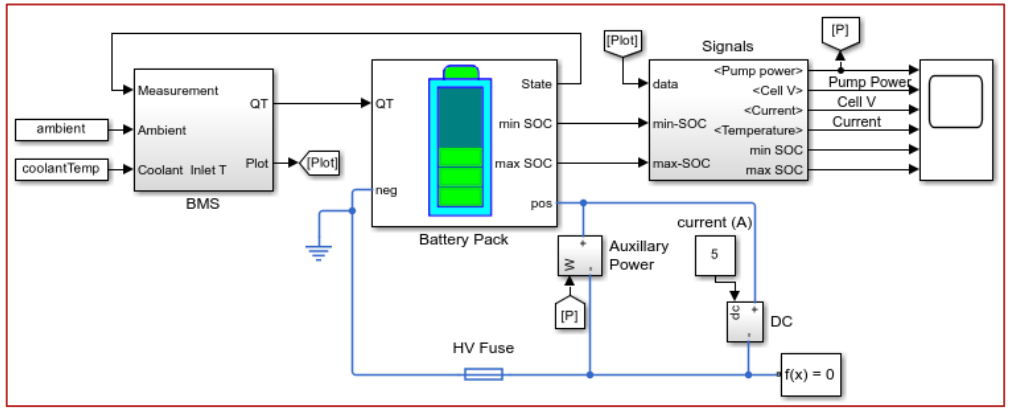

$Temperature = f(Current, SOC)$



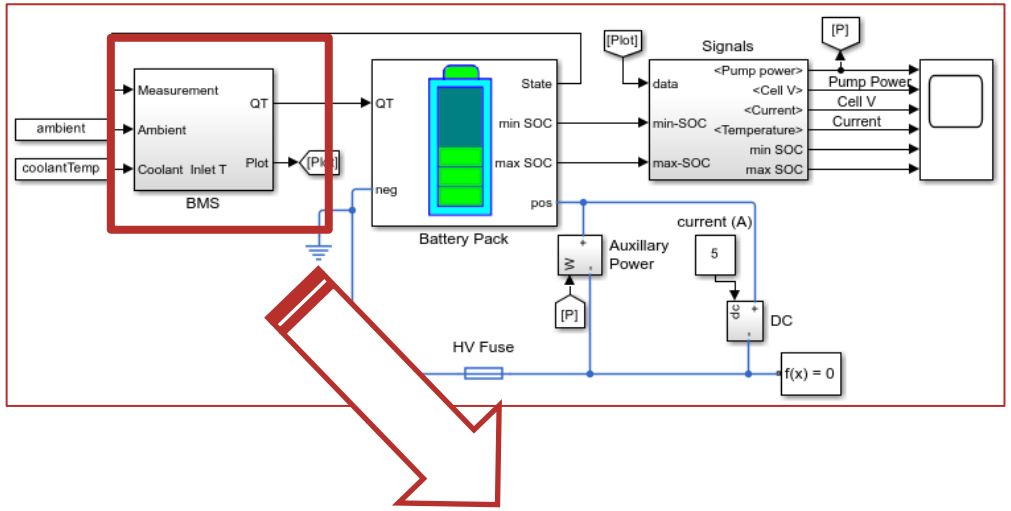

パワートレインモデル



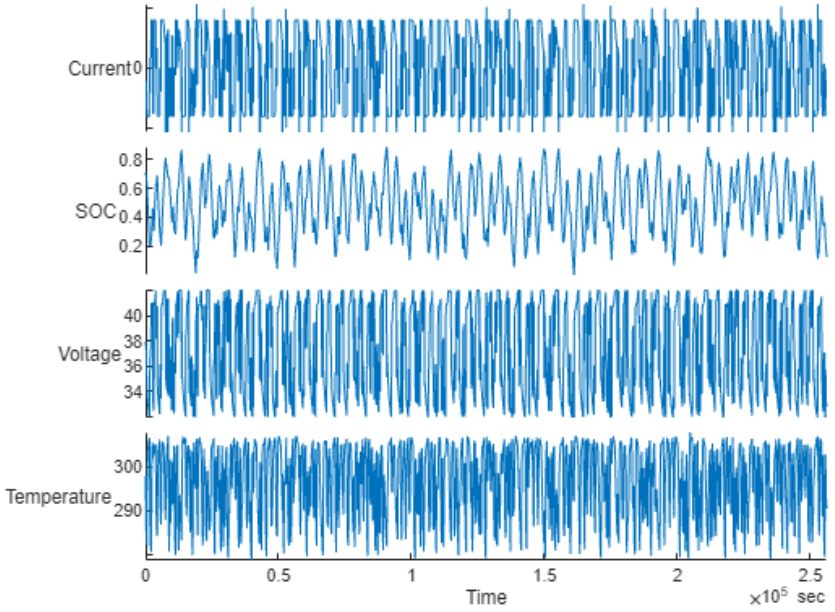

詳細なプラントモデル



# EVバッテリーシステムに対するNeural State Spaceの適用例

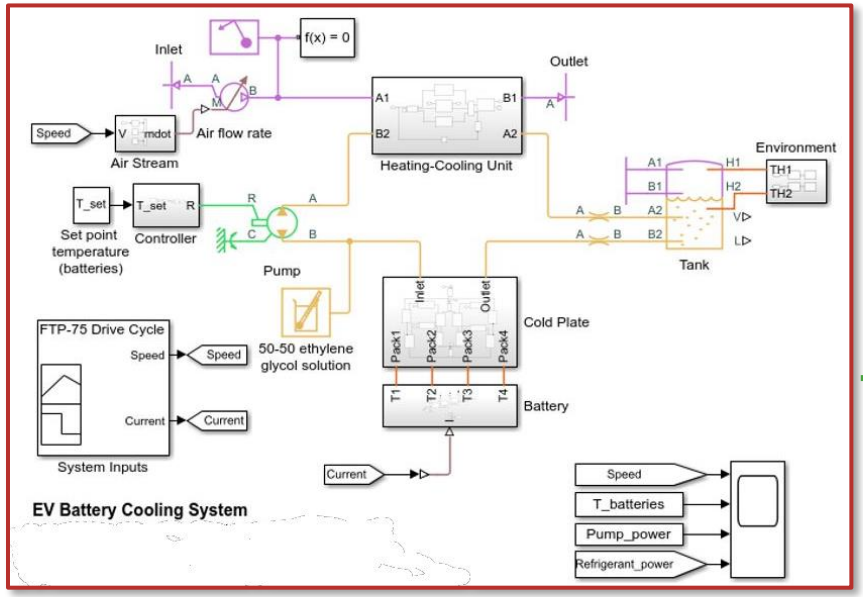



## Training Data




Current →

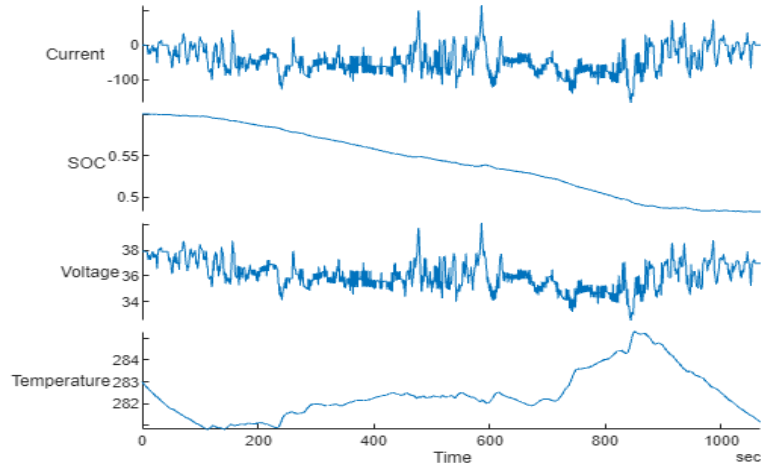
SOC →




→ Voltage

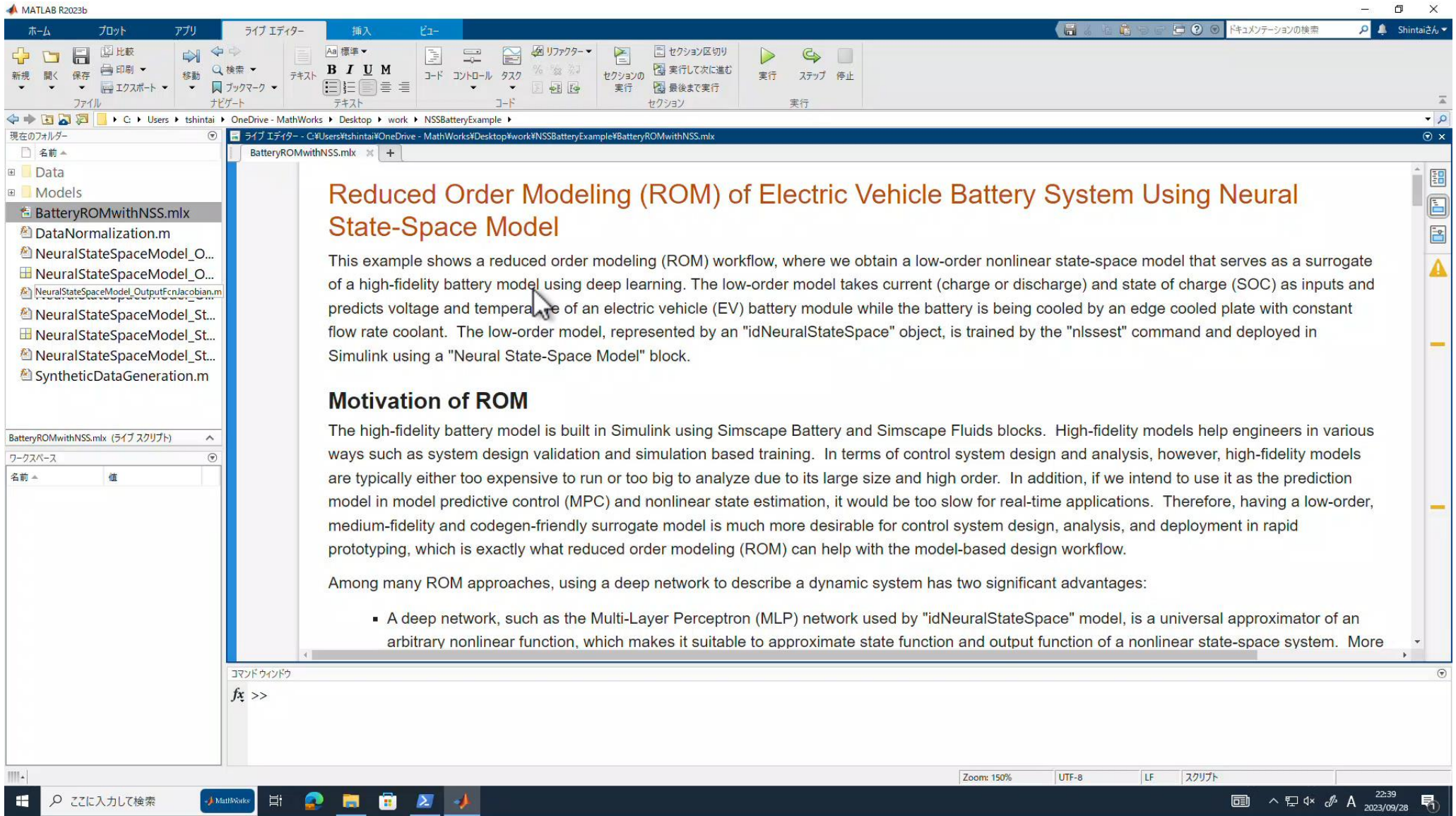
→ Temperature

## Test Data



# EVバッテリーシステムに対するNeural State Spaceの適用例

https://www.mathworks.com/videos/reduced-order-modeling-using-neural-state-space-1666078100484.html



# 現場で使える制御設計：
# 周波数応答推定と制御応答最適化

# はじめに

## 本セッションの対象者

- PIDなどのフィードバック制御器を設計している方、興味のある方
- 制御理論に詳しくないが、現場で制御設計をする必要のある方
  - 制御設計をしたいが、知見が無くて何から始めていいか分からない方

## 本セッションでお伝えしたいこと

- 周波数応答推定によりプラントを同定する方法
- 最適化アルゴリズムを用いた自動ゲイン調整・実装する方法

# アジェンダ

- 背景
- 実験的な周波数応答推定によりプラントを同定
  - Frequency Response Estimatorブロック
  - モデル線形化器
- 最適化アルゴリズムによる制御ゲイン最適化
  - 応答オプティマイザー

# アジェンダ

- 背景

- 実験的な周波数応答推定によりプラントを同定
  - Frequency Response Estimatorブロック
  - モデル線形化器

- 最適化アルゴリズムによる制御ゲイン最適化
  - 応答オプティマイザー

# 今後、産業界において「制御系設計」の役割は一段と重要になる

## 制御系設計の課題

- 制御性能向上・開発工数削減
  - 複雑化する制御系に、高性能・高品質化かつ迅速な開発が求められる
- 次世代に向けた価値創発
  - 製品/サービスの高度化を実現する先進的な制御技術が求められる
- システム制御工学が提供する解決手段の浸透
  - 設計や実装における問題解決（例：目標性能維持、実装コスト削減）の手段として、システム制御工学視点の考え方やアプローチを定着させる
- 社会・組織における認知度向上
  - 制御系設計はあらゆる産業において必要不可欠な存在であり、社会・組織におけるさらなるアピールが必要となる

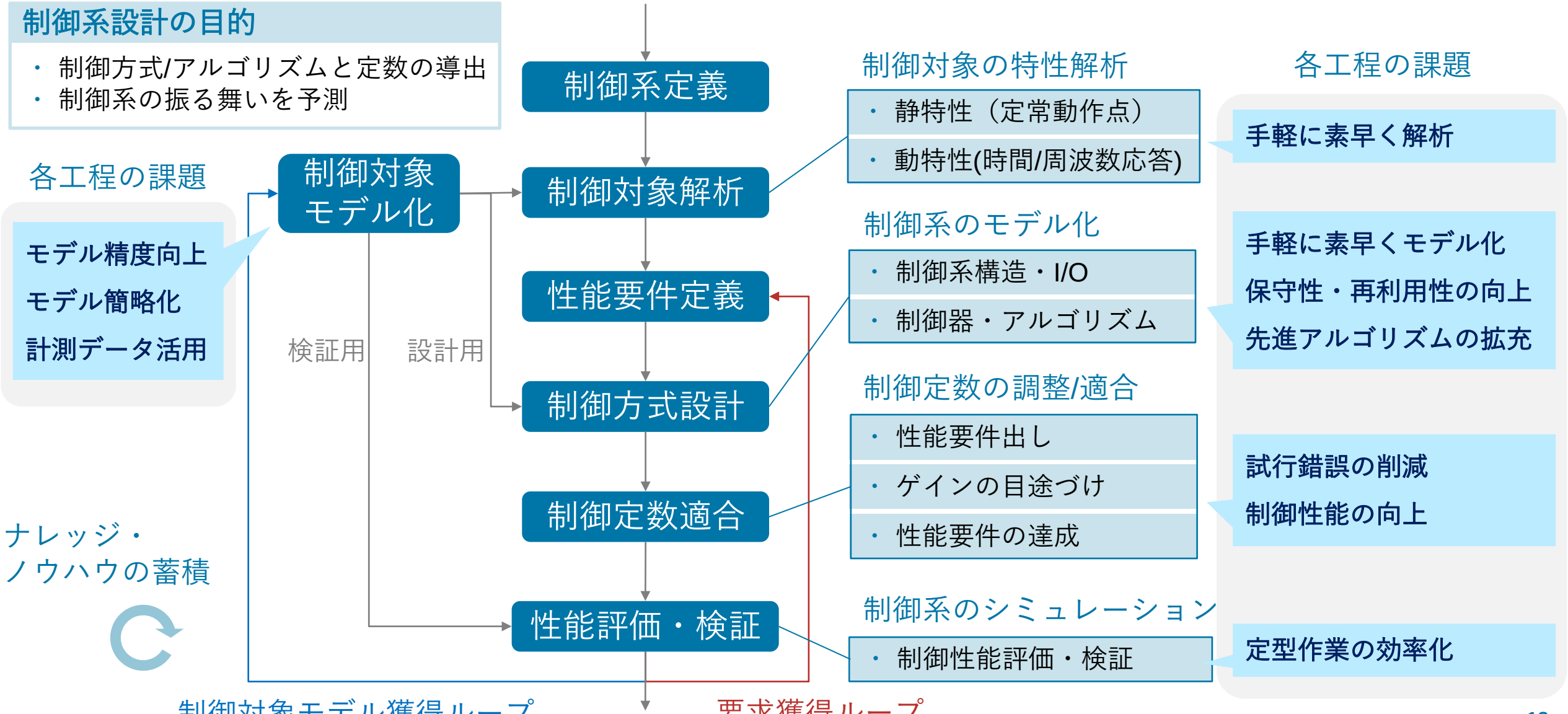
## 制御の難易度は上昇

- 厳しい要求
  - 規制、性能、信頼性
  - 制約条件
- システムの複雑さ
  - コンポーネント増加
  - 場合分け増加
  - 複雑な相互作用
  - 複雑な非線形性
- バラつき・不確実性
  - 環境変化、部品、劣化
  - 人・環境・経済の介在

# 制御系設計のサイクルを 「早く、賢く繰り返す」 ことが課題

**制御系設計の目的**

- 制御方式/アルゴリズムと定数の導出
- 制御系の振る舞いを予測



# 各工程を支援する「ツール」を活用できます

**制御系設計の目的**

- 制御方式/アルゴリズムと定数の導出
- 制御系の振る舞いを予測

制御系定義

制御対象解析

性能要件定義

制御方式設計

制御定数適合

性能評価・検証

制御対象モデル化

設計用

アプリ・機能

- パラメータ推定器
- システム同定
- モデル・リデューサー

ナレッジ・ノウハウの蓄積

制御対象モデル獲得ループ

要求獲得ループ

制御対象の特性解析

- 静特性（定常動作点）
- 動特性(時間/周波数応答)

制御系のモデル化

- 制御系構造・I/O
- 制御器・アルゴリズム

制御定数の調整/適合

- 性能要件出し
- ゲインの目途づけ
- 性能要件の達成

制御系のシミュレーション

- 制御性能評価・検証

アプリ・機能

- 定常状態マネージャー
- モデル線形化器
- ブロックライブラリ
- PID 調整器
- 制御システム調整器
- 応答オプティマイザー
- 感度アナライザー

# 各工程における制御系設計ソリューションの例

**制御系設計の目的**

- 制御方式/アルゴリズムと定数の導出
- 制御系の振る舞いを予測

制御系定義

制御対象解析

性能要件定義

制御方式設計

制御定数適合

性能評価・検証

制御対象モデル化

1 データ活用による制御対象のモデル化と精度向上

2 モデルの簡略化・低次元化を直感的に手軽に実践

制御対象の特性解析

- 静特性（定常動作）
- 動特性(時間/周波数応答)

3 制御対象や制御系の特性をSimulink モデルで素早く解析

制御系のモデル

- 制御系構造・I/O
- 制御器・アルゴリズム

4 ツールのライブラリブロックで制御系を手軽に素早くモデル化

制御定数の調整/適合

- 性能要件出し
- ゲインの目途づけ
- 性能要件の達成

5 制御器の自動調整ツールで工数削減・制御性能向上を達成

制御系のシミュレーション

- 制御性能評価・検証

ナレッジ・
ノウハウの蓄積

制御対象モデル獲得ループ

要求獲得ループ

# 日産自動車が空燃比コントローラーに要するキャリブレーション時間を短縮し、排出ガス性能を改善

## 課題

エンジン排出ガスの削減、および、キャリブレーション時間短縮によるAFR コントローラーの製品開発の加速

## 解決策

モデルベースデザインに Optimization Toolbox と Simulink Design Optimization を使用することで、コントローラー性能を最適化し、パラメーターのキャリブレーションを自動化

## 結果

- $NO_x$ および CO の排出を半分以上削減
- キャリブレーション時間を 90% 短縮
- キャリブレーション品質の安定化

日産 アルティマ

「日産は、長年にわたり、モデルベースデザインを用いて開発を加速し製品を改善する新しい方法を模索してきました。その中で、MATLAB と Simulink の最適化製品を導入し、既存の AFR コントローラーの性能改善により排ガスを削減させることができました。」

-日産自動車株式会社、加藤浩志氏

Link to user story

# アジェンダ

- 背景
- 実験的な周波数応答推定によりプラントを同定
  - Frequency Response Estimatorブロック
  - モデル線形化器
- 最適化アルゴリズムによる制御ゲイン最適化
  - 応答オプティマイザー

# 周波数応答推定の基本

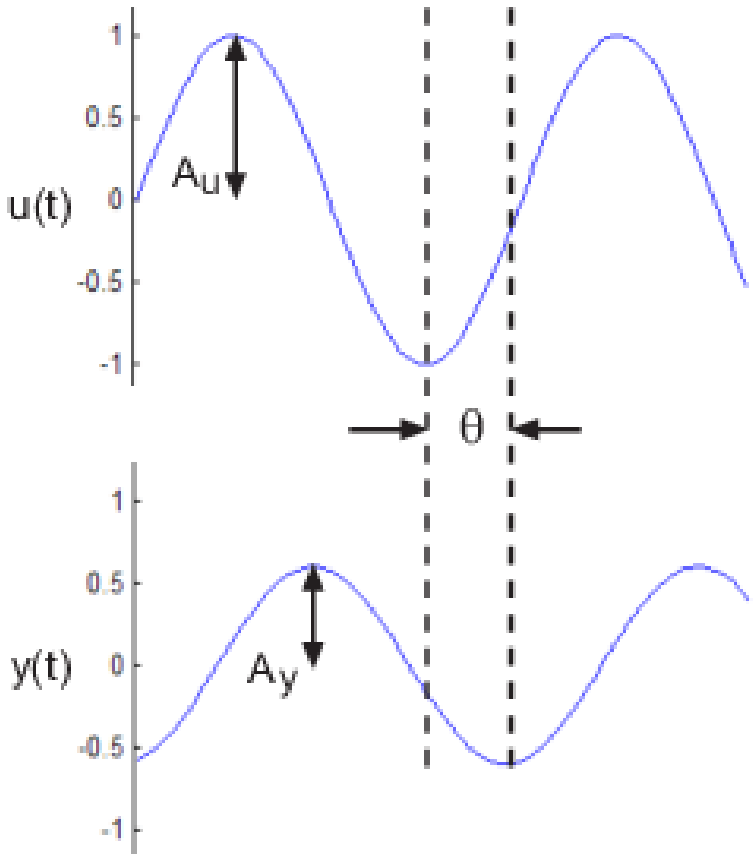
「周波数応答」は、正弦波入力に対するシステムの定常状態応答のこと。

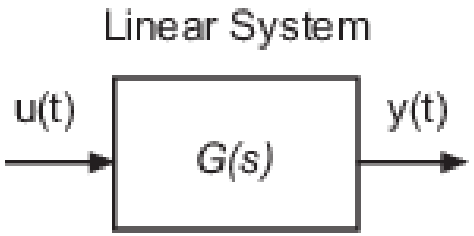
線形システムに正弦波u(t)を入力する。

$$u(t) = A_u \sin\omega t$$

出力y(t)は、以下のようになる。

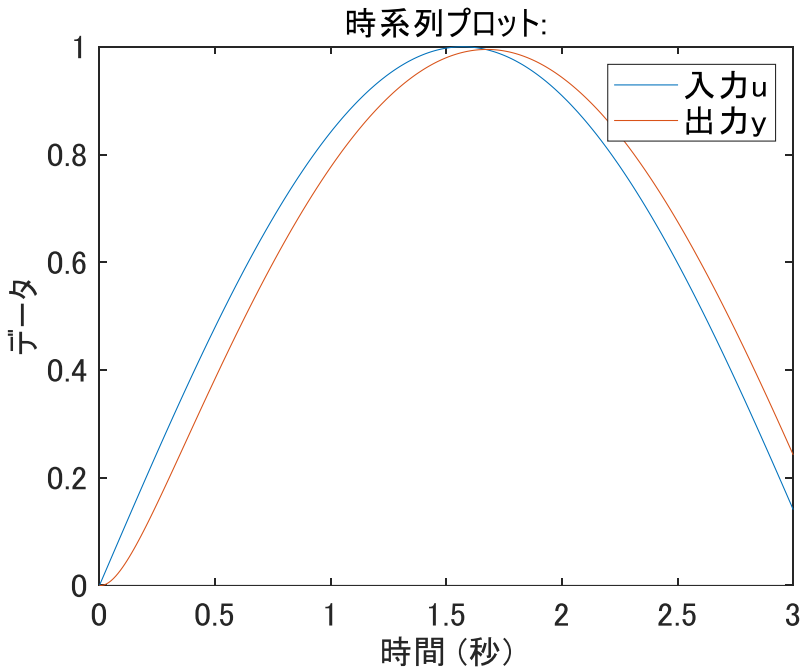
$$y(t) = A_y \sin(\omega t + \theta)$$

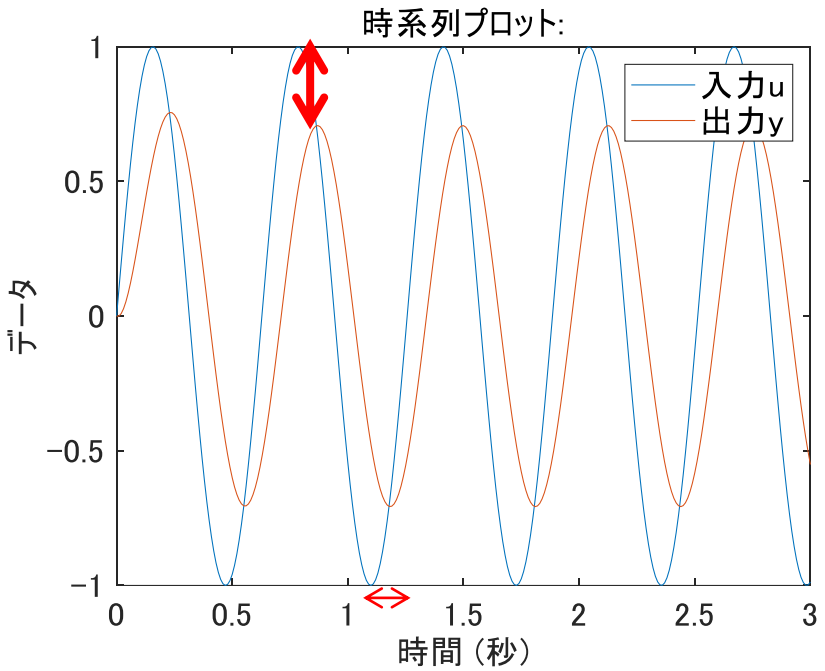


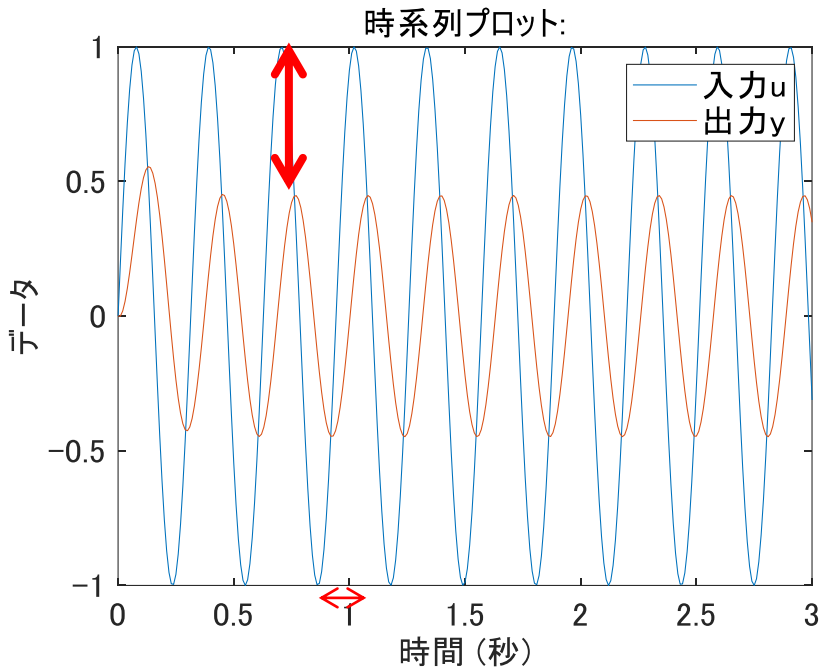



# 周波数応答推定の基本

周波数を変えると、振幅比（$A_y / A_u$）と位相差（θ）は変化する。



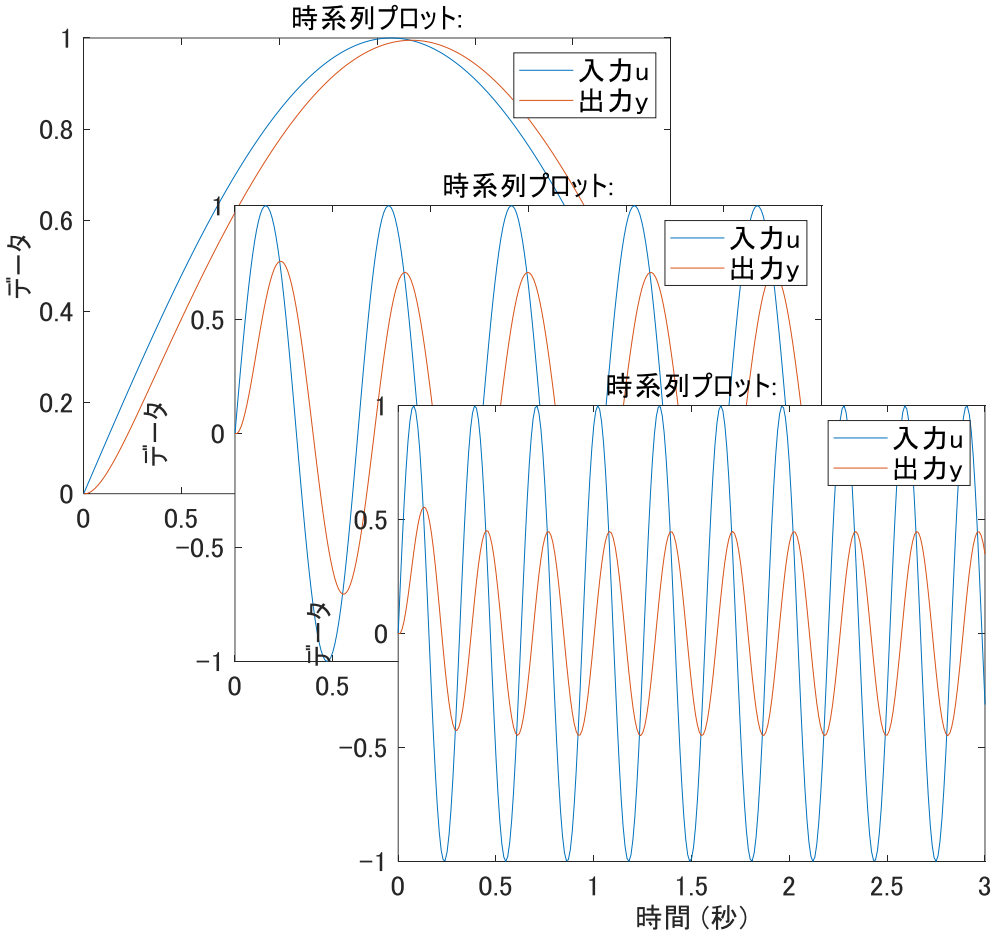



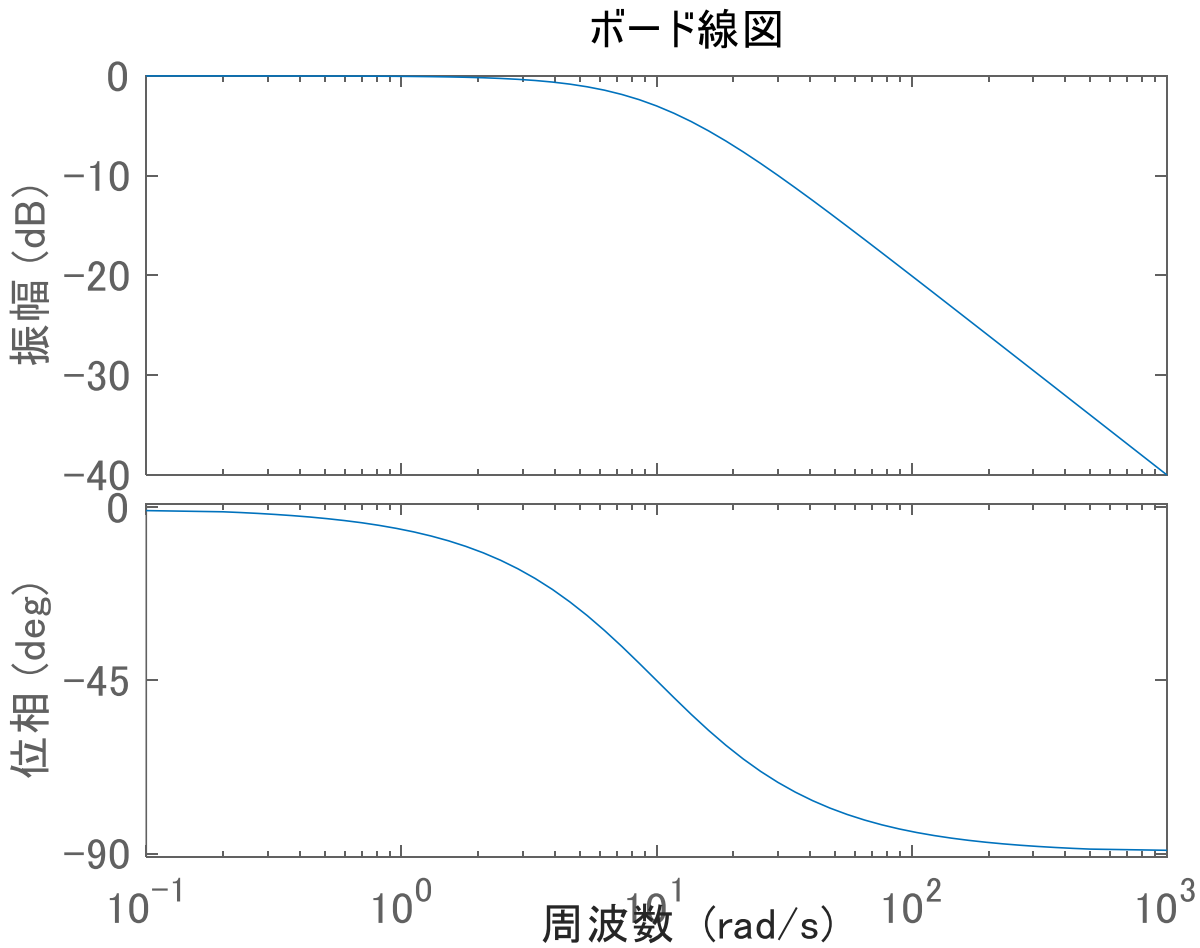

低い ⟷ 高い

周波数 Hz

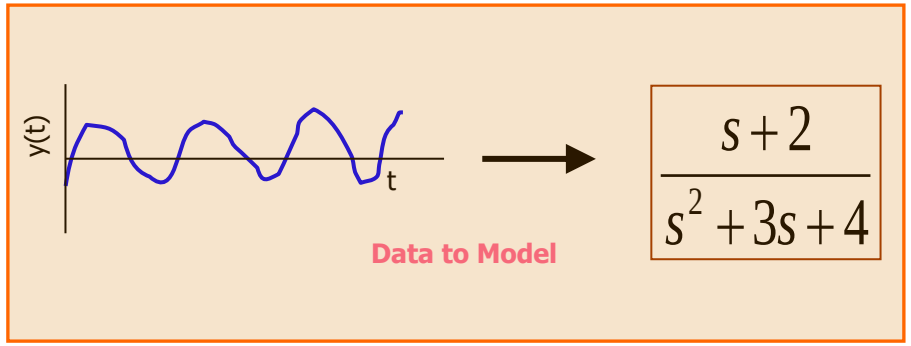
# 周波数応答推定の基本

各周波数ごとの振幅比、位相差の特性を取れば、プラントの特性が分かる。



# 周波数応答が分かると何がいいのか

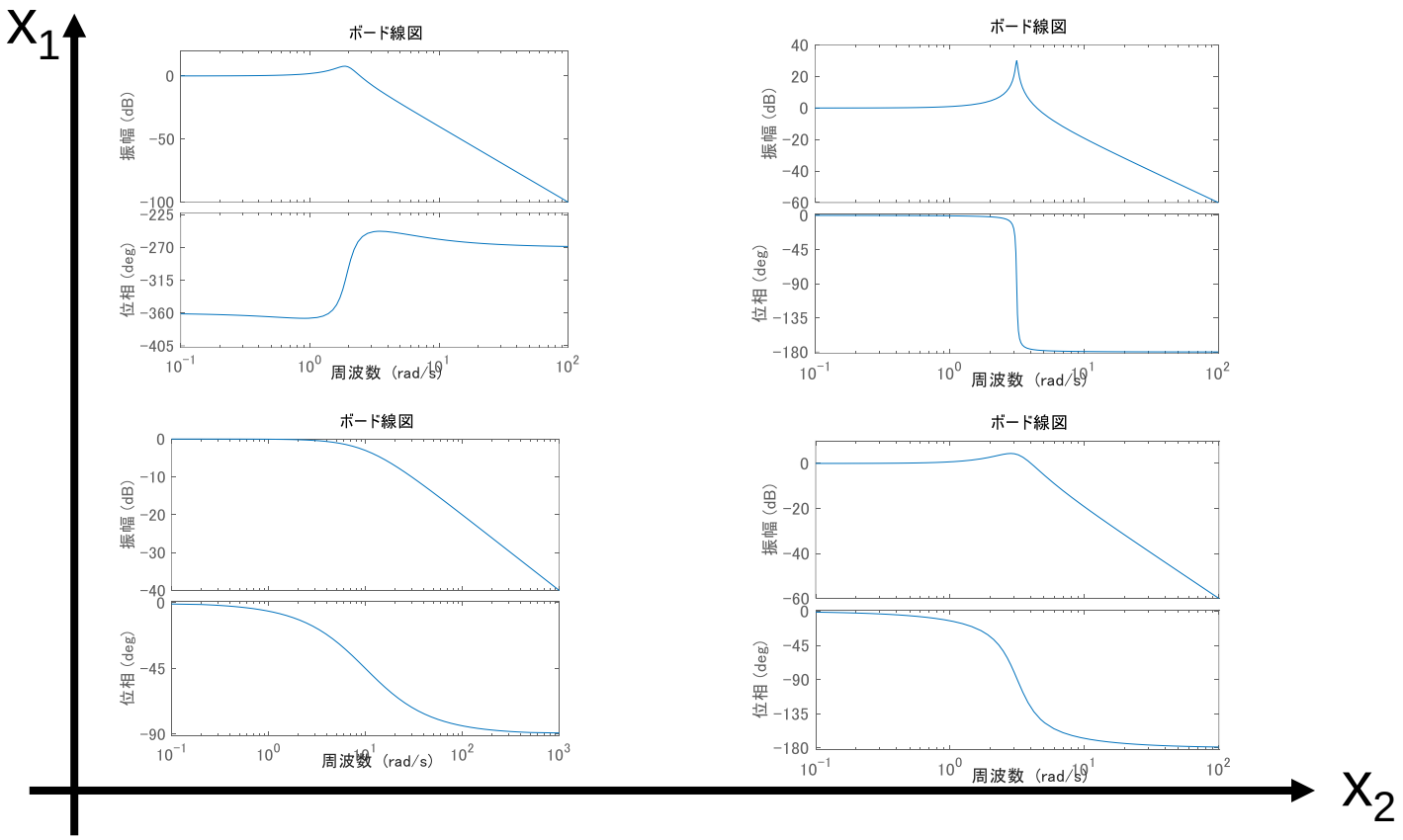



- プラントの特性が理解できる
- プラントの伝達関数を推定できる
  - System Identification Toolboxを用いてシステム同定可能
- 伝達関数モデルを使ってシミュレーションできる
- 制御理論を用いて最適な制御ゲインを導出できる

プラントに対する科学的知識を持っていなくても、周波数特性さえ分かれば制御設計できるようになります！

# 非線形システムでは？

いくつかの非線形システムでは、特定の状態近傍では線形であると見なすことができる。そうであれば、部分ごとに周波数応答推定を行い、領域全体の特性を掴むこともできる。

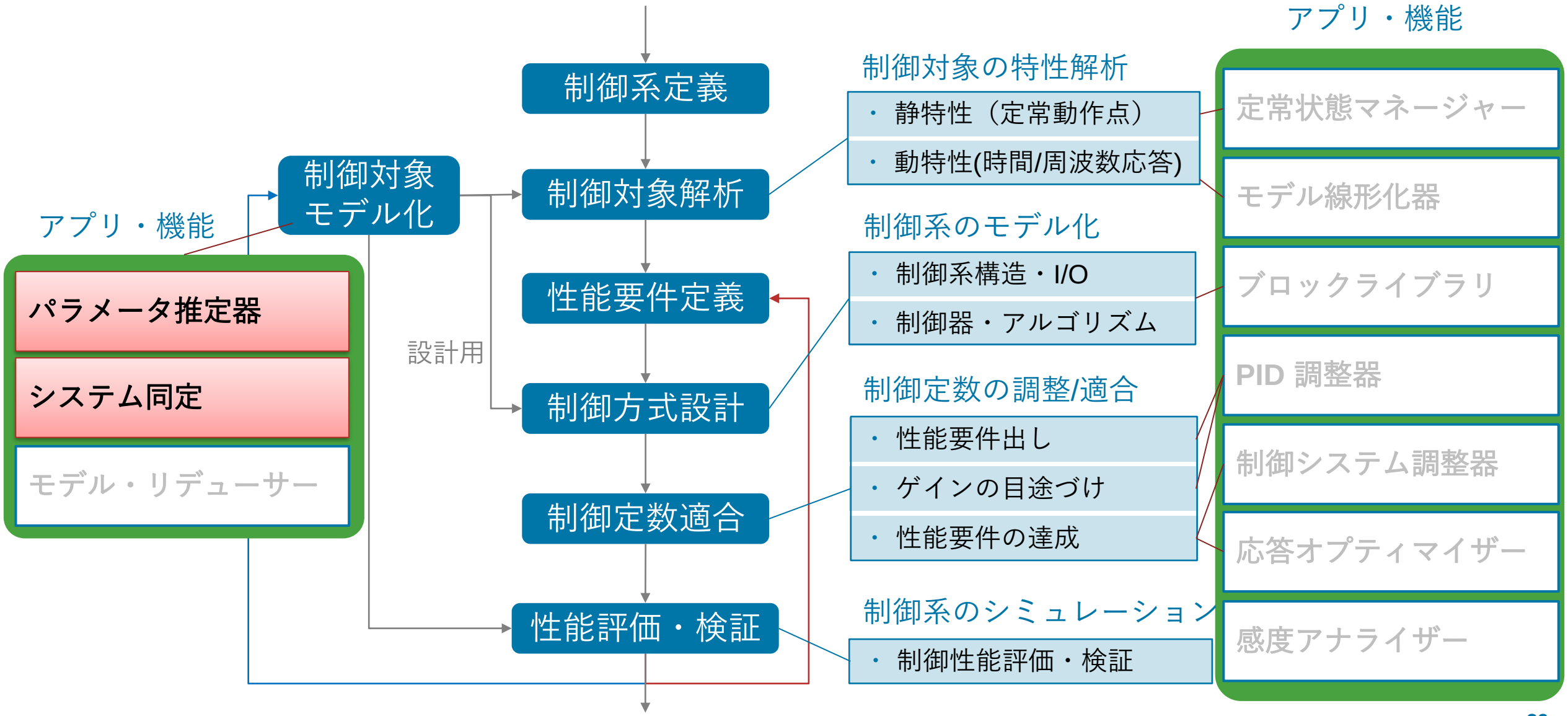



強い非線形性を持つシステム（例えば不連続に変化するシステムなど）は、周波数特性を得ることはできませんが、滑らかに特性が変化するシステムであれば、局所的に推定することができます。

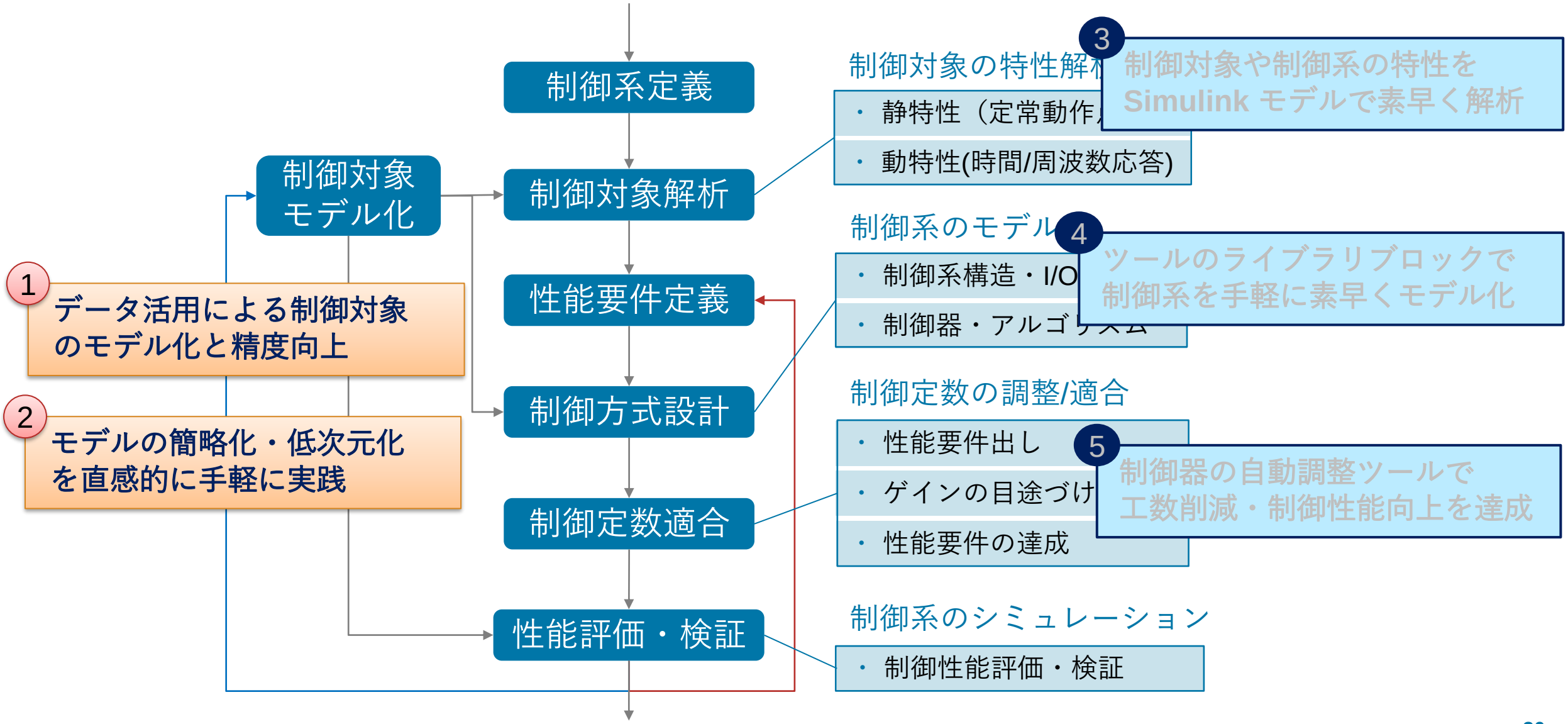
# アジェンダ

- 背景
- 実験的な周波数応答推定によりプラントを同定
  - Frequency Response Estimatorブロック
  - モデル線形化器
- 最適化アルゴリズムによる制御ゲイン最適化
  - 応答オプティマイザー

# 使うツールの立ち位置


制御系定義
制御対象解析
性能要件定義
制御方式設計
制御定数適合
性能評価・検証
制御対象モデル化
設計用
アプリ・機能
パラメータ推定器
システム同定
モデル・リデューサー
制御対象の特性解析
・静特性（定常動作点）
・動特性(時間/周波数応答)
制御系のモデル化
・制御系構造・I/O
・制御器・アルゴリズム
制御定数の調整/適合
・性能要件出し
・ゲインの目途づけ
・性能要件の達成
制御系のシミュレーション
・制御性能評価・検証
アプリ・機能
定常状態マネージャー
モデル線形化器
ブロックライブラリ
PID 調整器
制御システム調整器
応答オプティマイザー
感度アナライザー

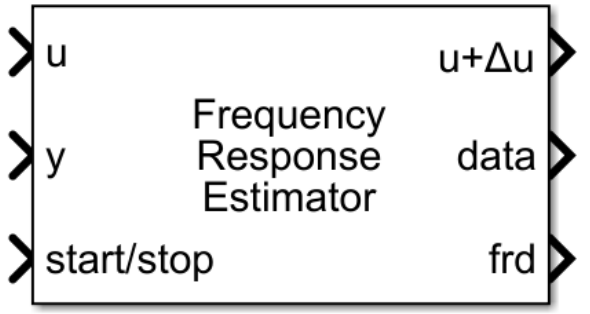
# 扱うソリューション


制御系定義
制御対象解析
性能要件定義
制御方式設計
制御定数適合
性能評価・検証
制御対象
モデル化
1
データ活用による制御対象
のモデル化と精度向上
2
モデルの簡略化・低次元化
を直感的に手軽に実践
制御対象の特性解
・静特性（定常動作
・動特性(時間/周波数応答)
3
制御対象や制御系の特性を
Simulink モデルで素早く解析
制御系のモデル
・制御系構造・I/O
・制御器・アルゴ
4
ツールのライブラリブロックで
制御系を手軽に素早くモデル化
制御定数の調整/適合
・性能要件出し
・ゲインの目途づけ
・性能要件の達成
5
制御器の自動調整ツールで
工数削減・制御性能向上を達成
制御系のシミュレーション
・制御性能評価・検証

# 周波数応答推定器, Frequency Response Estimator

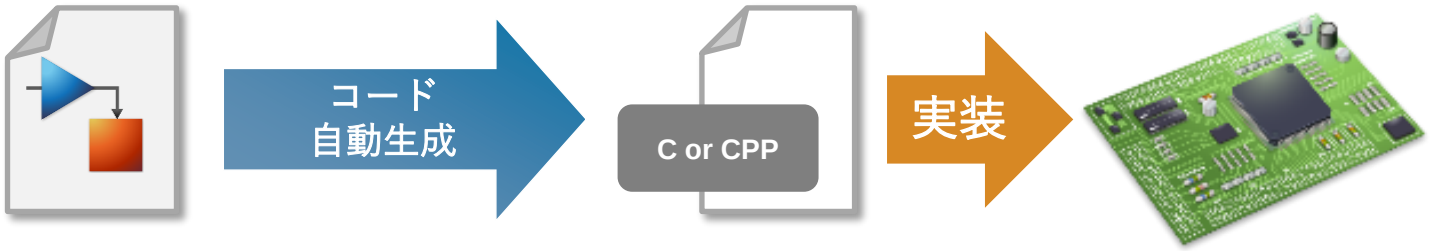
R2019a

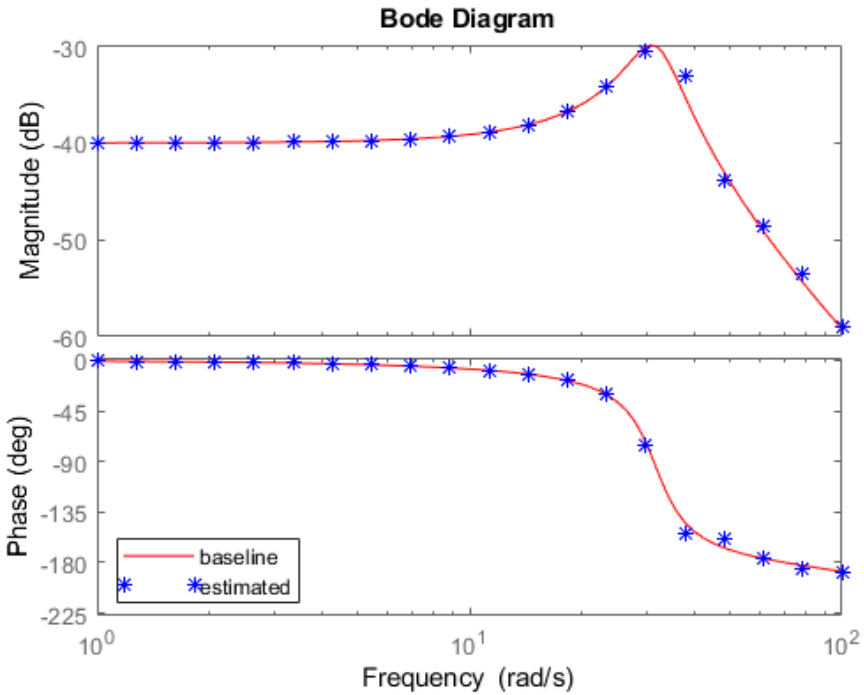
入力信号に対する応答を計測し、数値解析を行って周波数特性を計算する。




特徴

- SISOのシステムに対して周波数応答の推定が可能
- 推定機能をコード生成して実機に実装可能
- 推定対象が非線形であっても推定実験可能



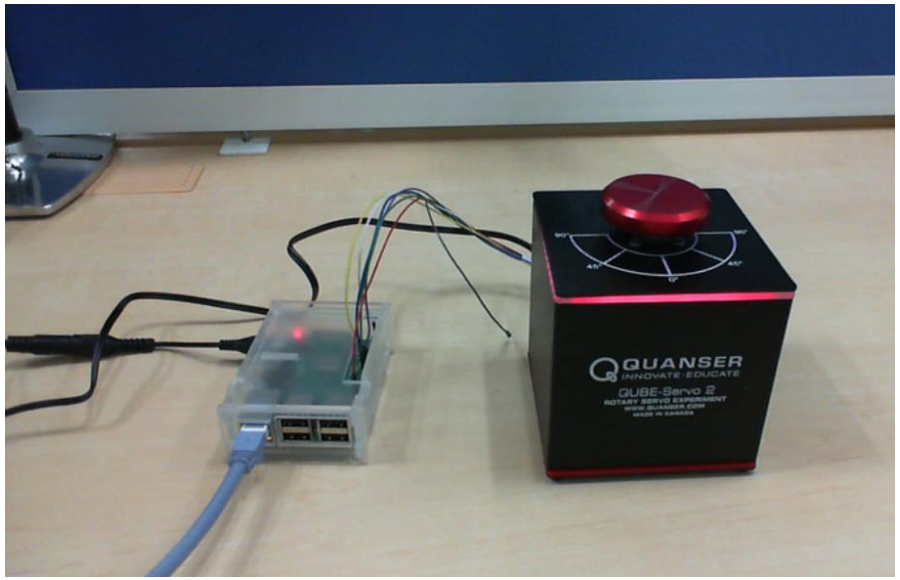



# 【デモ】Quanser “QUBE – Servo 2” を用いた推定

- Quanser社の”QUBE – Servo 2”という倒立振子装置を用いる
- 振子部分を外し、シンプルな円盤を取り付けることでDCモーターのみのシステムにすることができる

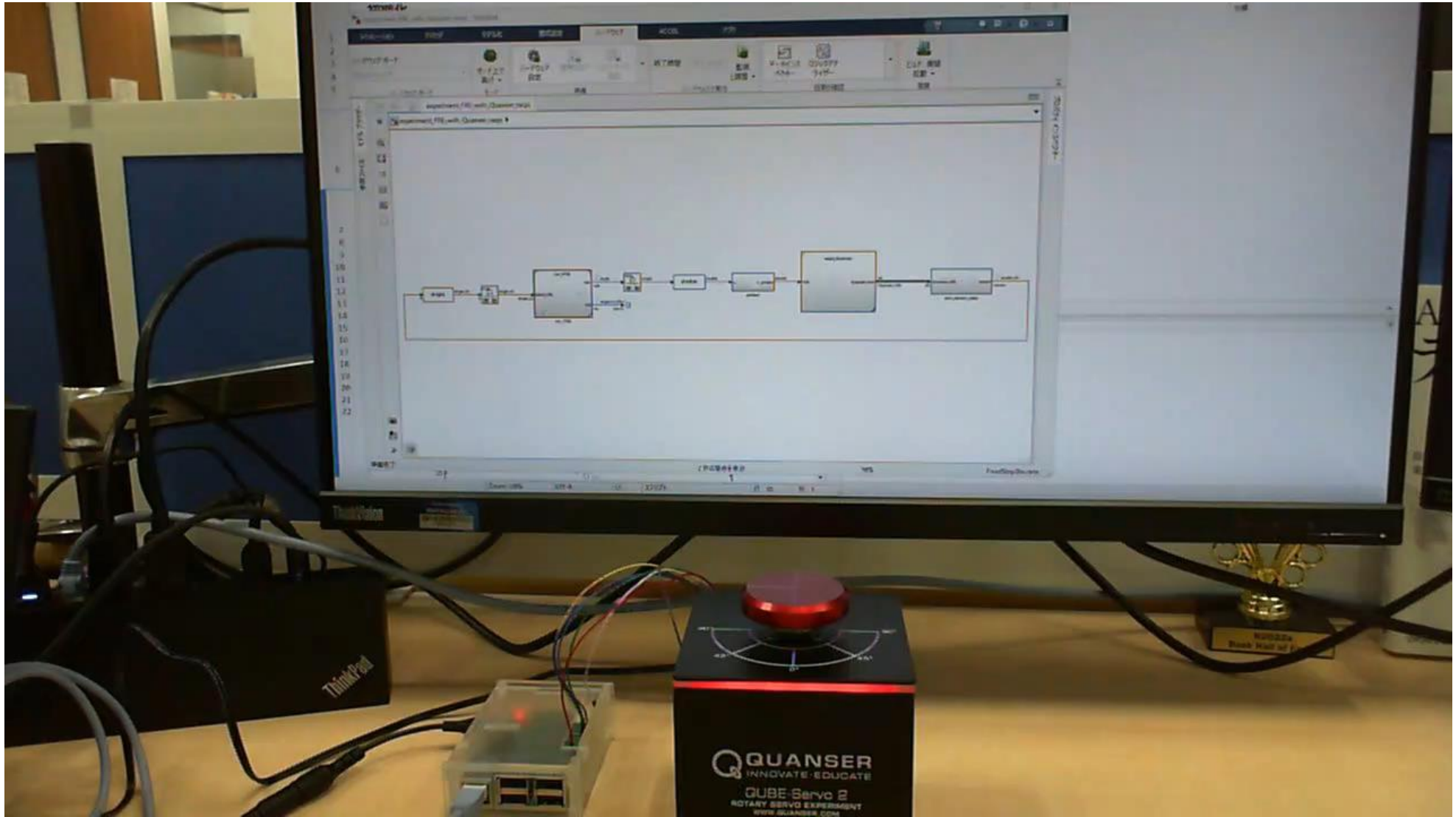
本来は倒立振子の実験装置であるが、


今回は振子を外して円盤を取り付ける

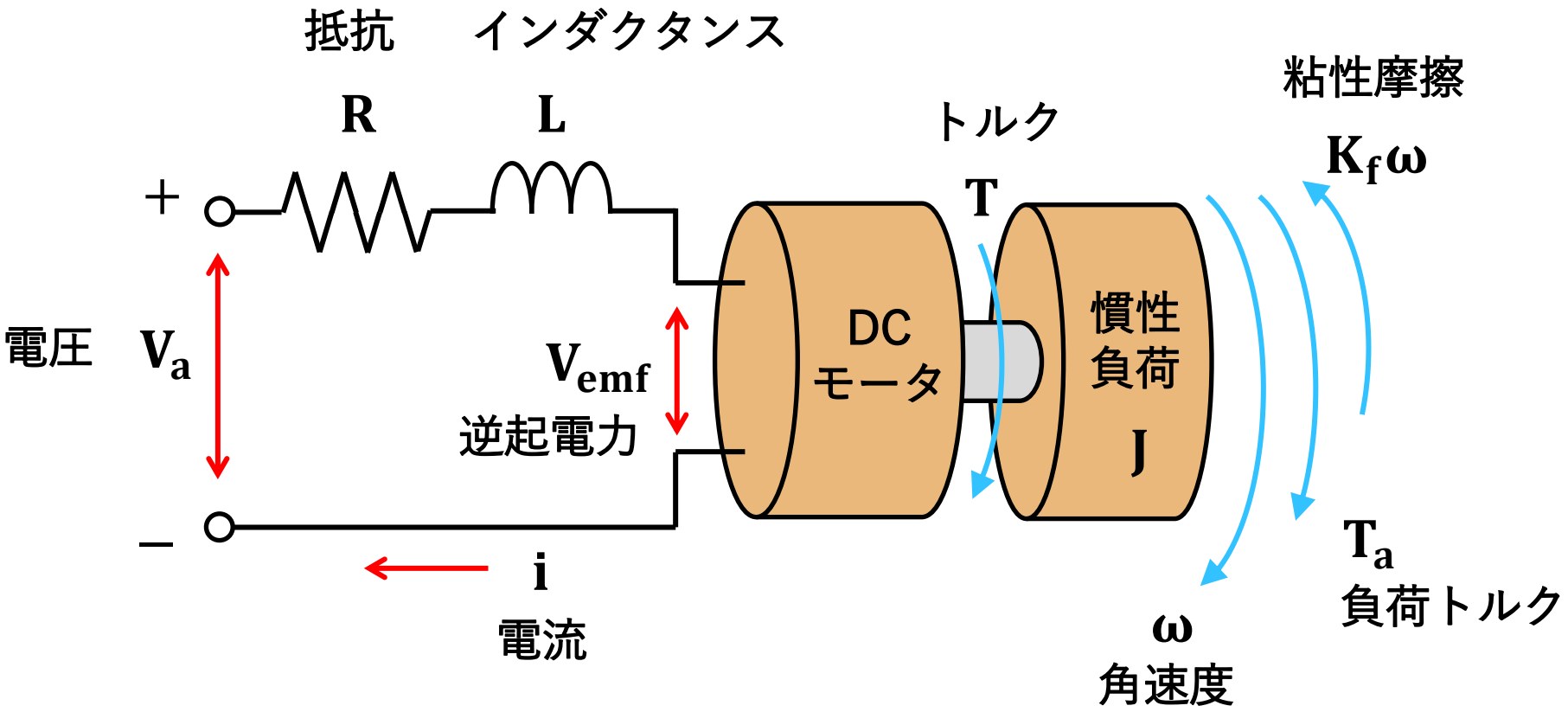


# 実機による周波数応答推定

https://www.mathworks.com/videos/model-based-design-improves-power-electronics-and-motor-drive-systems-development-efficiency-1699973729138.html

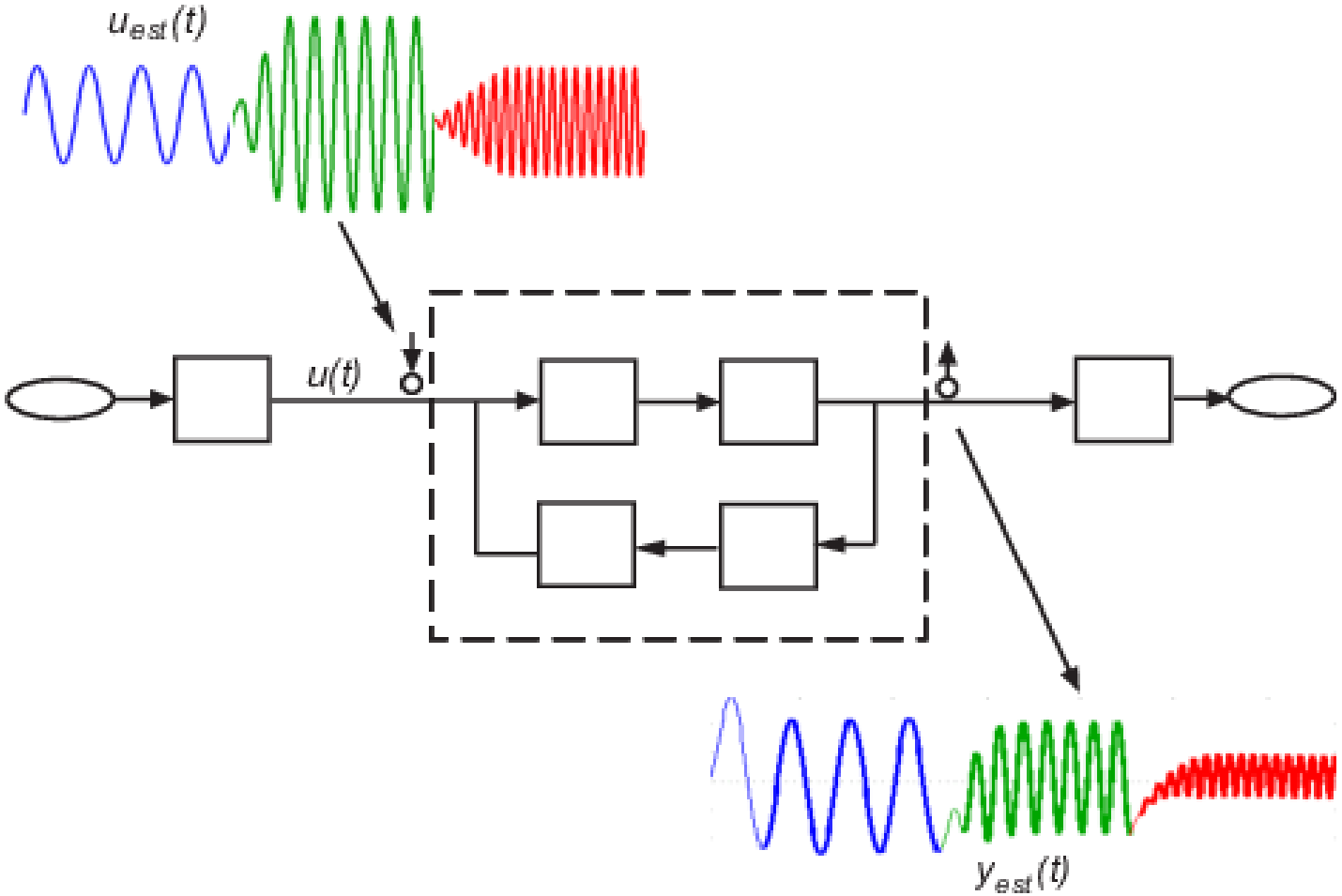


# 【参考】DCモータの等価回路



# 推定用の入力信号

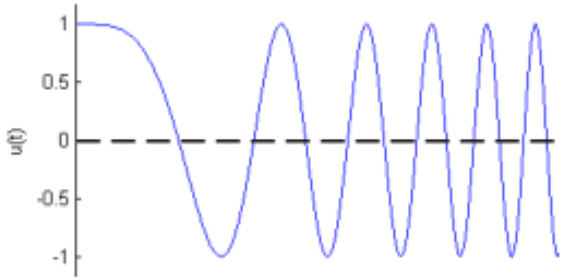
- Sinestream
  - ある一定の周波数信号を順番に流していく



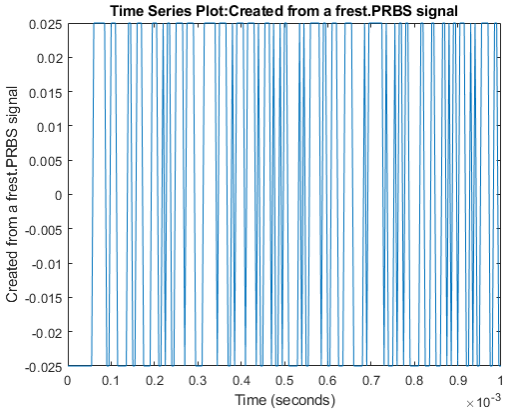

Sinestreamは、特定の周波数点ごとに確実に特性を得ることができるが、推定のための実験時間が長くなる。

# Sinestream信号よりも短い時間で推定できる信号

- チャープ
  - 周波数が徐々に変化していく信号

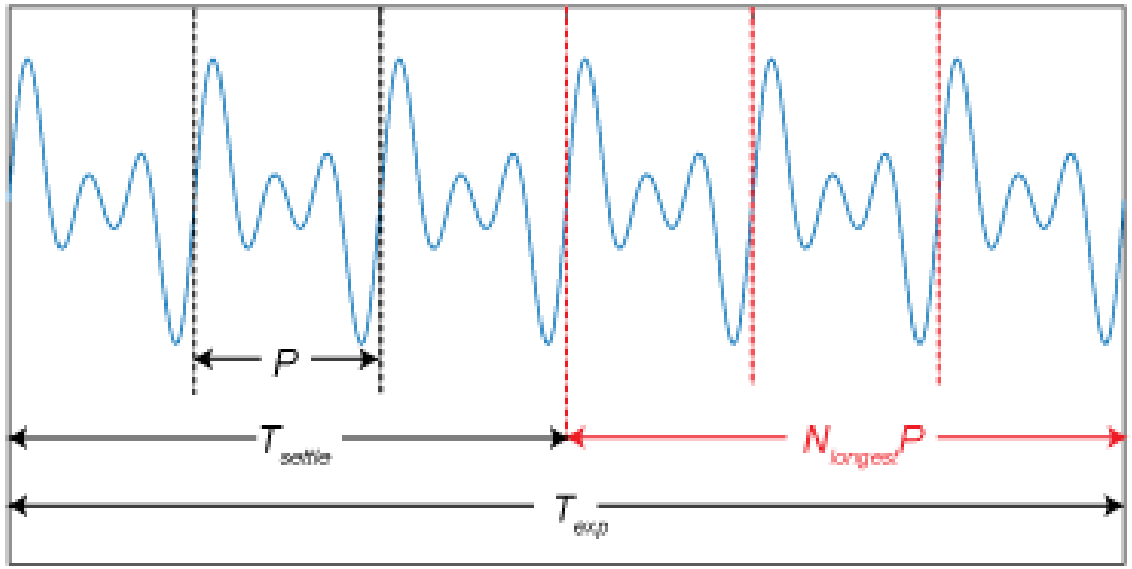



- PRBS
  - 2 つの値の間で切り替わるホワイトノイズのような信号



# Sinestream信号よりも短い時間で推定できる信号

- 重ね合わせ
  - 特定の周波数、振幅を持つ信号の重ね合わせ

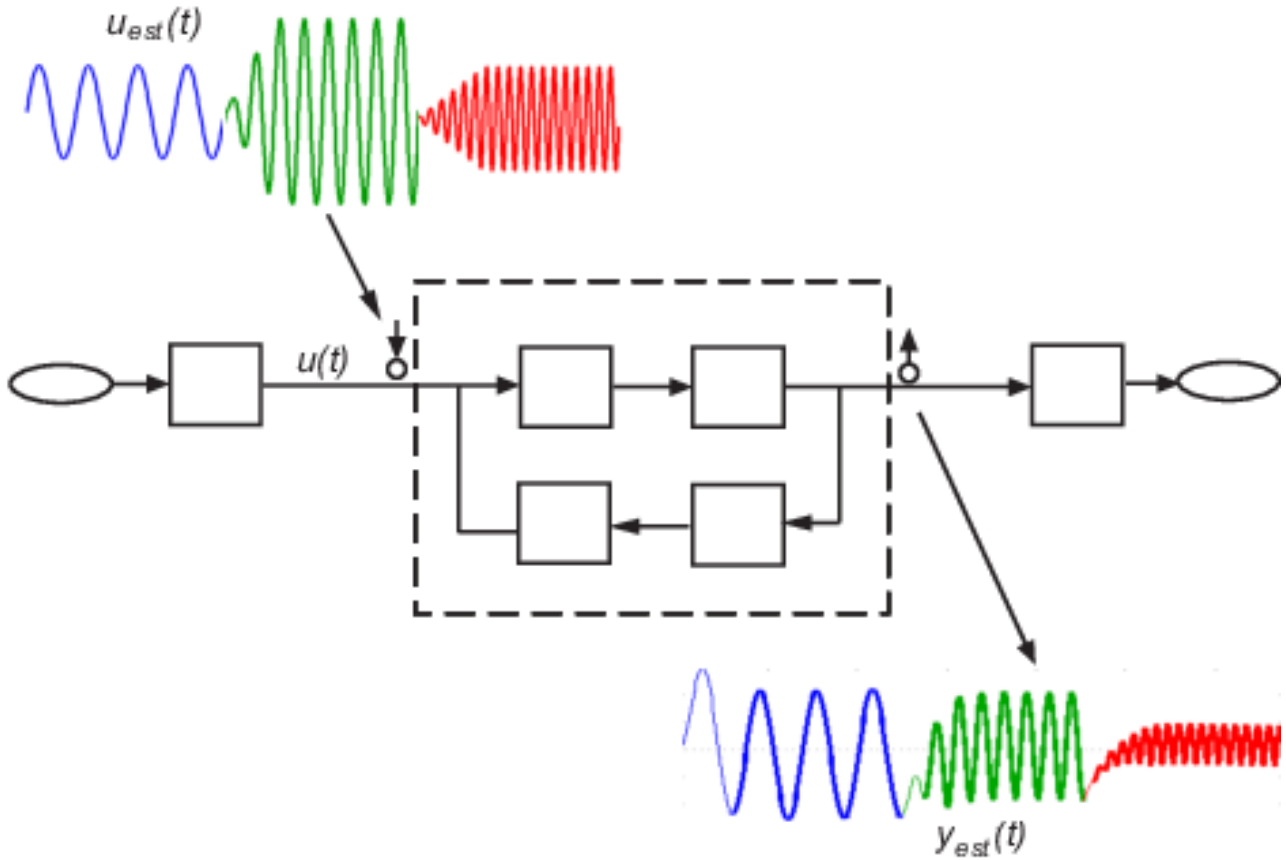



$$\sum_i A_i \sin \omega_i t$$

# 推定アルゴリズム

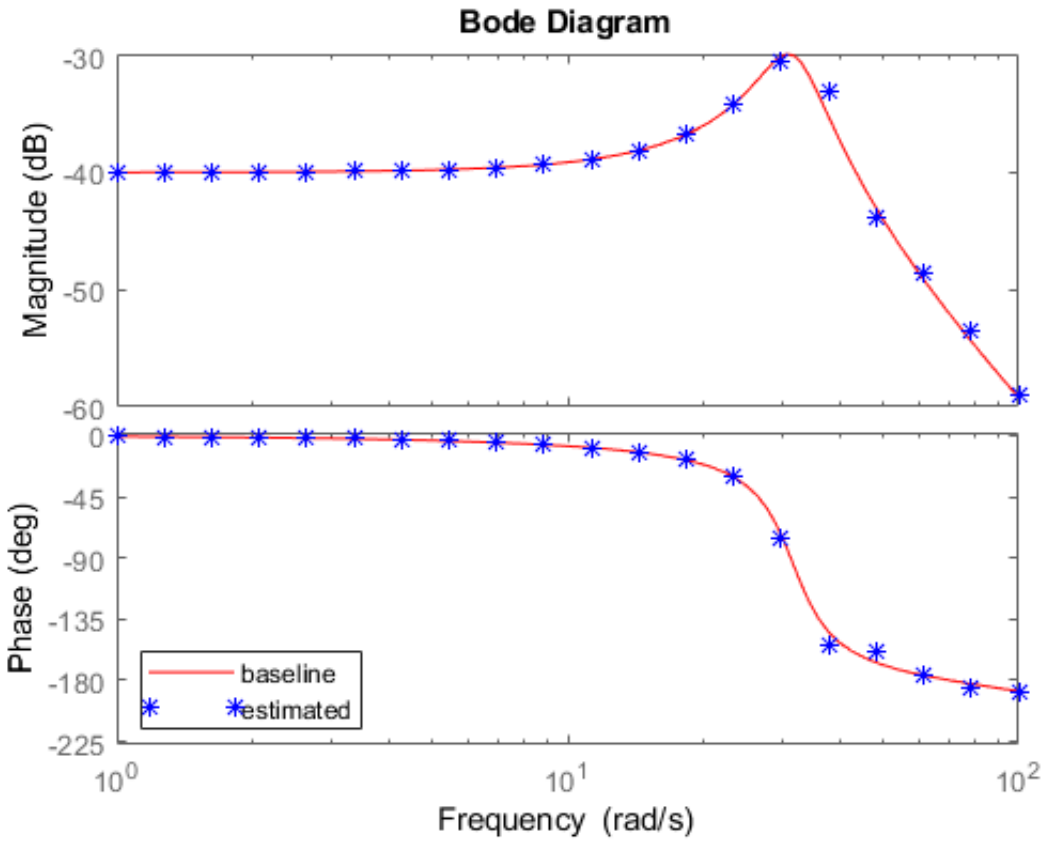
高速フーリエ変換の出力信号と入力信号の比率を計算することで推定する。

$$R_{esp} = \frac{FFT\left(y_{est}(t)\right)}{FFT\left(u_{est}(t)\right)}$$



# システム同定

周波数応答のデータから伝達関数を得る。




$$\frac{b_1 s + b_0}{a_2 s^2 + a_1 s + a_0}$$

# アジェンダ

- 背景
- 実験的な周波数応答推定によりプラントを同定
  - Frequency Response Estimatorブロック
  - モデル線形化器
- 最適化アルゴリズムによる制御ゲイン最適化
  - 応答オプティマイザー

# 多入力多出力（MIMO）システムに対して周波数応答推定を行う

## 入力

**解析入出力**

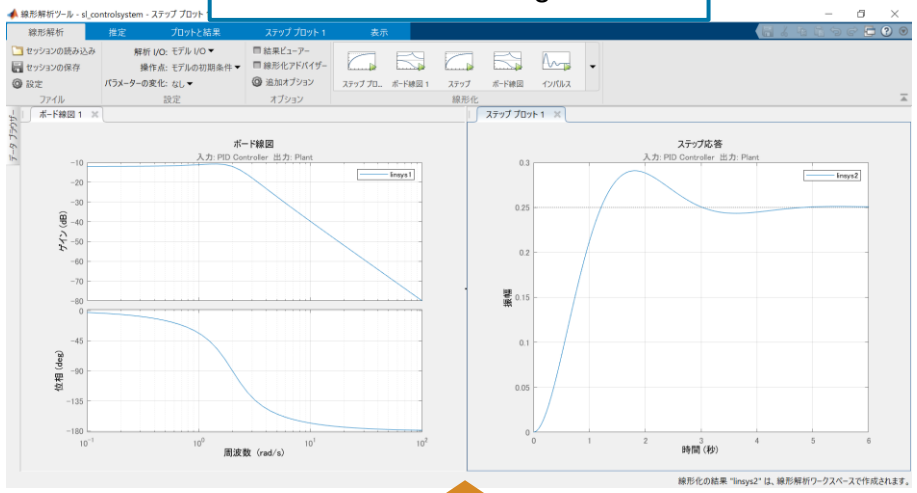
任意の入出力信号

**解析手段**

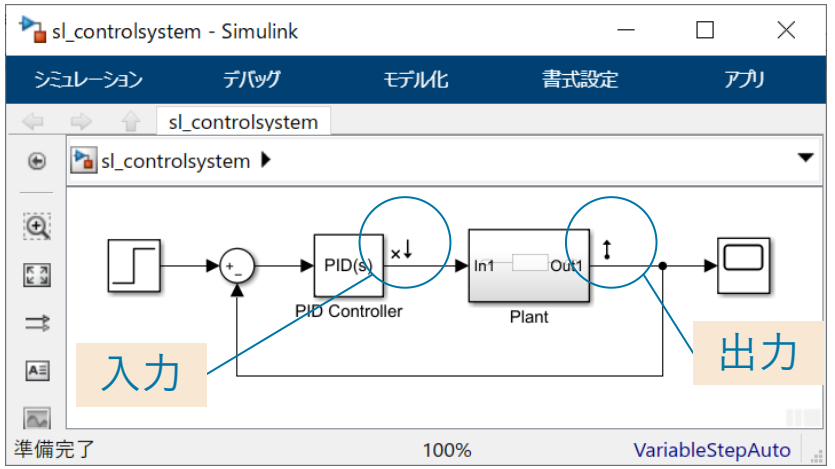
プロット選択

| 時間/周波数応答 |
|---|
| ステップ |
| インパルス |
| ボード線図 |
| ナイキスト線図 |
| ニコルス線図 |
| 特異値 |
| 極/零点 |

**モデル線形化器**
Simulink Control Design



モデル線形化




## 出力

**解析プロット**

制御対象の動特性

**線形化モデル**

線形時不変モデル

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

### 使いどころ

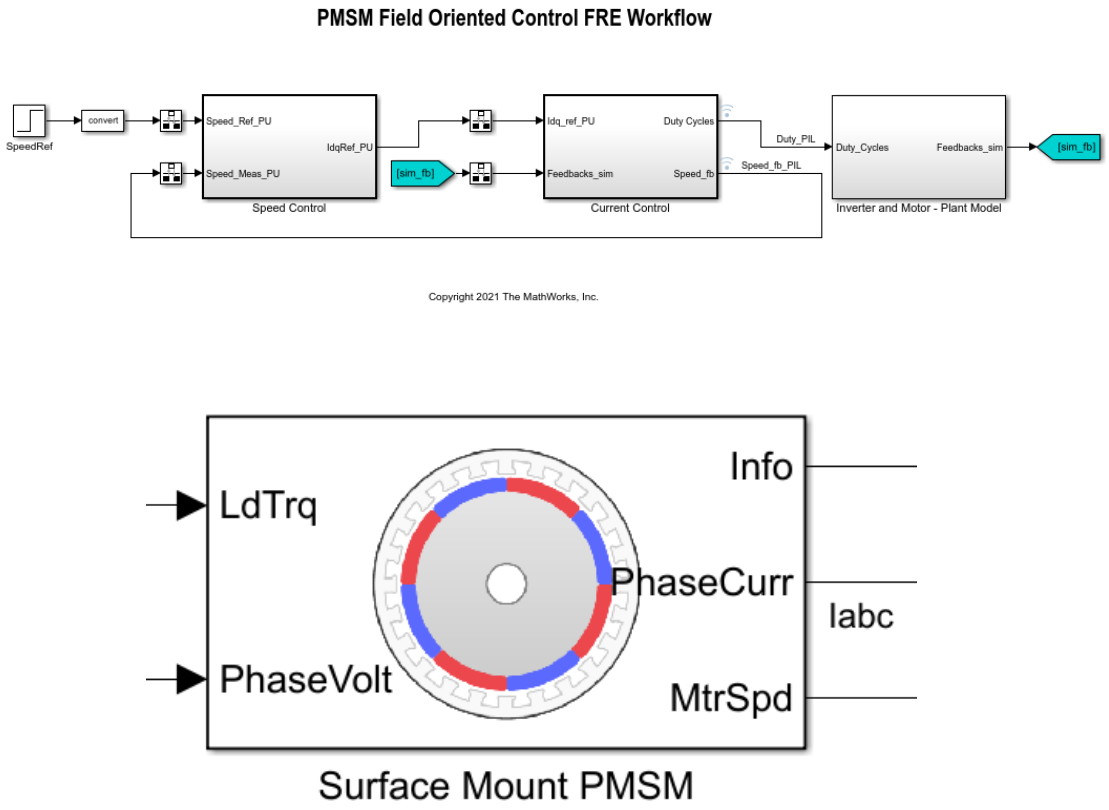
- 制御対象の性格の把握
- 制御系の性能・安定性の把握

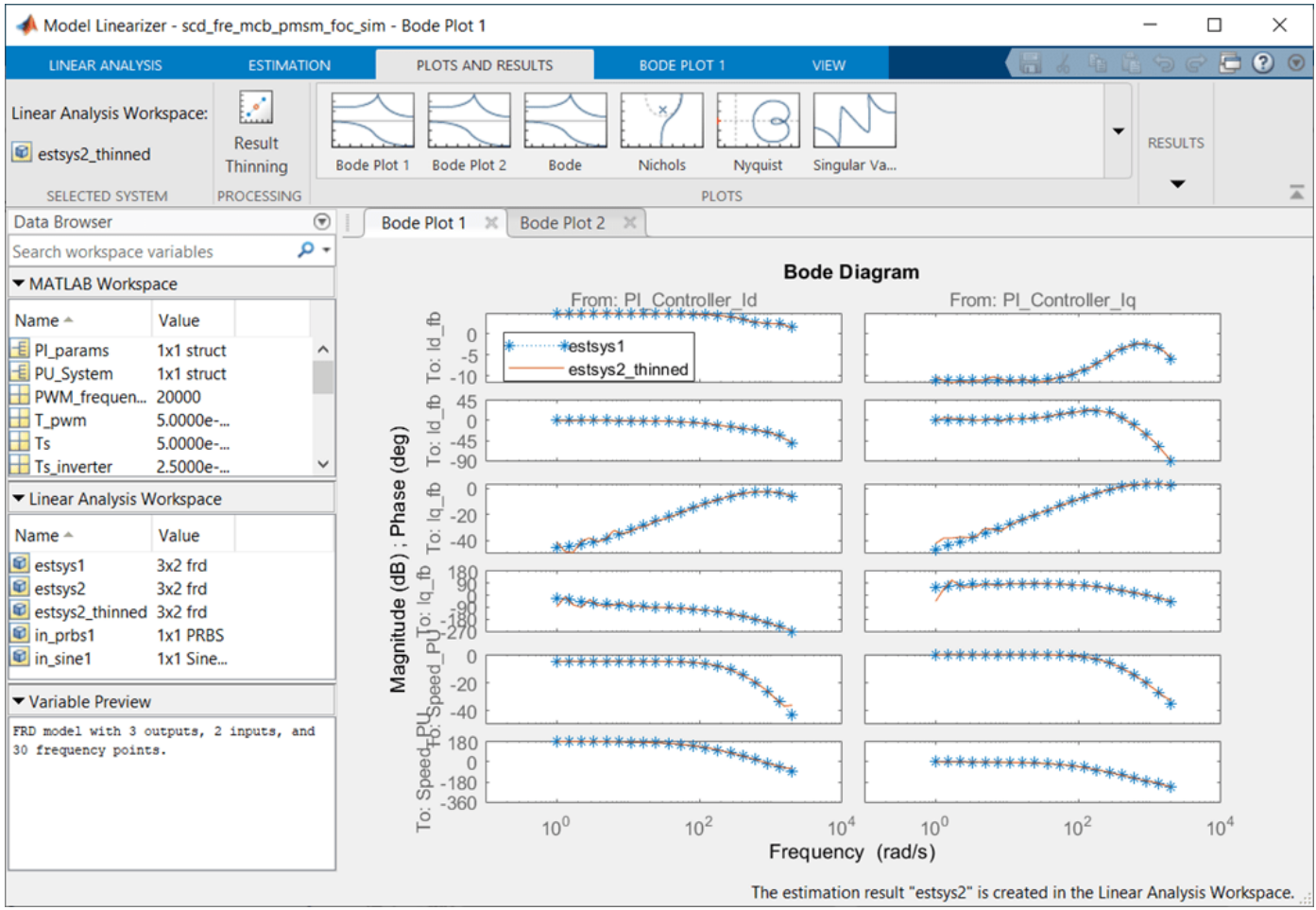
### デメリット

- 推定機能をコード生成して実機実装することができない

# 【デモ】永久磁石同期モーターの周波数応答の推定

操作量「d, q軸電圧」に対する制御量「d, q軸電流、回転数」の周波数特性を得る。



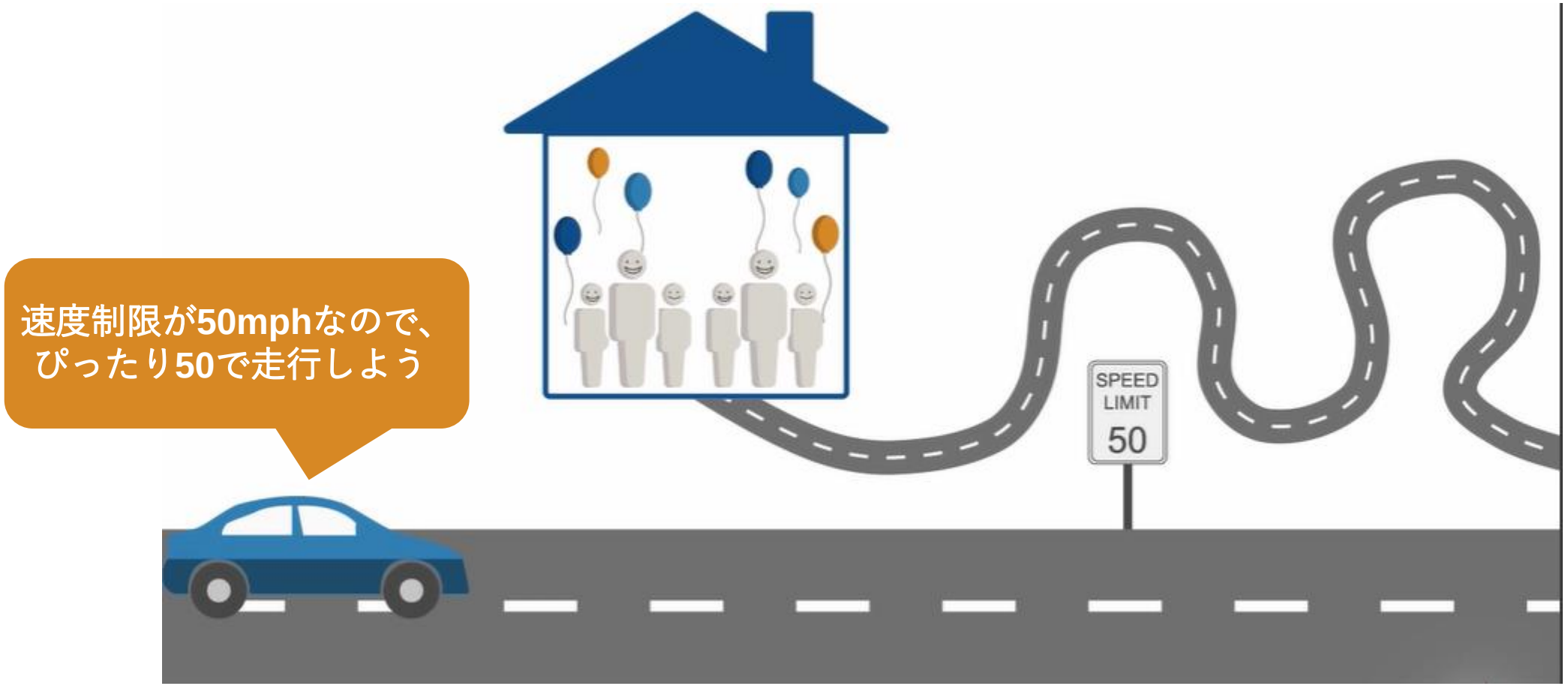

https://mathworks.com/help/slcontrol/ug/frequency-response-estimation-for-pmsm-model.html

# アジェンダ

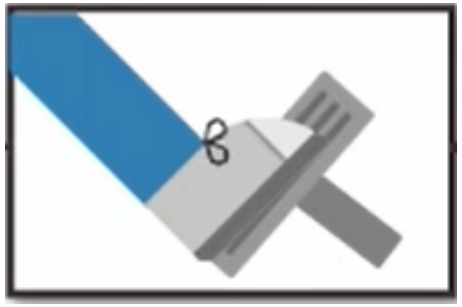
- 背景
- 実験的な周波数応答推定によりプラントを同定
  - Frequency Response Estimatorブロック
  - モデル線形化器
- 最適化アルゴリズムによる制御ゲイン最適化
  - 応答オプティマイザー

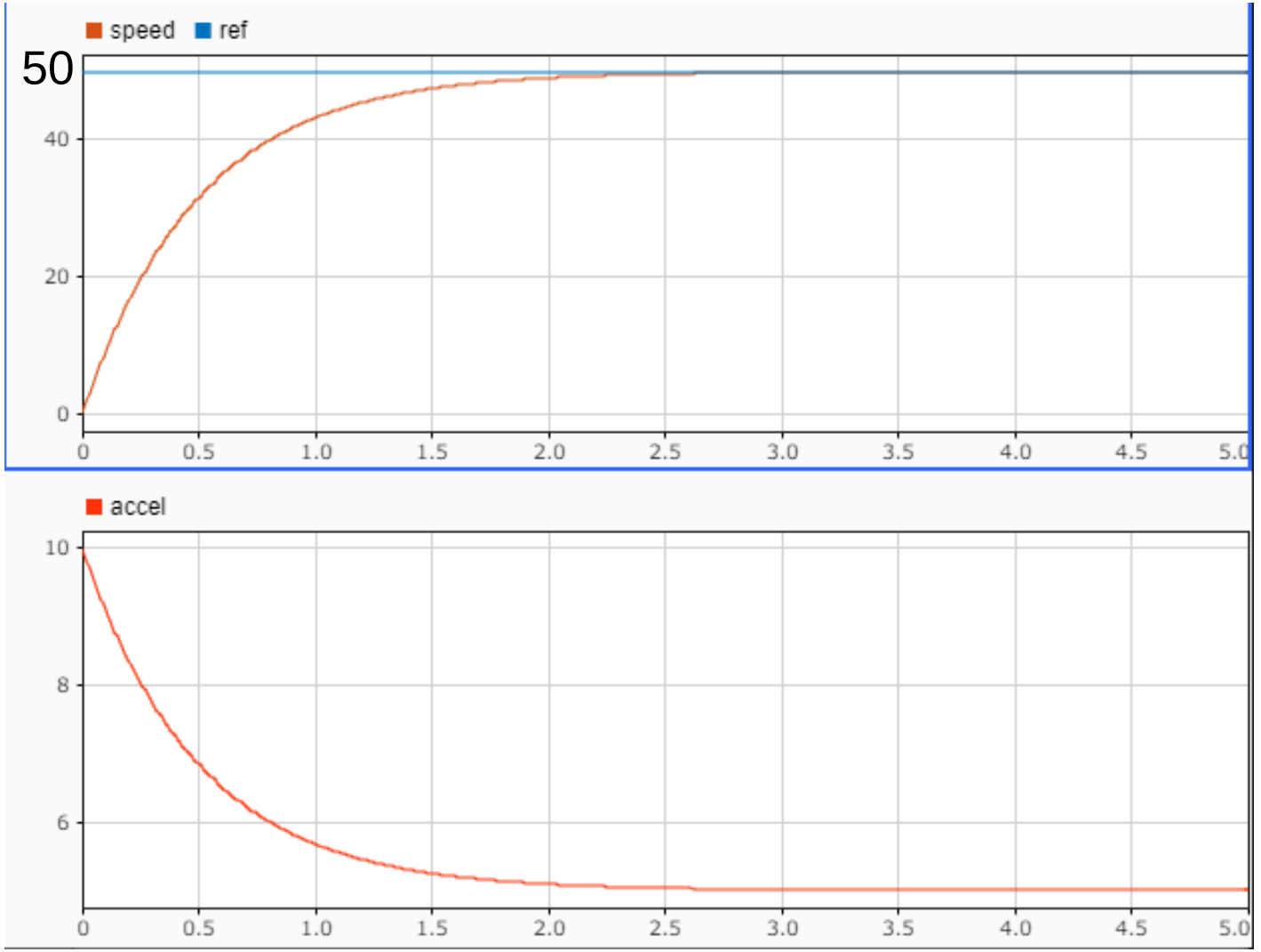
# 制御とは

- 制御対象の値を目標値に近づける、または可能なら一致させること

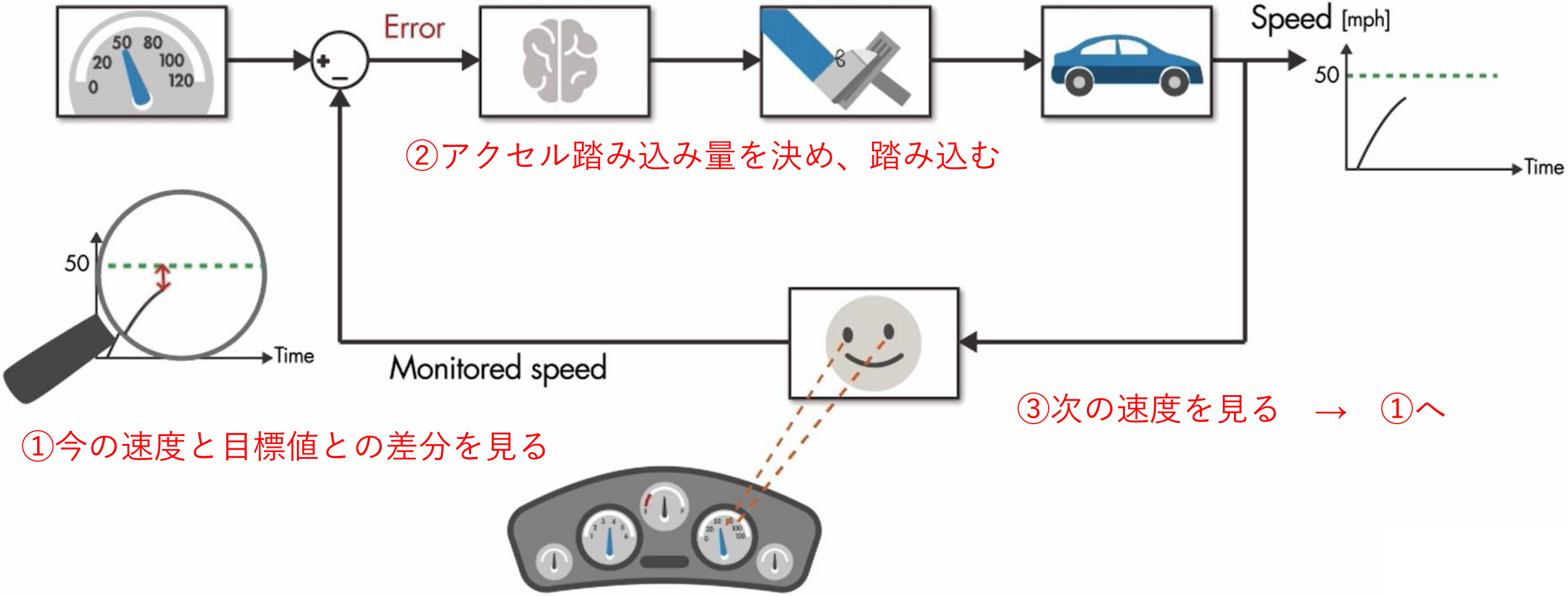


# どうすれば50mphになるか

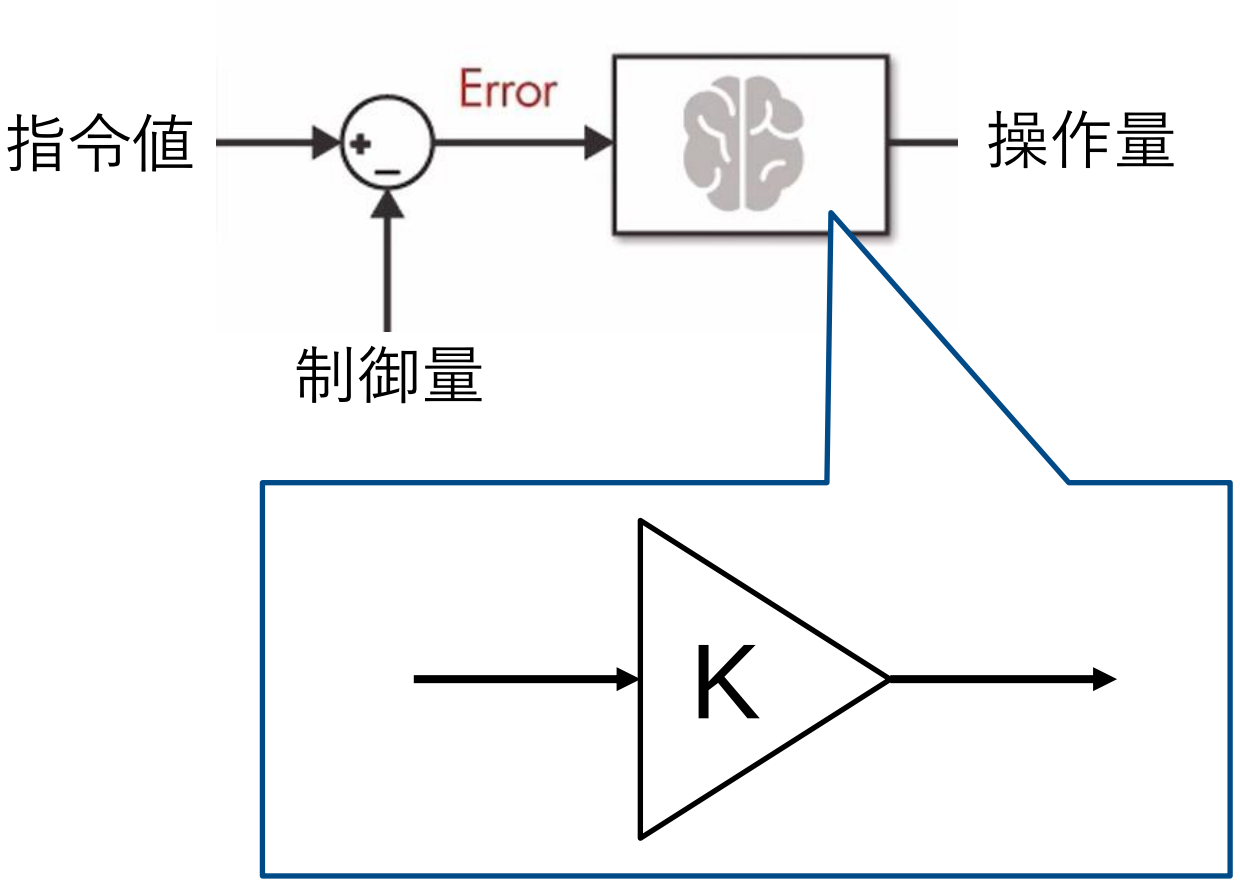




# 自動的に目標値に追従させる計算アルゴリズム
## 以下のような形式をフィードバック制御と呼ぶ

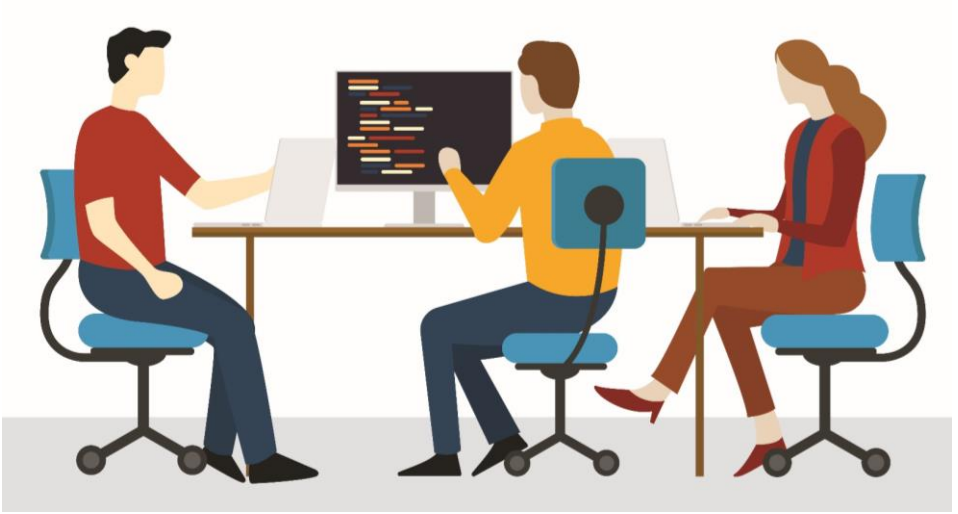


# フィードバック制御




差分に対してゲインKを掛け算し、
それを操作量として出力する。

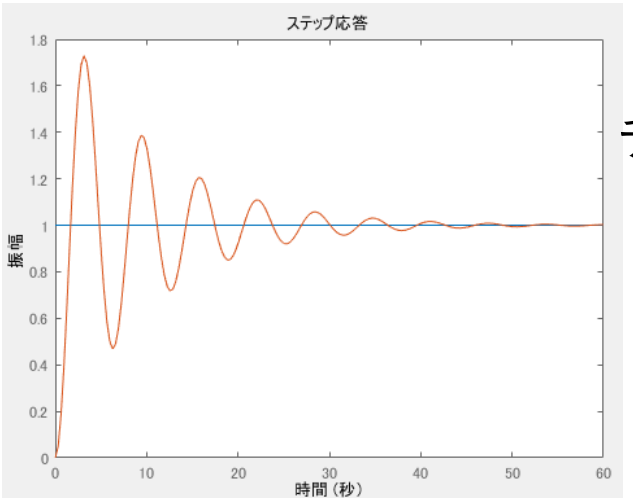
Kを上手く調整して、
望む応答を得たい！



# 【参考】制御性能評価

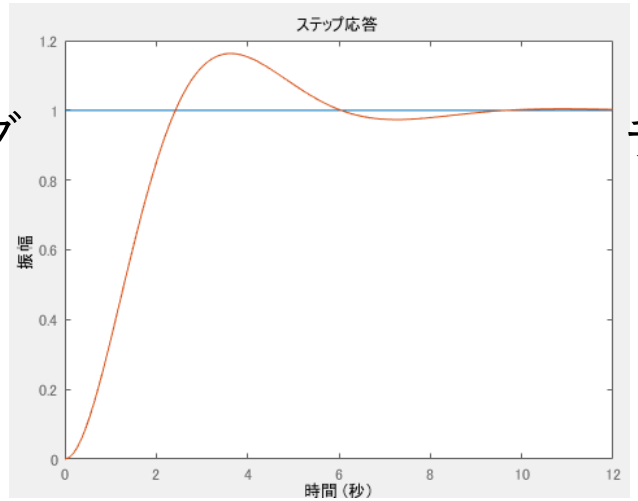
- 制御性能が良い　＝　指令値にピッタリ追従できる
- よく用いられる評価方法は、ステップ応答を見る方法

安定がよくないが、
指令値追従が速い



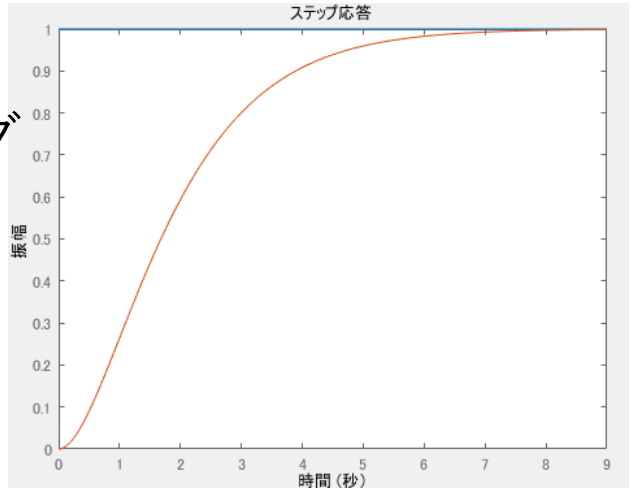

チューニング

ちょうどよい




チューニング

安定しているが、
指令値追従が遅い

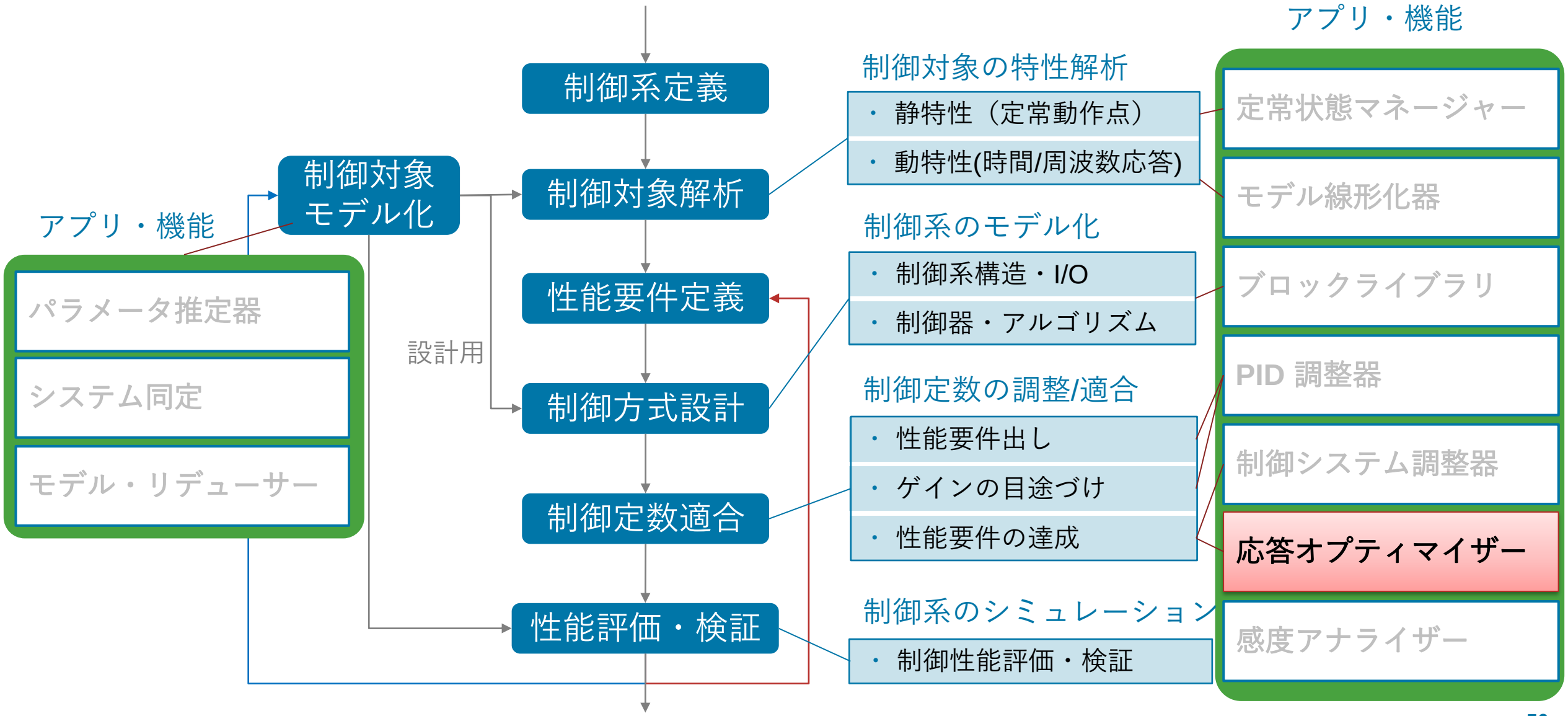



【重要】指令値に完全追従する、理想的な制御はこの世に存在しない

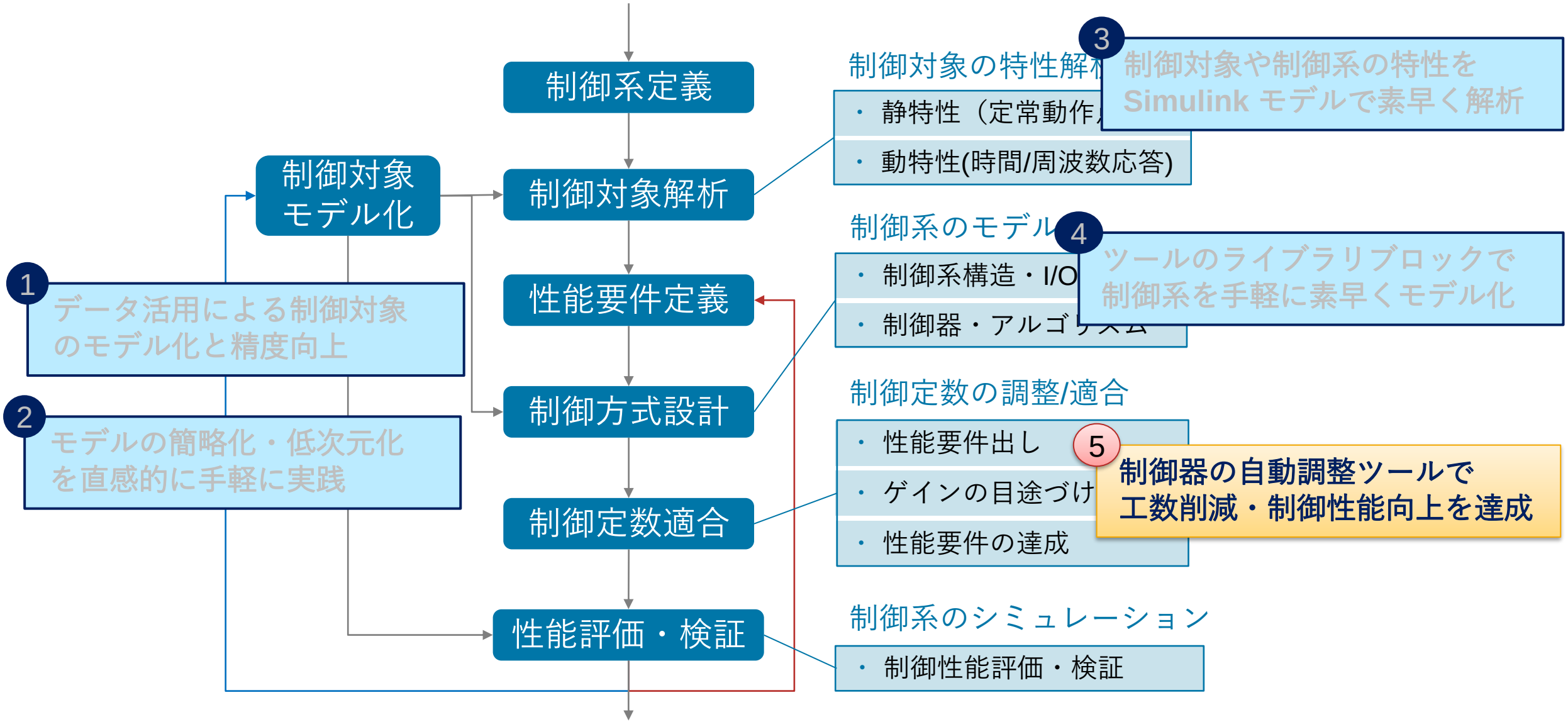
# アジェンダ

- 背景
- 実験的な周波数応答推定によりプラントを同定
  - Frequency Response Estimatorブロック
  - モデル線形化器
- 最適化アルゴリズムによる制御ゲイン最適化
  - 応答オプティマイザー

# 使うツールの立ち位置


制御系定義
制御対象解析
性能要件定義
制御方式設計
制御定数適合
性能評価・検証
制御対象モデル化
設計用
アプリ・機能
パラメータ推定器
システム同定
モデル・リデューサー
制御対象の特性解析
・静特性（定常動作点）
・動特性(時間/周波数応答)
制御系のモデル化
・制御系構造・I/O
・制御器・アルゴリズム
制御定数の調整/適合
・性能要件出し
・ゲインの目途づけ
・性能要件の達成
制御系のシミュレーション
・制御性能評価・検証
アプリ・機能
定常状態マネージャー
モデル線形化器
ブロックライブラリ
PID 調整器
制御システム調整器
応答オプティマイザー
感度アナライザー

# 扱うソリューション


制御系定義
制御対象解析
性能要件定義
制御方式設計
制御定数適合
性能評価・検証
制御対象
モデル化
1
データ活用による制御対象
のモデル化と精度向上
2
モデルの簡略化・低次元化
を直感的に手軽に実践
制御対象の特性解析
・静特性（定常動作）
・動特性(時間/周波数応答)
3
制御対象や制御系の特性を
Simulink モデルで素早く解析
制御系のモデル
・制御系構造・I/O
・制御器・アルゴリズム
4
ツールのライブラリブロックで
制御系を手軽に素早くモデル化
制御定数の調整/適合
・性能要件出し
・ゲインの目途づけ
・性能要件の達成
5
制御器の自動調整ツールで
工数削減・制御性能向上を達成
制御系のシミュレーション
・制御性能評価・検証

# 最適化アルゴリズムを用いて制御ゲインを自動調整

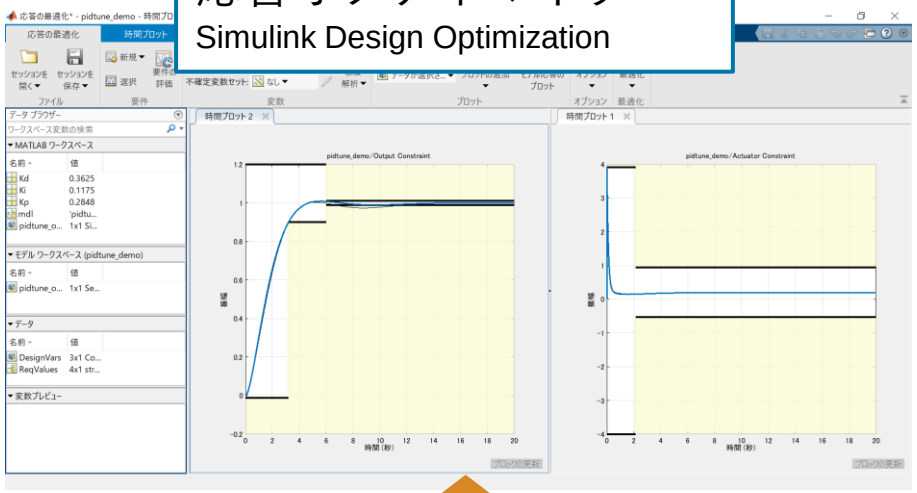
## 入力

**制御定数指定**

モデルパラメータ
（ワークスペース変数）

**目標性能仕様**

## 応答オプティマイザー
Simulink Design Optimization



自動調整

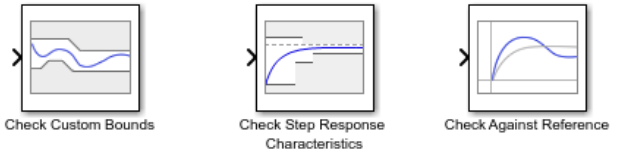
## 出力

**調整結果**

- パラメータ値
- 応答性能可視化
- 最適化進行状況レポート

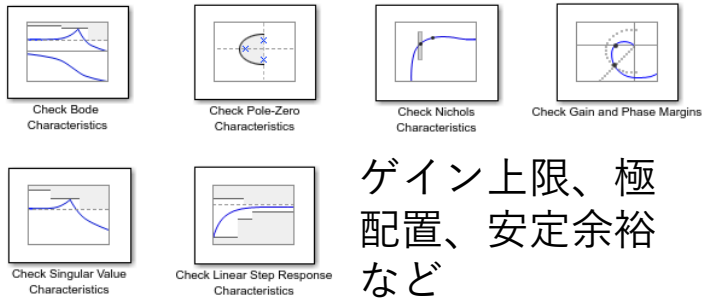
### 性能仕様定義ブロック

時間応答

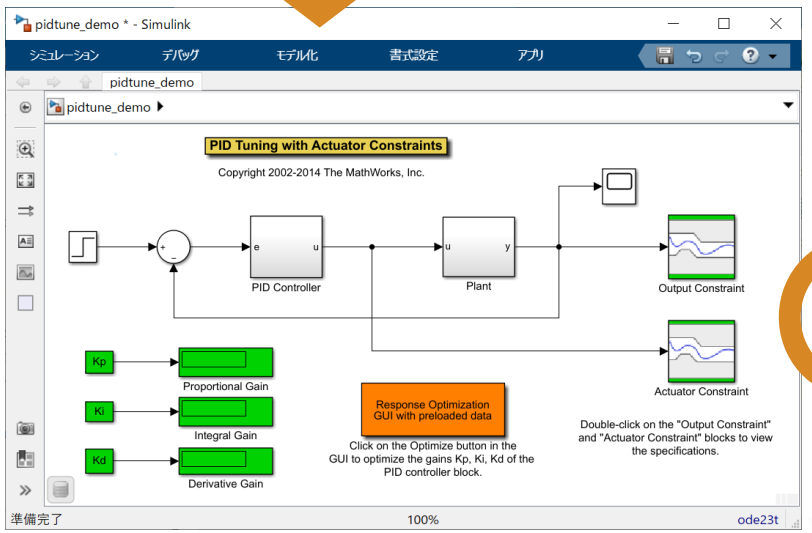



上下限　ステップ応答目標軌道

周波数応答




ゲイン上限、極配置、安定余裕など




### 使いどころ

- 制御性能仕様決め
- 制御性能仕様を満足する制御定数の適合

高速リスタート、並列化にも対応

数値最適化技術

**Optimization Toolbox**

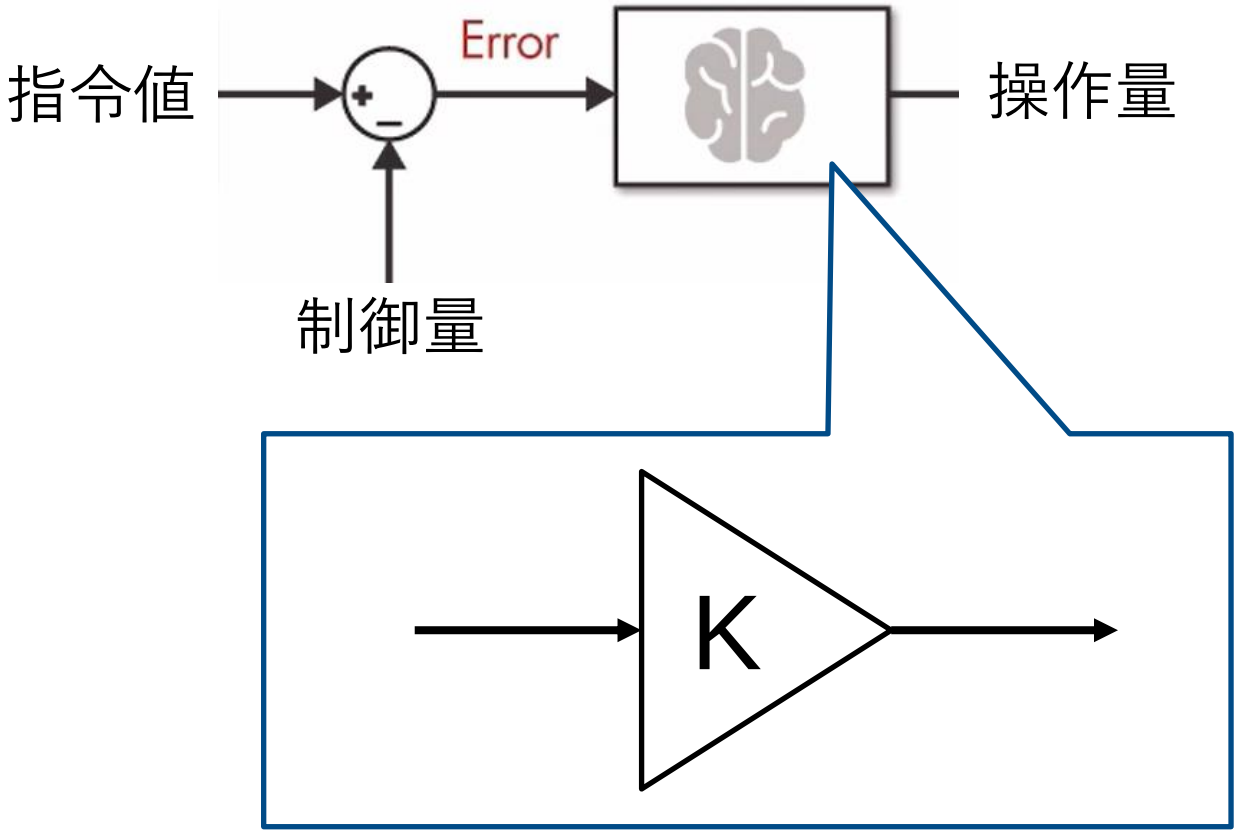
# 【デモ】 Quanser “QUBE – Servo 2” の振れ止め制御

真下に向いた振り子の振動を抑えながら、振り子の位置を制御する。




振れを意識せずにやると、

# フィードバック制御をどのように設計するか

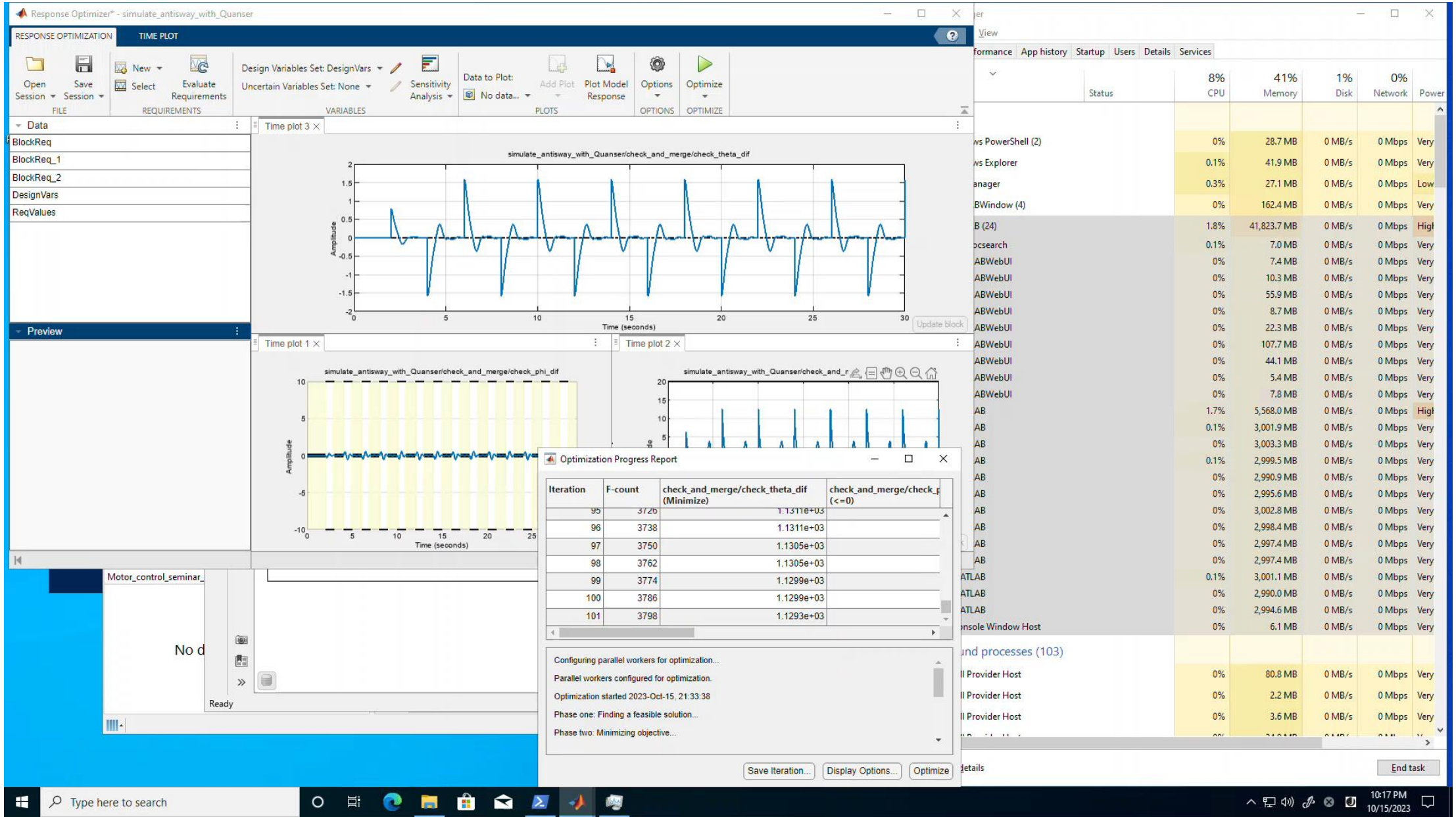



- 指令値
  - モーター角度
  - 振り子角度
- 制御量
  - モーター角度
  - 振り子角度
  - モーター角速度
  - 振り子角速度
- 操作量
  - モーター電圧
- ゲイン
  - ？

変化率が測定できているのであれば、是非制御に活用しましょう。

# 応答オプティマイザーで調整した結果

最適化にかかった時間：約45分



# 【参考】応答オプティマイザーを実行したPCの性能

## Device specifications

| | |
|---|---|
| Device name | |
| Processor | Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz 2.30 GHz (2 processors) |
| Installed RAM | 120 GB |
| Device ID | |
| Product ID | |
| System type | 64-bit operating system, x64-based processor |
| Pen and touch | |

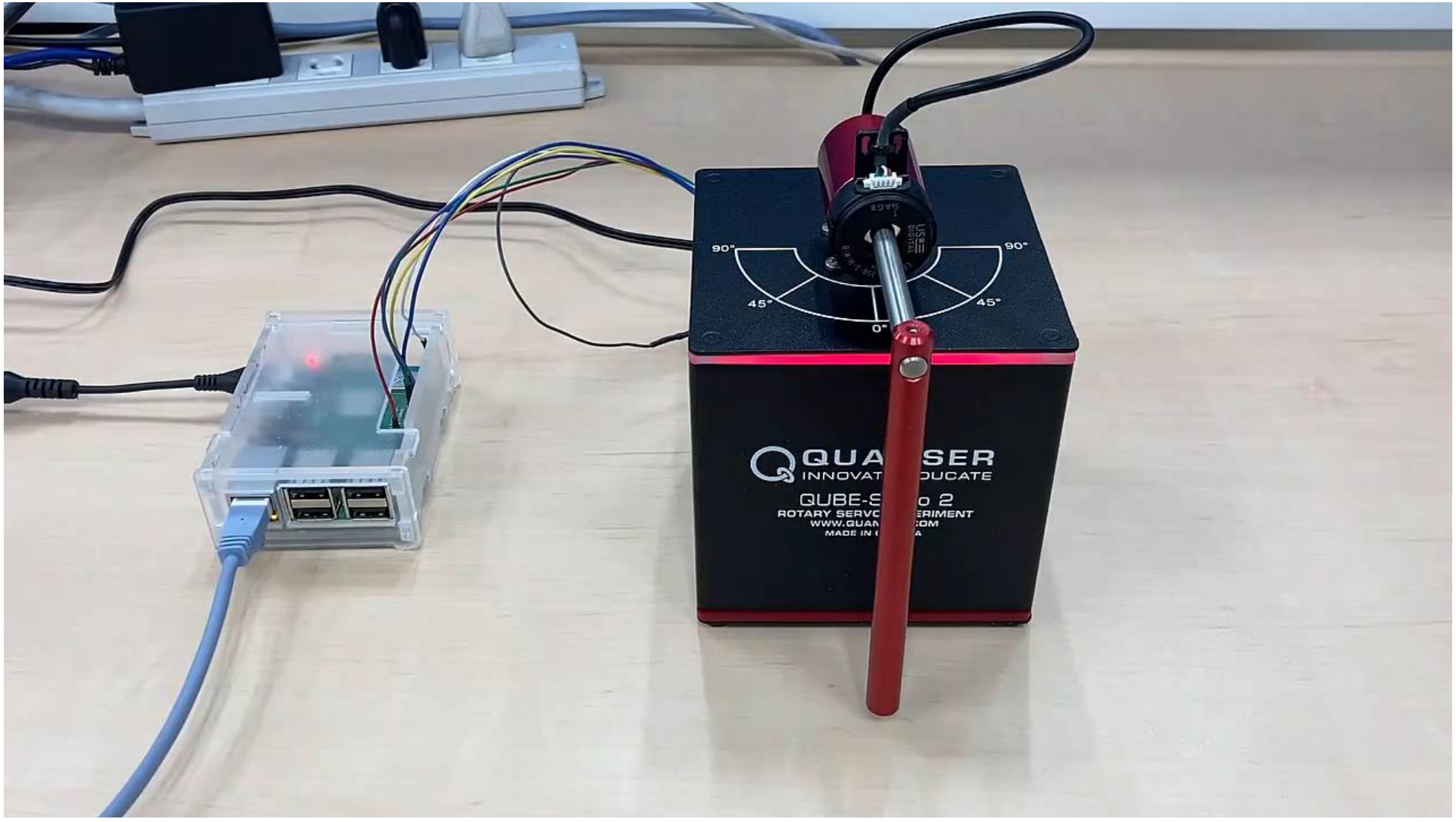
Rename this PC

## Windows specifications

| | |
|---|---|
| Edition | Windows 10 Enterprise |
| Version | 1909 |

# 調整したゲインを使って実機試験

https://www.mathworks.com/videos/model-based-design-improves-power-electronics-and-motor-drive-systems-development-efficiency-1699973729138.html



# 実機試験

angle_ref: 角度指令値[rad]
semsor_sim(1): モーター角度（シミュレーション）[rad]
semsor_exp(1): モーター角度（実機）[rad]

semsor_sim(2): 振り子角度（シミュレーション）[rad]
semsor_exp(2): 振り子角度（実機）[rad]

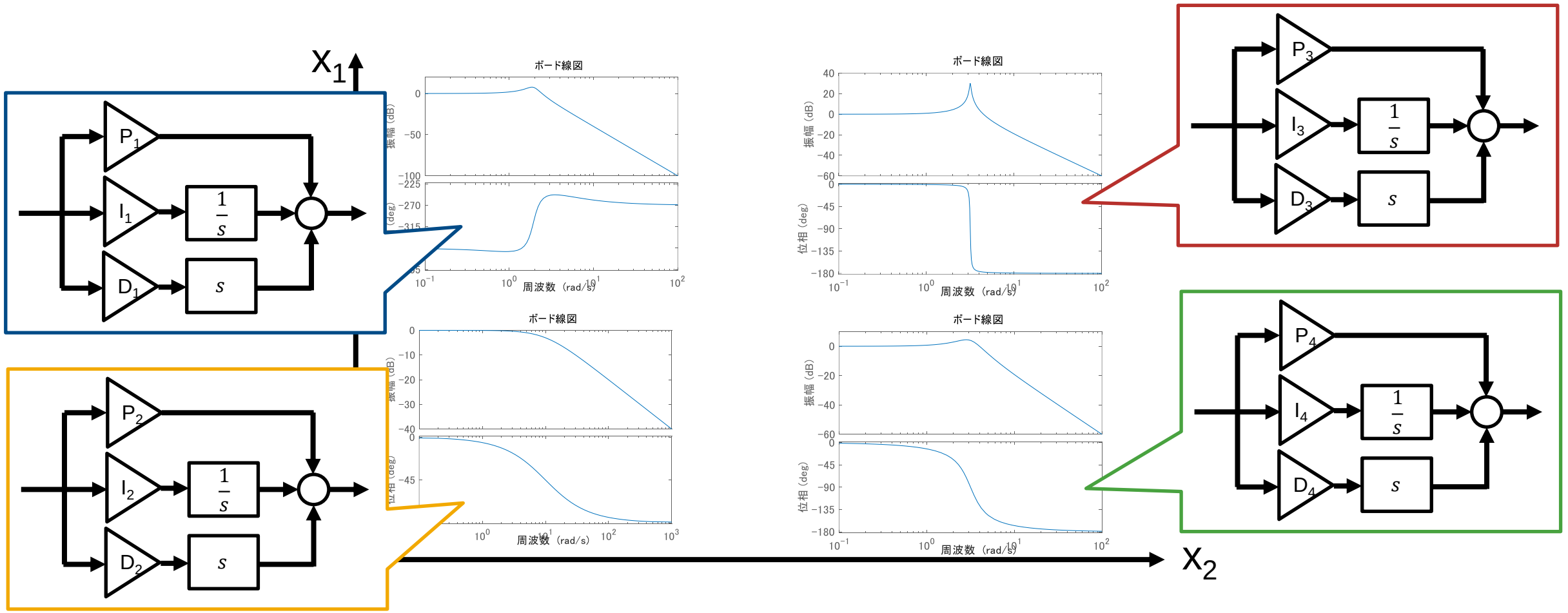
volt_sim: 電圧指示値（シミュレーション）[V]
volt_exp: 電圧指示値（実機）[V]

# 結果の考察

- ある程度プラントモデルが正確であれば、プラントモデルに対して調整したゲインをそのまま実機に適用できる
- プラントモデルが正確であればあるほど良いが、制御目的によっては、完璧な精度を求める必要はない
- 最後のゲイン調整は実機で行う必要があるが、最初から実機でやるよりも、事前にシミュレーションで当たりを付けておく方が効率が良い

# ゲインスケジューリングPID制御

非線形システムを状態の領域ごとに区分し、その領域において最適なPIDゲインを設定する。領域間は線形補間などで繋ぐ。

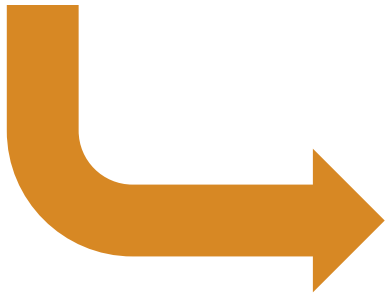


# まとめ

# まとめ

- Frequency Response Estimatorブロックを使って、実機からプラントモデルを同定できる
- 多入力多出力システムの場合は、モデル線形化器を使ってシステム同定できる

- 応答オプティマイザーを使って、最適化アルゴリズムによる制御ゲインの調整ができる



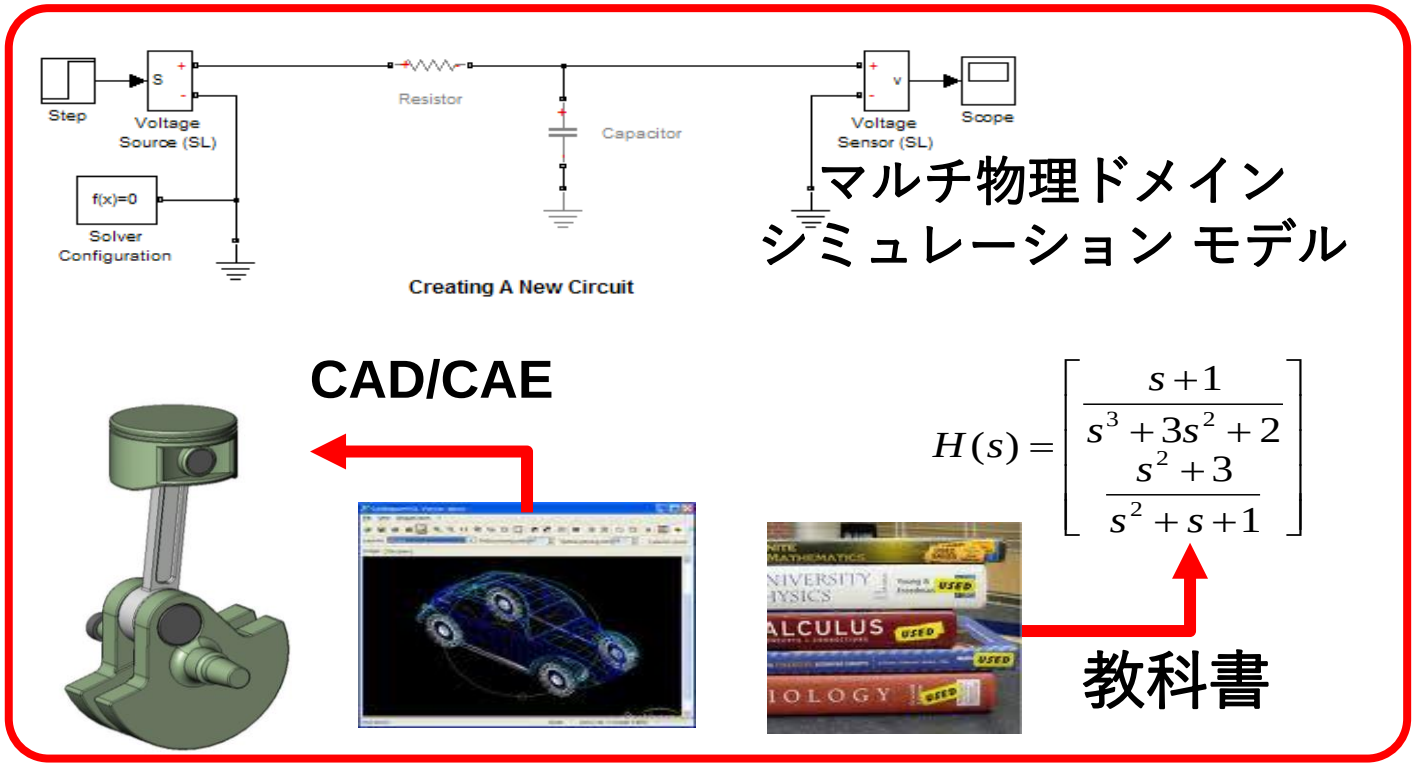
MATLABを使えば、制御理論に関する知見がなくても、すぐに制御設計を始められます！

# 次のステップ

# プラントに対する知見は、あった方がよい

理由：知見があれば、より良いプラントモデルを効率的に作れる

- 数式からモデリング
- ニーズに合わせて詳細度の調整
- パラメーターの当たりづけ
- 分野独特のセオリーが使える



# 制御理論に対する知見は、あった方がよい

理由：制御系設計業務を効率化でき、より高品質な制御を実現できる

- 古典制御
  - 制御システムの安定性解析、ゲイン位相余裕
- 現代制御
  - 多入力多出力の制御器設計、センサーで計測できない状態をリアルタイム推定（カルマンフィルター）
- ロバスト制御
  - モデル化誤差を許容し、制御性能を担保
- モデル予測制御
  - PID制御では実現できない制御応答を実現

# 補足資料

# MathWorks 制御系設計ツール一覧

| | 製品 | 概要 |
|---|---|---|
| 1 | Control System Toolbox | 線形制御理論（古典/現代）に基づく制御系のモデリング・解析・設計 |
| 2 | System Identification Toolbox | ブラックボックス（システム同定）、グレーボックスモデリング |
| 3 | Robust Control Toolbox | 不確かさをもつプラントに対するロバスト制御系解析・設計 |
| 4 | Model Predictive Control Toolbox | モデル予測制御の設計・シミュレーション |
| 5 | Fuzzy Logic Toolbox | ファジーロジック系の設計・シミュレーション |
| 6 | Simulink Control Design<br>線形制御理論 | Simulink モデルを起点とした制御系解析・設計<br>・ Frequency Response Estimatorブロック<br>・ 制御システム調整器 |
| 7 | Simulink Design Optimization<br>数値最適化 | Simulink モデルのパラメータ自動調整<br>・ パラメーター調整器<br>・ 応答オプティマイザー |
| 8 | Predictive Maintenance Toolbox | 状態監視・予知保全向けアルゴリズムの設計・評価 R2018a |
| 9 | Reinforcement Learning Toolbox | 強化学習による方策（Policy）の設計・学習 R2019a |

# トレーニング

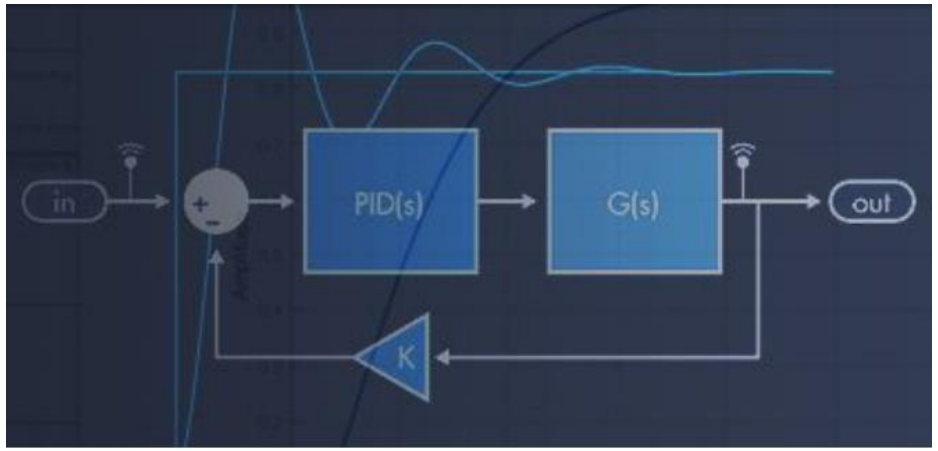
- Simulinkによる制御設計
  - https://mathworks.com/learn/training/control-system-design-with-simulink.html

## コースの詳細

この2日コースではモーター駆動システムを取り上げ、MATLAB と Simulink を使用した閉ループ制御システムの設計方法（モデル化、制御設計、検証）について学びます。本コースでは MATLAB、Simulink の他に、主に Control System Toolbox、System Identification Toolbox、Simulink Control Design、Simulink Design Optimization を使用します。

詳しくはコース概要をご確認ください。

- 制御システム設計の概要
- システムモデリング
- システム解析
- 制御設計
- コントローラーの実装



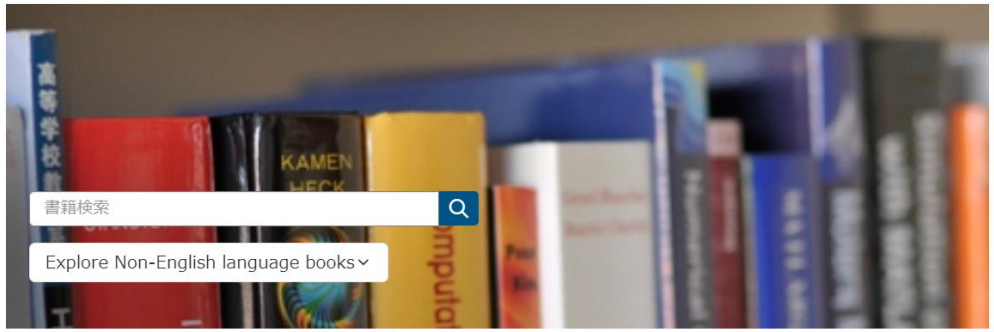
# MATLAB/Simulink 関連書籍を検索

https://www.mathworks.com/academia/books.html

MathWorks® 製品 ソリューション アカデミア サポート コミュニティ イベント MATLAB を入手する

MATLAB/Simulink 関連書籍 書籍検索 書籍

関連書籍トップページ | 検索 | Book Program に参加する




## 2000 冊以上の教員、学生、専門家向け書籍

MathWorks のツールの研究や開発用途での普及を反映して、MathWorks 製品の関連書籍の数も増加を続けています。内容には、MATLAB、Simulink などの MathWorks 製品に関する理論、実世界での実践例、および演習が含まれています。エンジニアリング、科学、金融、数学など各分野の講師が使用する教材にもなり、大学や産業界において信頼できる参考文献として活用されています。

### MATLAB 入門 - zyBook

このウェブベースの書籍では、インタラクティブな質問やアニメーション、自動評価を通して、MATLAB の基礎全般を学べます。

結果： 1 - 25 / 299

**MATLAB/Simulinkによる制御工学入門**
はじめて制御工学を学ぶ読者に最適な、わかりやすい入門書です。電気系と機械力学系を中心として、例や問題を豊富に収録するとともに、図やグラフを多く用いて、理解しやすいよう解説を工夫されています。
**作成者:** 川田 昌克
**Copyright:** 2020
**言語:** 日本語
**出版社:** 森北出版
**使用可能なコンパニオン ソフトウェア**

**自動車業界MBDエンジニアのためのSimulink入門: 4週間で学ぶSimulink実践トレーニング 第2版**
本書は、自動車業界で新たにモデルベース開発(MBD) を始めるエンジニアにお勧めの Simulink の入門書です。自動車業界の実際の開発業務で使うことを目的に、基礎的な物理法則の復習モデルからトルクコンバーターやクルーズコントロールといった実製品の機能モデルまで Simulink で作成していきます。
**作成者:** 久保 孝行
**Copyright:** 2019
**言語:** 日本語
**出版社:** TechShare
**使用可能なコンパニオン ソフトウェア**

## 分野

- Artificial Intelligence, Machine Learning, and Deep Learning
- 生物科学および生物医学
- 化学および化学工学
- 通信システム
- 制御システム
- Data Science and Statistics
- デジタル信号処理
- 地球科学
- 経済学および金融工学
- エレクトロニクス
- 総合
- 画像および動画処理
- 数学
  - 微分方程式
  - 線形代数
  - 数値的手法
- 機械工学
- ニューラル ネットワークおよびファジー論理
- 神経科学
- 物理学
- プログラミングとコンピューター サイエンス
- ロボット工学
- システム同定
- システム モデリングとシミュレーション
- 実験、計測
- MATLAB および Simulink の使用

# 制御理論を勉強できる書籍

| 分類 | コンテンツ |
| --- | --- |
| 書籍 | 川田 昌克 ： MATLAB/Simulinkによる制御工学入門, 森北出版 (2020)<br>https://www.mathworks.com/academia/books/introduction-to-control-engineering-with-matlab-simulink.html<br><br>内容： はじめて制御工学を学ぶ読者に最適な、わかりやすい入門書です。<br>電気系と機械力学系を中心として、例や問題を豊富に収録するとともに、<br>図やグラフを多く用いて、理解しやすいよう解説を工夫されています。  |
| | 南 裕樹, 石川 将人： 制御系設計論, コロナ社 (2022)<br>https://www.mathworks.com/academia/books/control-system-design.html<br><br>内容： モノの動きをデザインするための方法をまとめた制御工学の教科書です。<br>制御工学のスペシャリストとしての素養を身につけたい、断片的に知っている制御工学の<br>知識を整理したい、辞書的な書籍を手元においておきたい、そんな人におすすめです。  |
| | 川田 昌克, 西岡 勝博： MATLAB/Simulink によるわかりやすい制御工学 第2版, 森北出版 (2022)<br>https://www.mathworks.com/academia/books/understanding-control-engineering-with-matlab-simulink.html<br><br>内容： 電気・機械系の例を中心とし、伝達関数に基づく時間応答/周波数応答による<br>特性解析だけでなく、具体的なコントローラの設計例を通じて、「制御工学」を<br>理解できます。各章の章末には MATLAB、Simulink、Control System Toolbox、<br>Symbolic Math Toolbox を利用した演習があります。  |

# 制御理論を勉強できる書籍

| 分類 | コンテンツ |
| --- | --- |
| 書籍 | 平田 光男： ArduinoとMATLABで制御系設計をはじめよう! 第2版, TechShare (2022)<br>https://www.mathworks.com/academia/books/getting-started-in-control-system-design-with-arduino-and-matlab.html<br><br>内容： 本書は、制御系設計をこれから学ぶ方のための入門書です。<br>Arduino® をベースにした安価な制御実験キットを使って、実際に制御系設計を<br>体験しながら学ぶことができます。実際に自分で制御対象の回路やメカ機構を作成し、<br>自作のコントローラーで制御することを体験できます。<br>  |
| MATLAB EXPO<br>公開スライド | 実例から学ぶ！ モデルを活用したモータ制御系開発<br>https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/images/events/matlabexpo/jp/2016/g1-motor-control-system-development.pdf |

# 深層強化学習による先進制御の実現

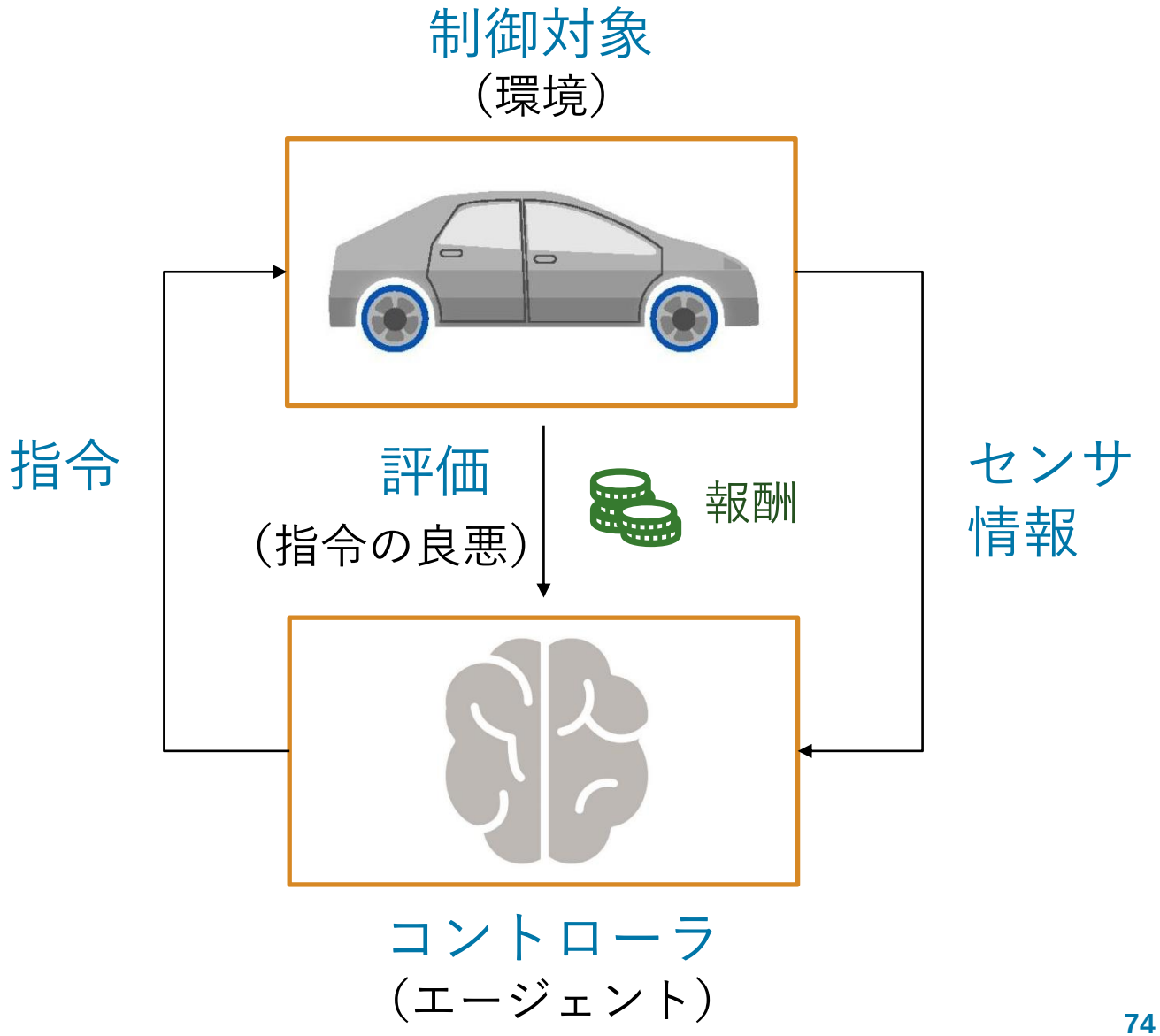
# 強化学習を用いて倒立振子を立たせる制御

# 強化学習は上手な制御方法を、試行錯誤しながら習得する仕組み

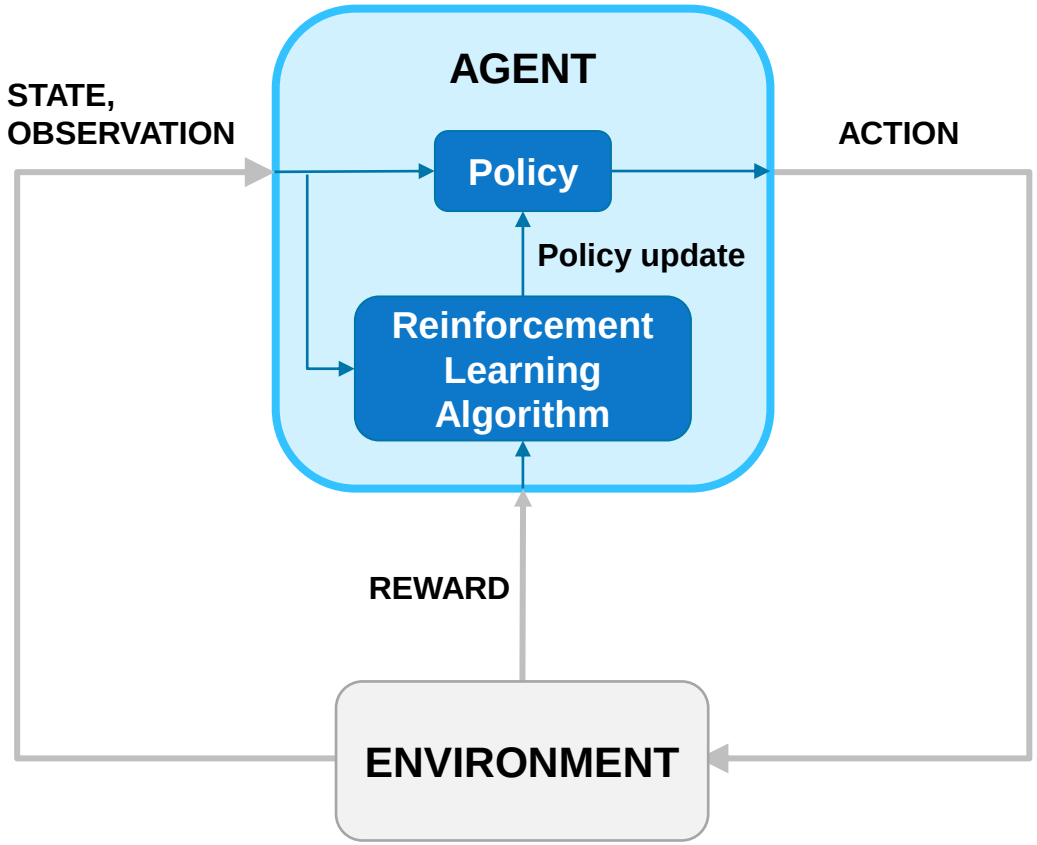
## 強化学習への期待

- 制御系の高度化・知能化
  - 自律性の獲得
  - 環境変化や不確実性への対応
  - 高い制御性能の達成
- 制御設計の自動化・省力化
  - 複雑な制御則を自動的に獲得
  - 制御則の再利用（再学習）

制御屋からすると、究極の制御の姿



# 強化学習　コンセプト
## 例：自動運転制御の場合

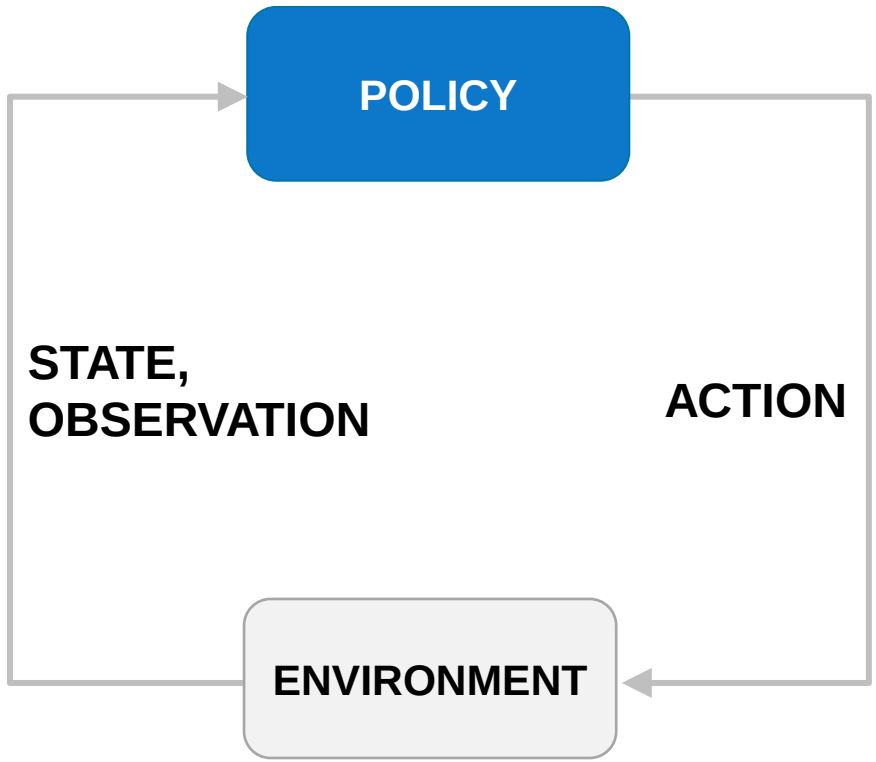



- 車両コントローラはどのように走るかを学習する
  - (**agent**)
- LIDARやカメラからのセンサー値
  - (**state, observation**)
- 路面状態や車両位置を表現する
  - (**environment**)
- ステアリング、ブレーキ、スロットル指令値
  - (**action**)
- (state)から次の(action)を生成する
  - (**policy**)
- ラップタイムや燃費効率などの最適化対象
  - (**reward**)

- **強化学習のアルゴリズム**により、ポリシーはトライ&エラーで更新される

# 強化学習　コンセプト
## 例：自動運転制御の場合

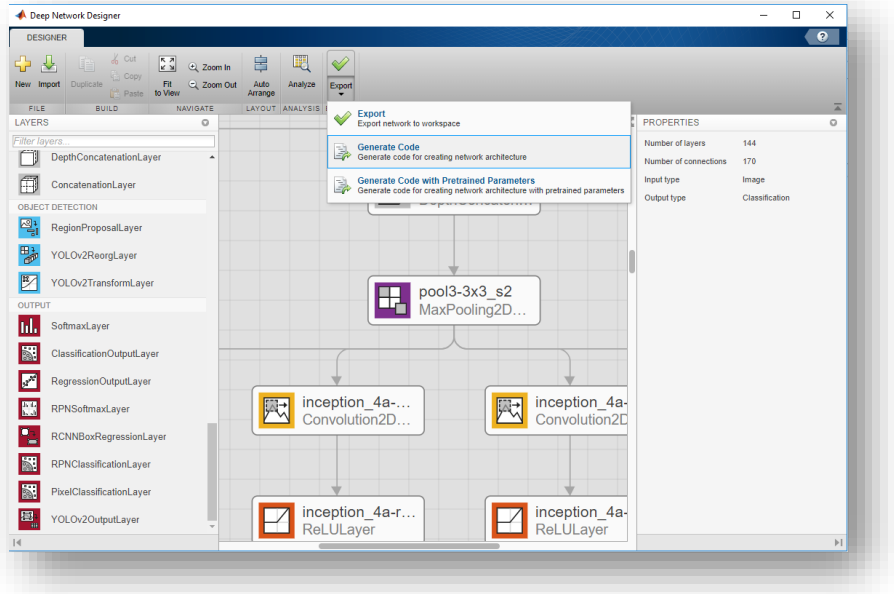
学習後は、学習済みポリシーのみ必要



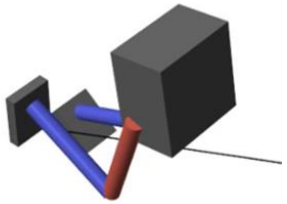

- 車両コントローラは学習後のアルゴリズムで(state)から(action)を得る
  - (**policy**)
- ステアリング、ブレーキ、スロットル指令値
  - (**action**)
- LIDARやカメラからのセンサー値
  - (**state, observation**)
- 路面状態や車両位置を表現する
  - (**environment**)

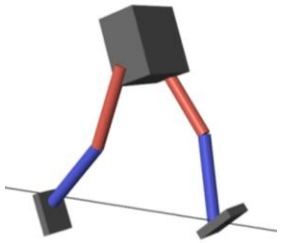
問題を定義することにより、この学習済みポリシーはラップタイムと燃費効率を最大化する

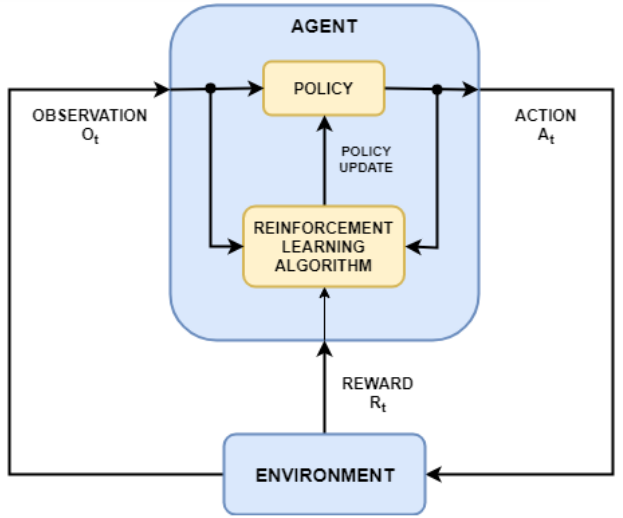
# Reinforcement Learning Toolbox™




- 強化学習のフローを網羅的にサポート
  - MATLAB 関数 / Simulink® モデルで表現された環境とのインターフェース
  - エージェント作成のためのネットワーク構築環境
  - 各種アルゴリズムを提供
    - DQN / Double DQN
    - SARSA
    - REINFORCE
    - DDPG / TD3
    - A2C / A3C
    - PPO
    - SAC
  - マルチエージェントに対応
  - 配布のための最適方策の関数化
  - レファレンス・アプリケーションを多数提供
    - https://www.mathworks.com/help/reinforcement-learning/examples.html



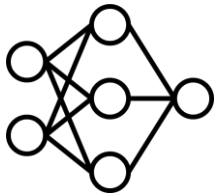




# 強化学習のワークフロー

Environment

Reward



Policy representation

Agent


Training

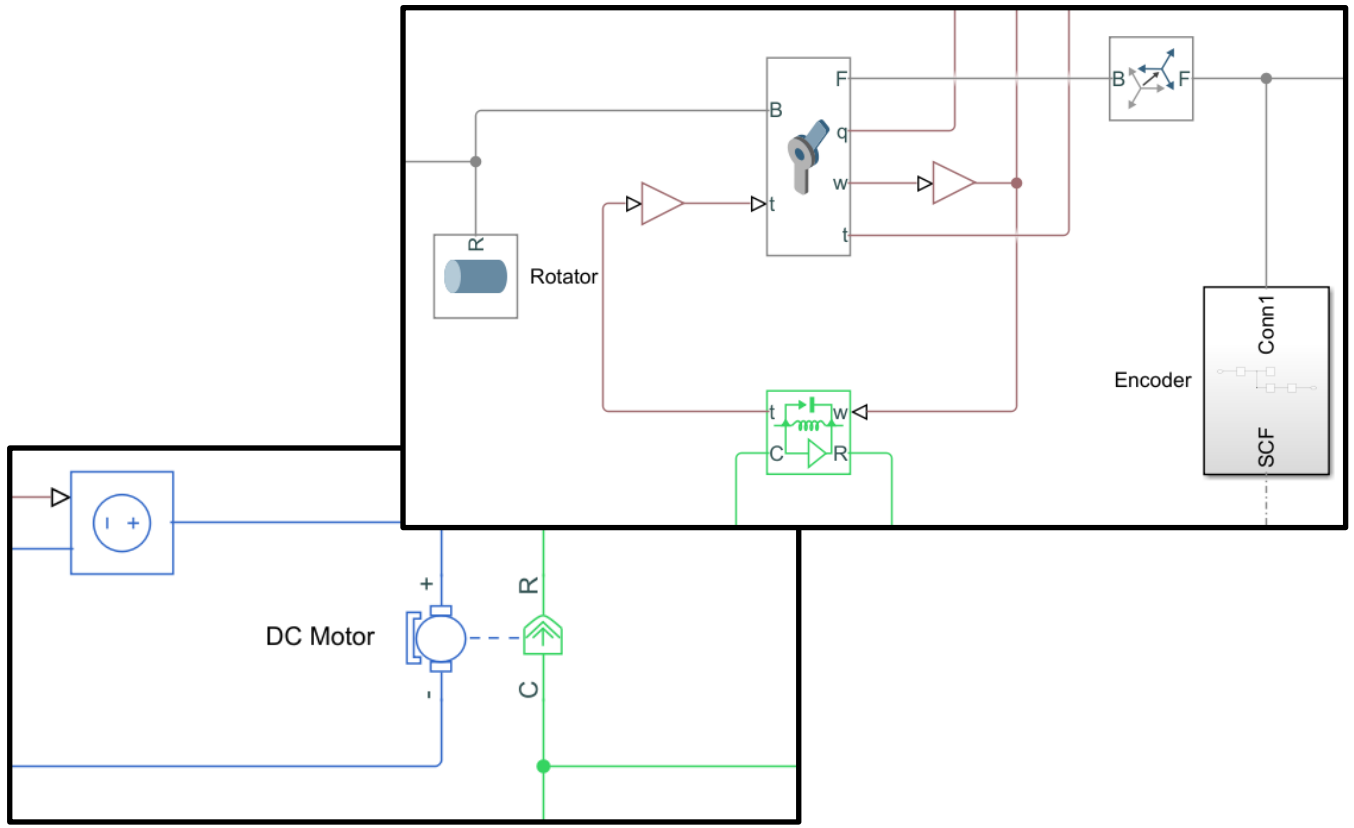
Deployment

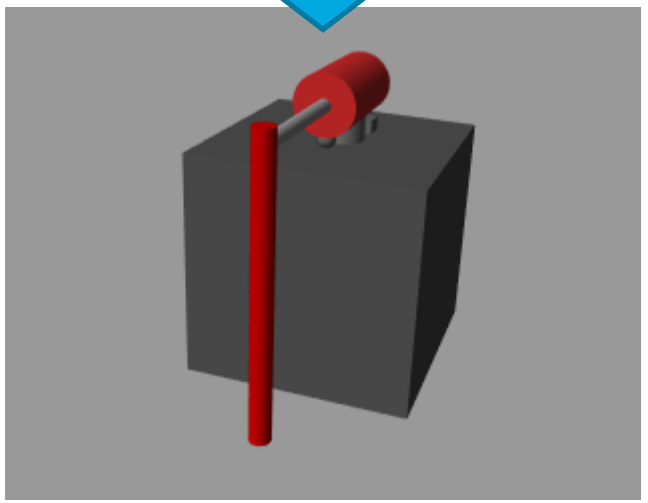
# 環境構築（プラントモデリング）


**Environment** Reward Policy representation Agent Training Deployment

# Simscapeによるモデル化

- DCモーターによって台座を回転させ、振り子にトルクを与えるシステムである
- Simscape Electrical™, Simscape Multibody™ を用いてモデル化が可能



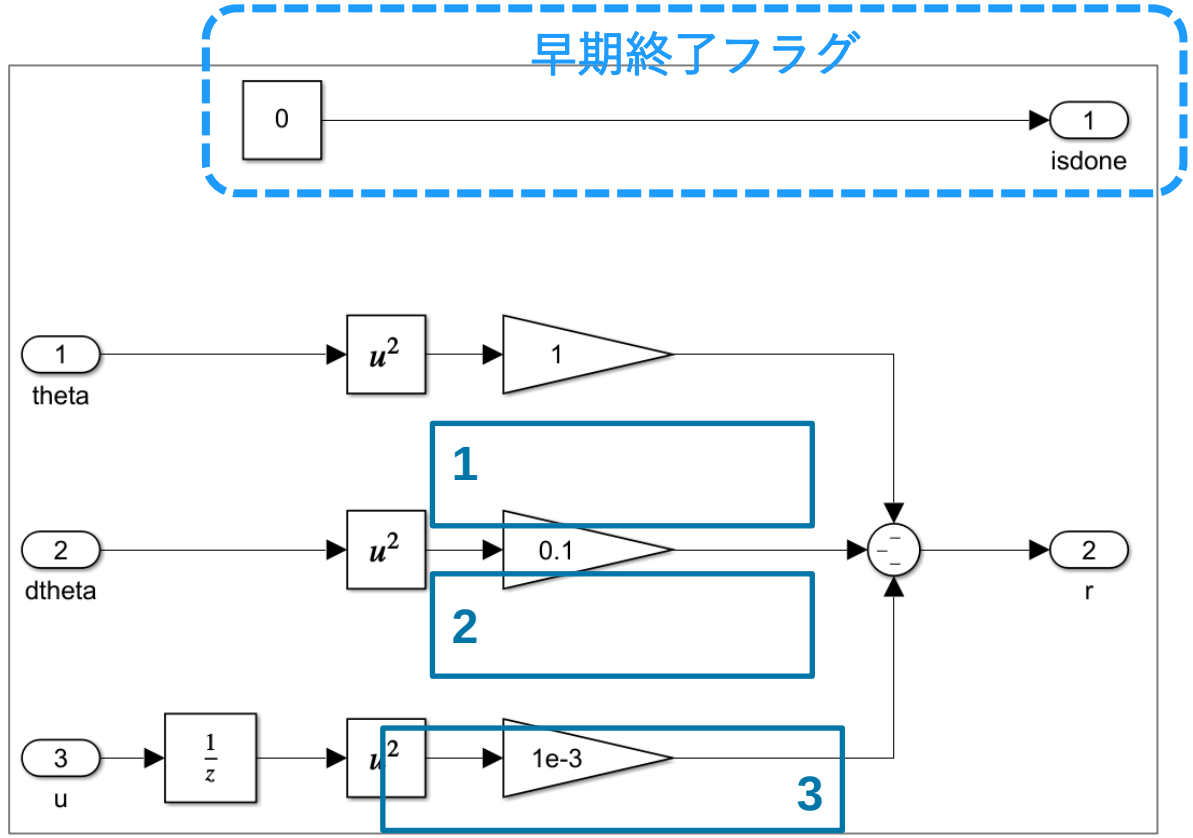
# 報酬設計


Environment **Reward** Policy representation Agent Training Deployment

# 報酬設計

- 報酬関数を数式で検討し、Simulinkモデル化

$$r_t := -\theta_t^2 - \frac{1}{10}\dot{\theta}_t^2 - \frac{1}{10^3}u_{t-1}^2 \qquad where\ u_{t-1} = \tau_{t-1}$$




数式以外にも、Stateflowを用いて状態遷移アルゴリズムを設計し、適用することも可能

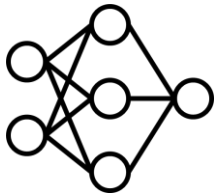
# ポリシー、エージェント構築



Environment



Reward



**Policy representation**

**Agent**




Training



Deployment

# エージェント構築

- サポートされるアルゴリズム

**Q-Learning**
**rlQAgent**

**DQN**
**Double DQN**
**rlDQNAgent**

**A2C /A3C**
**rlACAgent**

**PPO**
**rlPPOAgent**

**REINFORCE**
**(Policy Gradients)**
**rlPGAgent**

**DDPG**
**rlDDPGAgent**

**TD3**
**rlTD3Agent**

**SARSA**
**rlSARSAAgent**

**SAC**
**rlSACAgent**

# 学習



Environment





Reward





Policy representation



Agent




**Training**



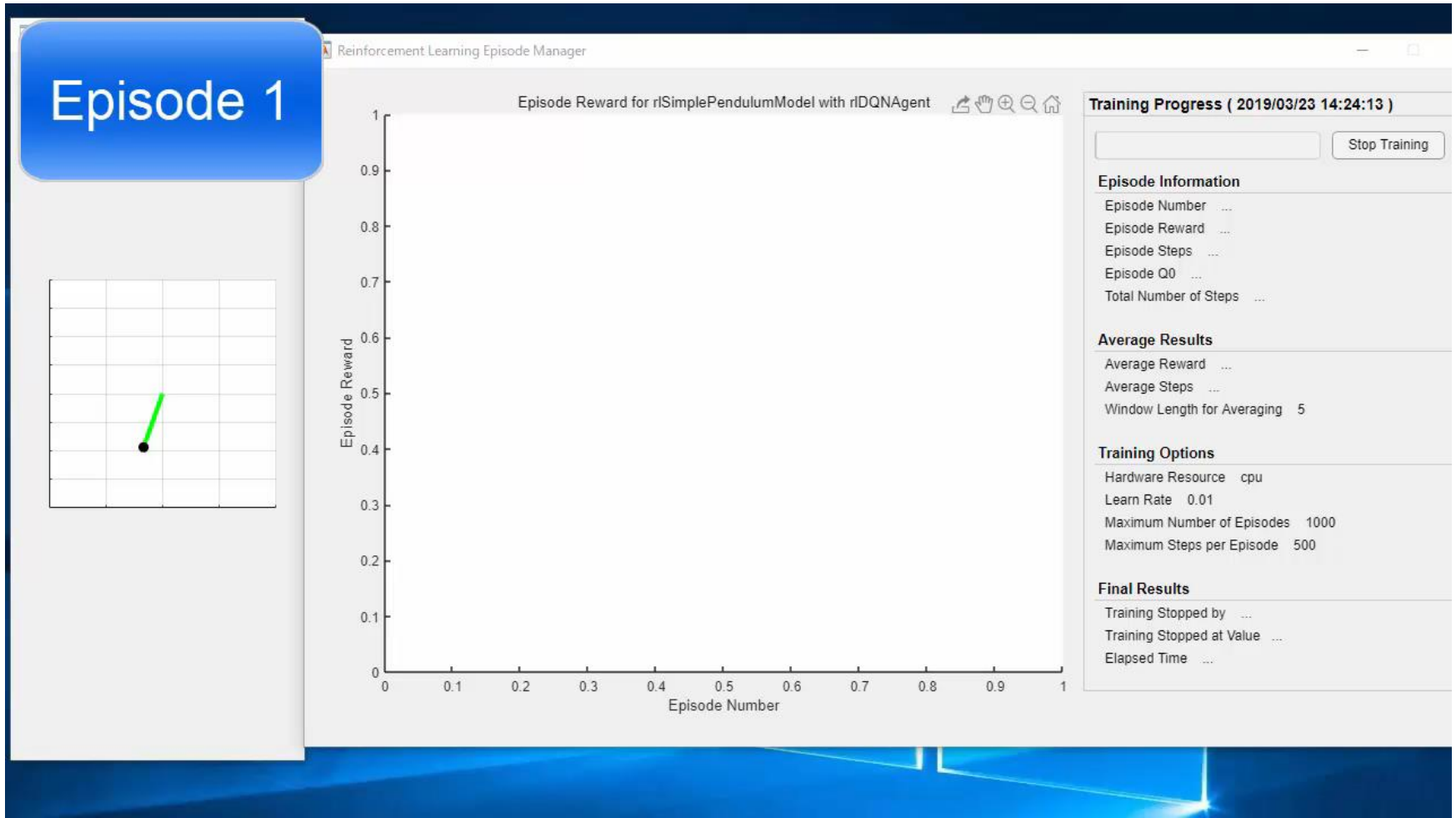


Deployment

# 学習 (最適化)

- 学習の実行

```
trainingStats = train(agent,env,trainOptions);
```

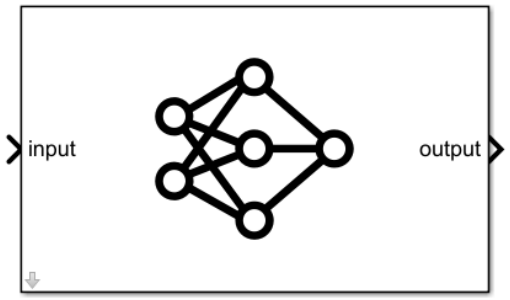


# 実装


Environment Reward Policy representation Agent Training **Deployment**

# 学習済みの深層ネットワークを展開する方法

学習済みネットワーク




Coder

GPU Coder

MATLAB Coder

cuDNN

TensorRT

MKL-DNN

ARM Compute

General

NVIDIA社GPU

NVIDIA社GPU

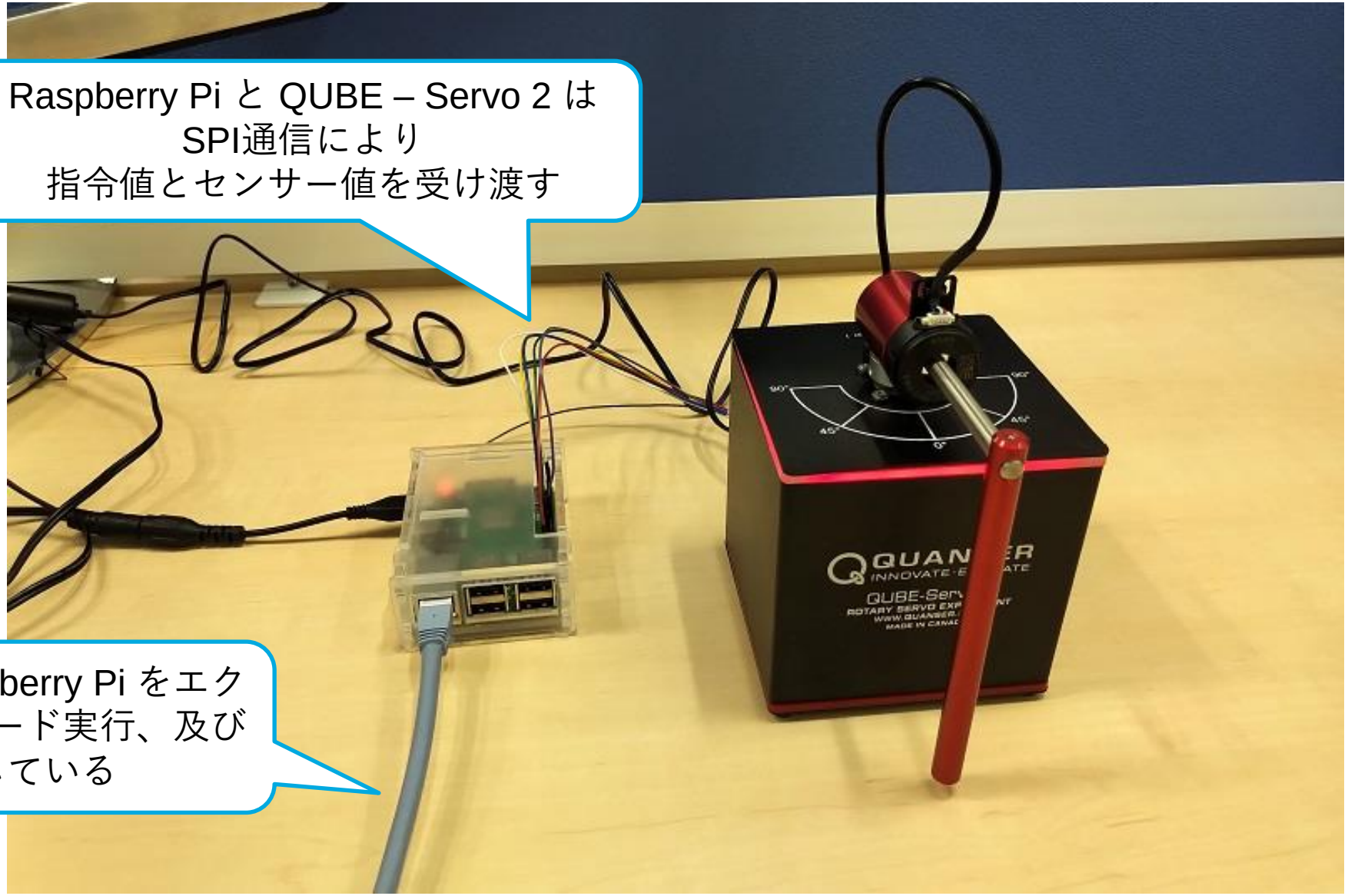
Intel社Xeon,
Xeon Phi

ARM社Mali,
Cortex-A

一般的なCPU

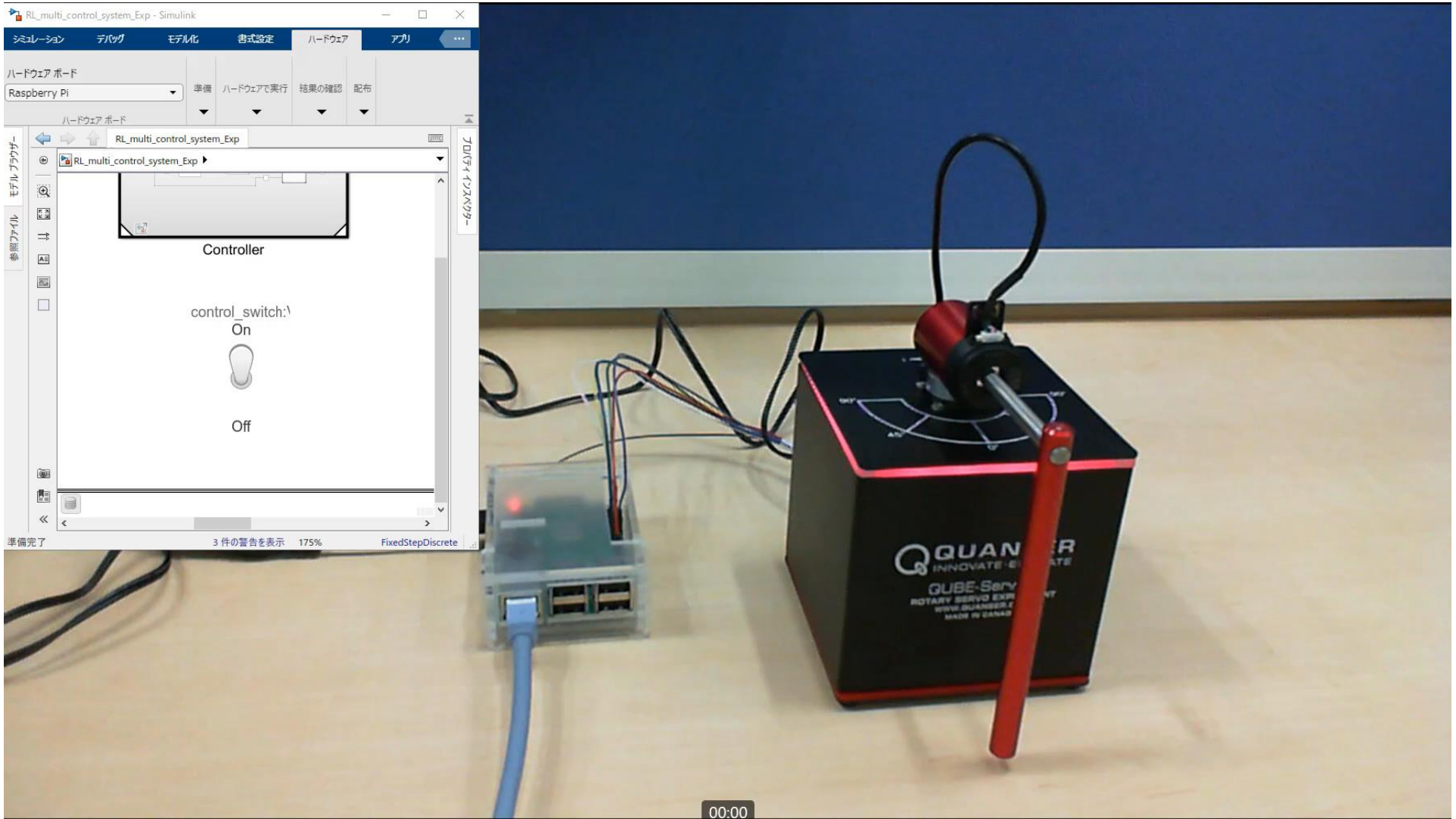
今回の深層ネットワークであれば、
どのタイプのライブラリでもコード
生成することができる

# 実機検証（Rapid Code Prototyping, RCP）



# 実機の動作

https://www.mathworks.com/videos/the-practical-application-of-deep-learning-and-reinforcement-learning-for-control-and-vehicle-design-1631201875485.html



# 制御系ツールの新機能

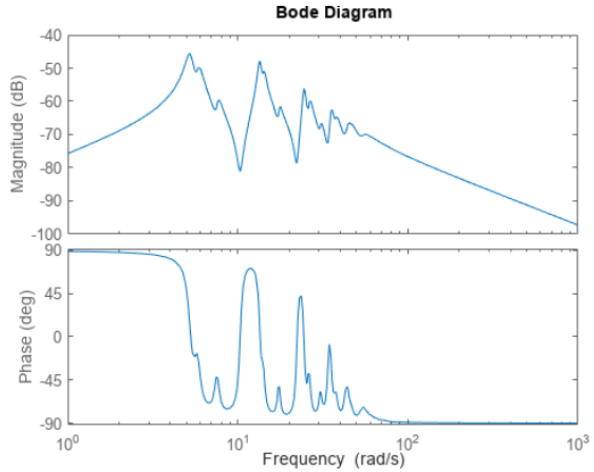
# Control System Toolbox

# モデル低次元化ワークフローの改善

R2023b

**新しい API を使用して、線形時不変（LTI）モデルとスパース LTI モデルの両方でモデル低次元化を実行**

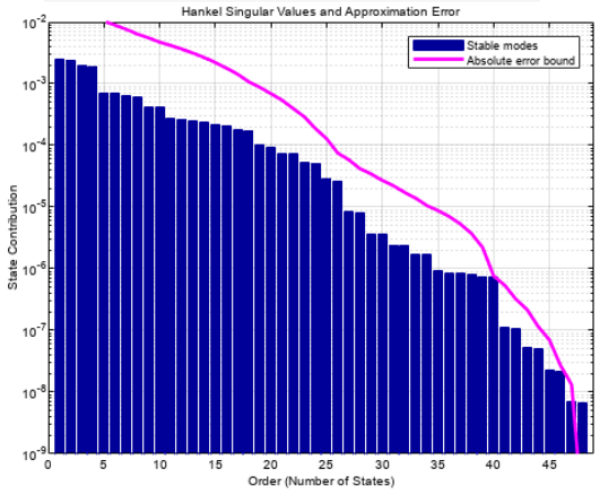
```
load('building.mat','G')
bodeplot(G)
```




- *reducespec* 関数を用いてモデル低次元化（MOR）の仕様に関するオブジェクトを作成
- 平衡化打ち切りやモード打ち切りのようなモデル低次元化アルゴリズムの実行
- 寄与が無視できる状態やモードを除外してモデル次数を選択し、低次元化モデルを取得

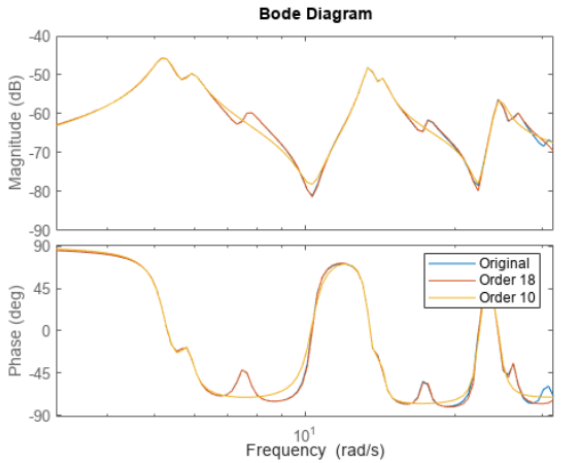
Balanced truncation

```
R = reducespec(G,"balanced");
view(R)
```




```
G18 = getrom(R,Order=18);
G10 = getrom(R,Order=10);

bodeplot(G,G18,G10,logspace(0.5,1.5,100));
legend('Original','Order 18','Order 10');
```

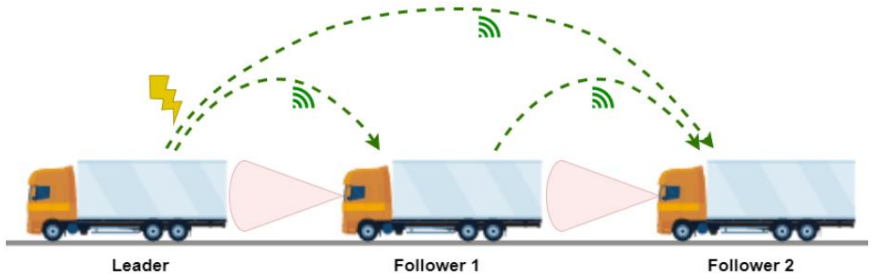


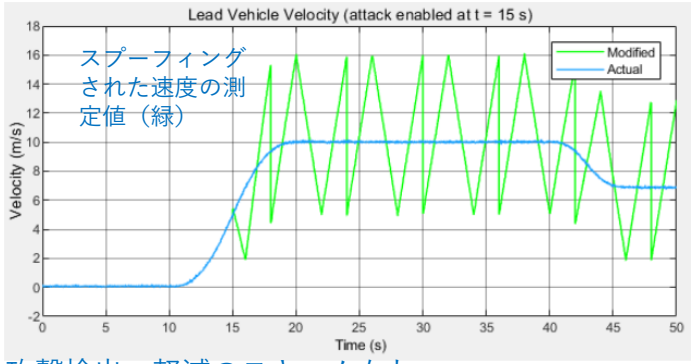
# サイバーフィジカルシステムのセキュリティに関するトラック隊列走行の例題

R2023b

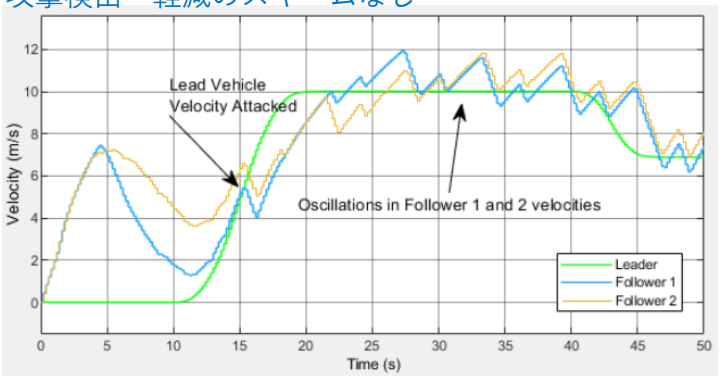
## トラック隊列走行への攻撃の検出と軽減

- カルマンフィルターを使用し、車車間の通信チャネルに対する偽データのインジェクション攻撃を検出
- トラックをアダプティブ・クルーズ・コントロールモードに切り替えることで、攻撃を軽減

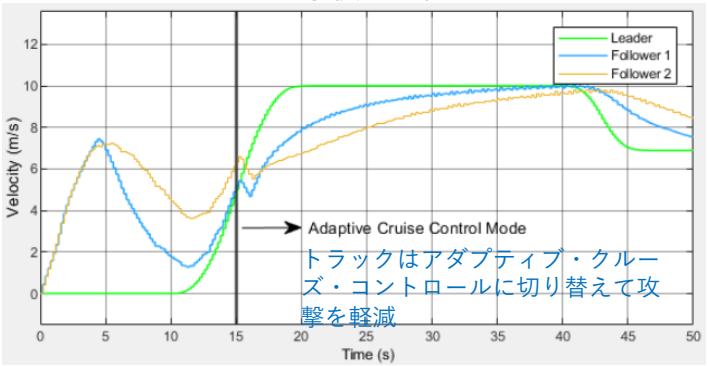



攻撃検出・軽減のスキームなし




モデルベースによる攻撃検出・軽減のスキーム




例題

# Simulink Control Design

# Simulink Control Designにおける実装可能な制御アルゴリズムの発展

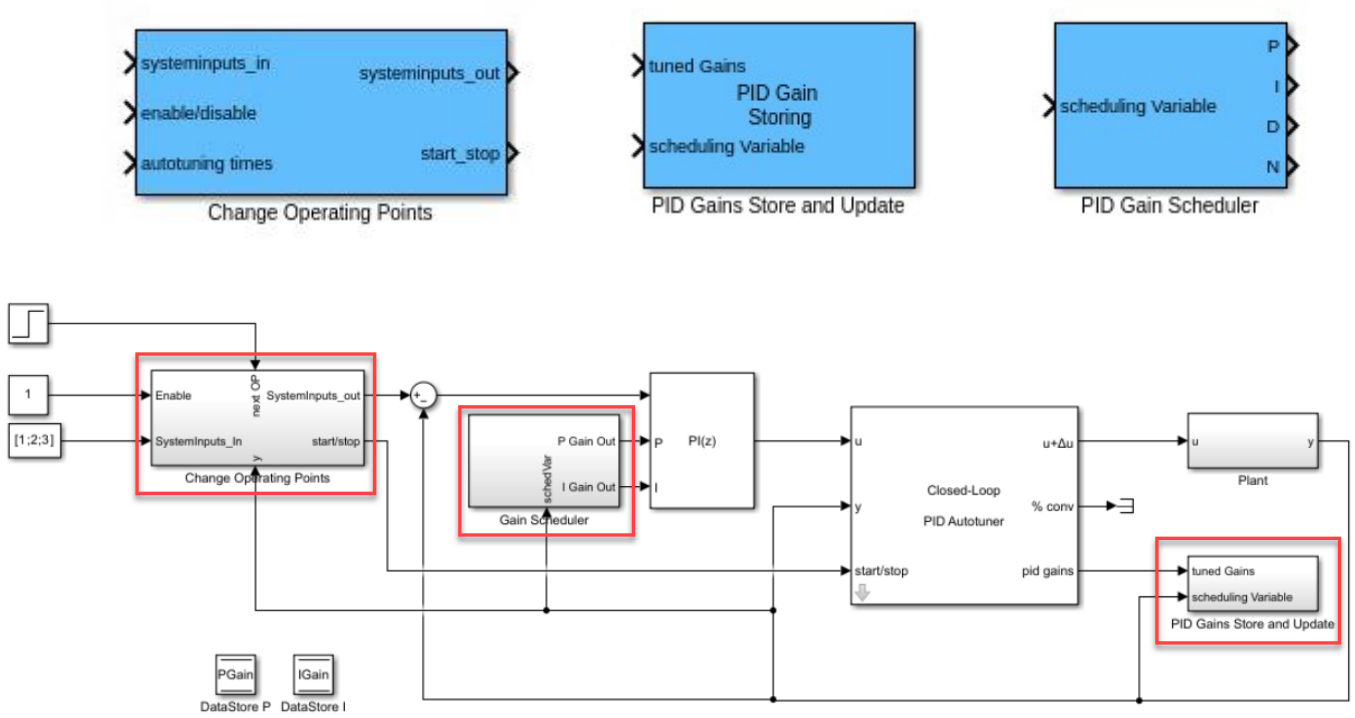
| | <2021 | 2021 | 2022 | 2023 |
|---|---|---|---|---|
| PID 調整 | **Closed-Loop PID Autotuner 改善**<br>• 実行時間低減 (R21b)<br>• ターゲットハードウェアのスループット要求を低減 (R21a) | • Field Oriented Control Autotuner (Motor Control Blocksetが必要) | | |
| AI/データ駆動制御 | | • **モデル規範型適応制御 (MRAC) (R21a)**<br>• **極値探索制御 (ESC) (R21a)**<br>• 離散ESC (R21b) | • **能動的外乱除去制御 (ADRC) (R22b)**<br>• 間接的 MRAC (R22a) | • 一つの隠れ層ニューラルネットワークを用いたMRAC (R23a) |
| 制約適用 | | • **制約適用ブロック (R21a)** | • **バリア制約適用 ブロック (R22a)** | • **受動性適用ブロック (R23a)** |
| 周波数応答推定 | • **Frequency Response Estimator ブロック (R19a)** | | | **Frequency Response Estimator ブロック改善:**<br>• PRBS信号をサポート (R23a)<br>• PRBS パラメーターの自動適用 (R23b) |

# ゲインスケジューリングPID制御

R2023b

**ゲインスケジューリングPID制御器のゲイン調整を簡単化するブロック**

- 自動的に操作点を切り替えるために、「Change Operating Points」ブロックを使用します。
- 「PID Gains Store and Update」ブロックを使用して、調整されたPIDゲインを保存、更新します。
- 「PID Gain Scheduler」ブロックを使用して、現在の運転点に基づいて動的にPIDゲインを調整します。
- コード生成系ツールを使用してこれらのブロックからコードを生成して実装し、ハードウェア上でゲインの自動チューニングできます。

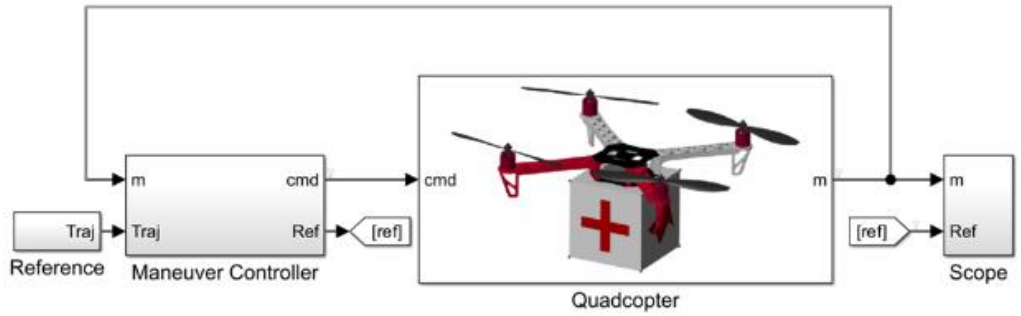


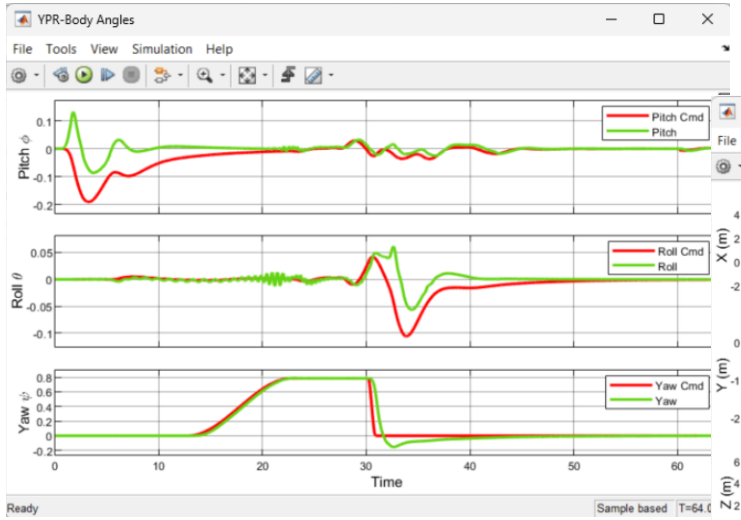
# 新規デモモデル

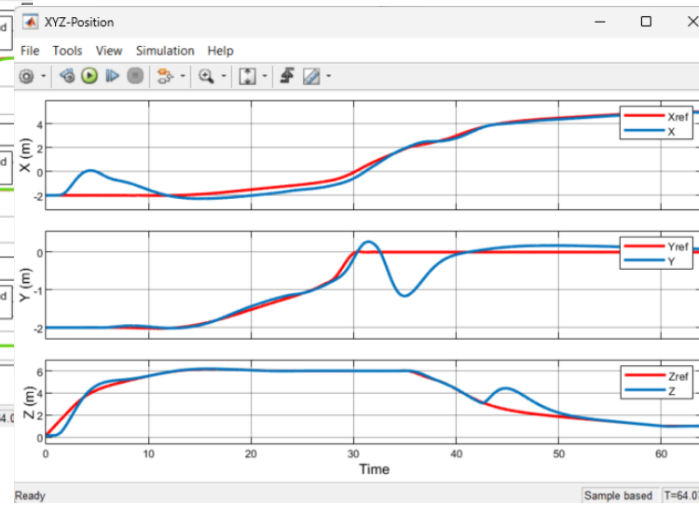
R2023b

## UAV、自動車、化学プロセスの制御器を調整

- Control a quadrotor performing waypoint guidance using model reference adaptive control
- Driver calibration for hybrid electric vehicles using Closed-Loop PID Autotuner block
- Distillate purity control for a MIMO pilot distillation column using active disturbance rejection control

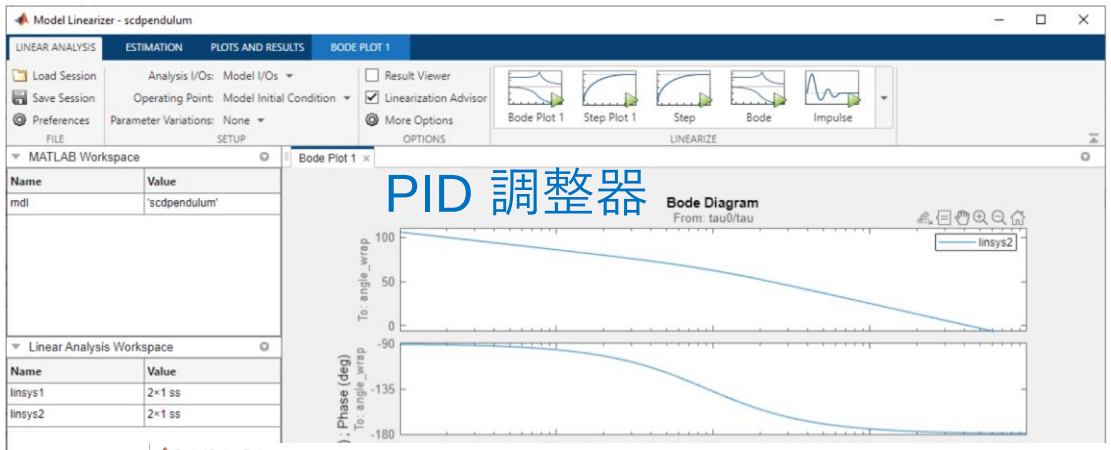


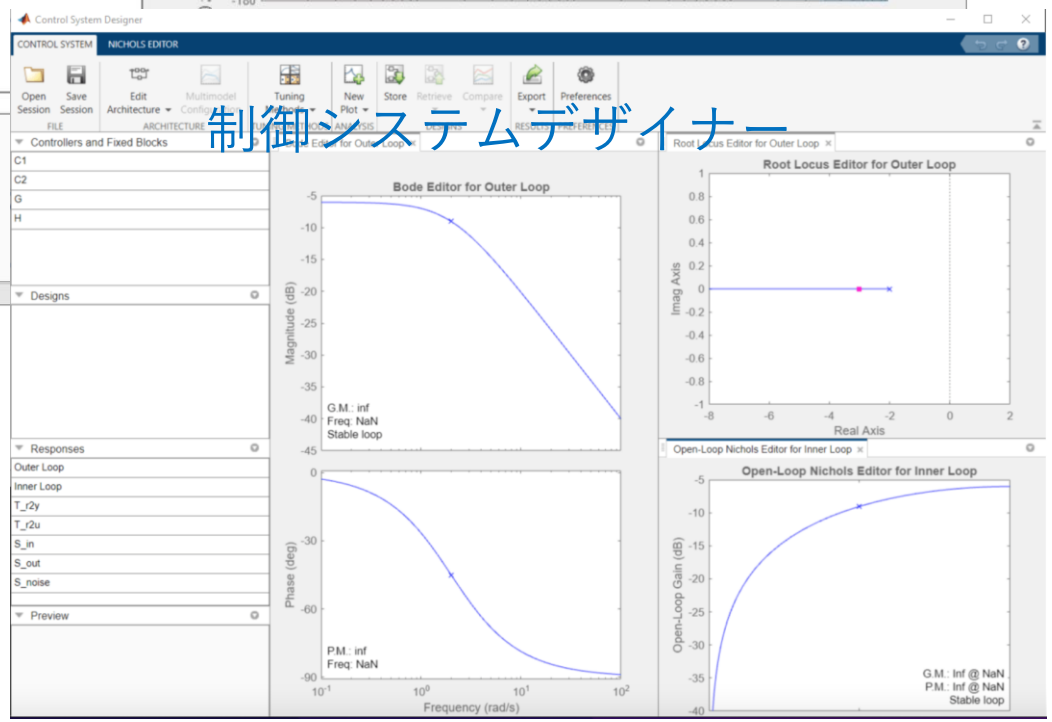



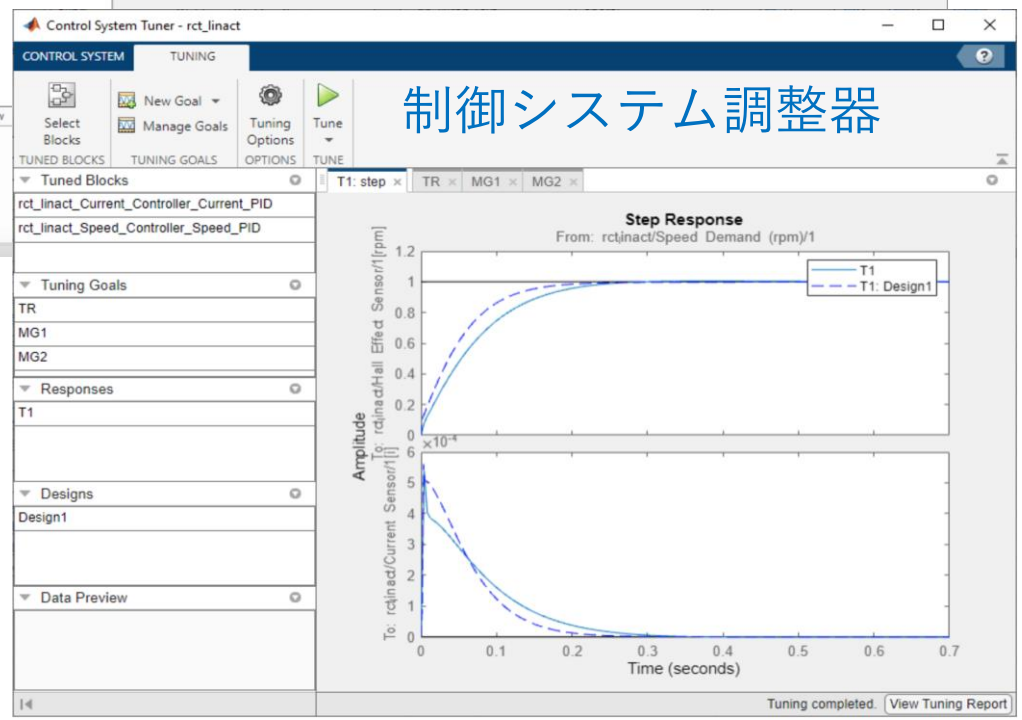
# Simulink Onlineが制御設計アプリをサポート

R2023b



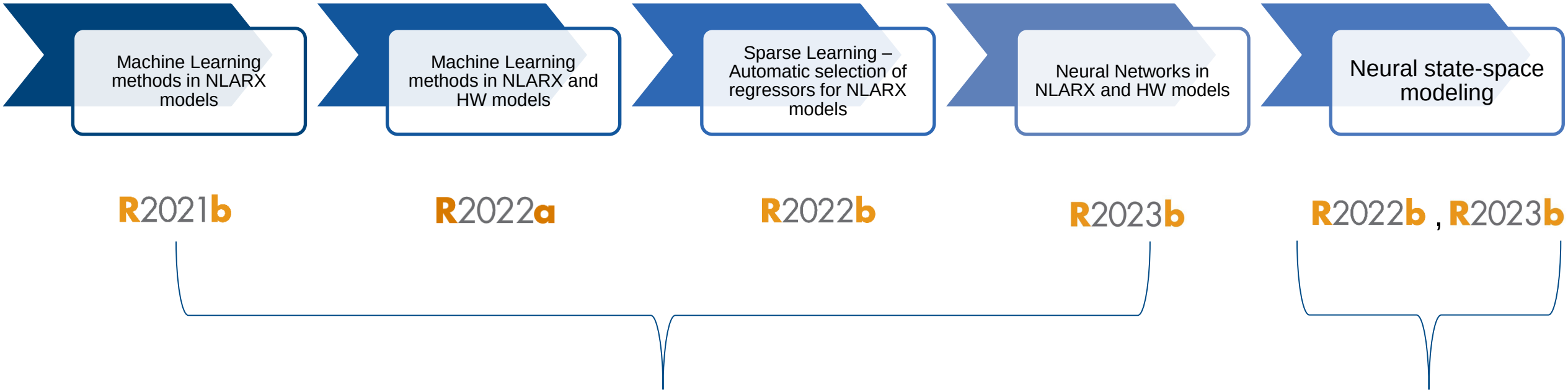




# System Identification Toolbox

# AI と動的システムの融合による非線形システムの同定




*NLARX – Nonlinear ARX*
*HW – Hammerstein-Wiener*

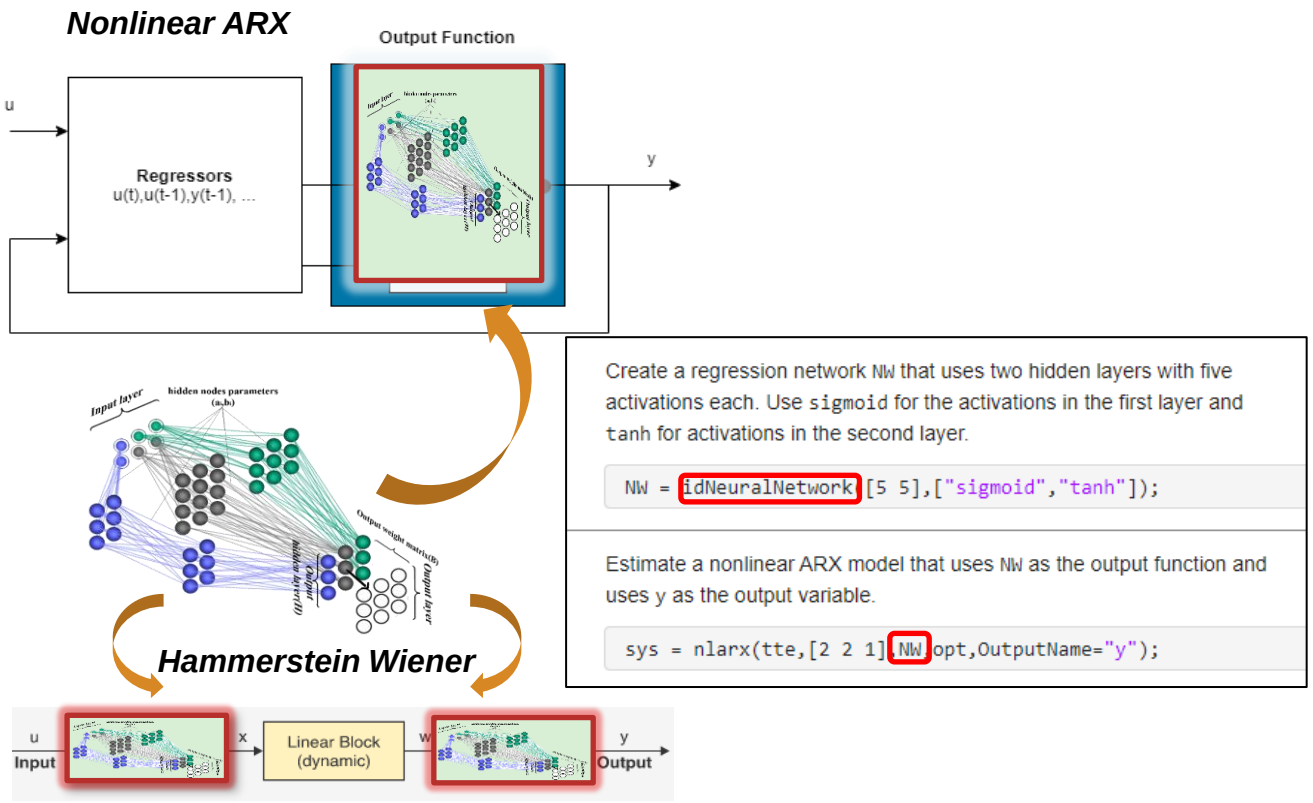
# Neural Networks in Nonlinear ARX and Hammerstein Wiener Models

R2023b

## 非線形システムのダイナミクスをより適切に捉えるモデルを同定する

- Statistics and Machine Learning Toolbox 及び Deep Learning Toolboxから提供されている浅いNN及び深層レイヤーネットワークを用いてNonlinear ARX 及び Hammerstein Wiener 、モデルを構築する

- ネットワークを物理ベースの線形コンポーネントと組み合わせる



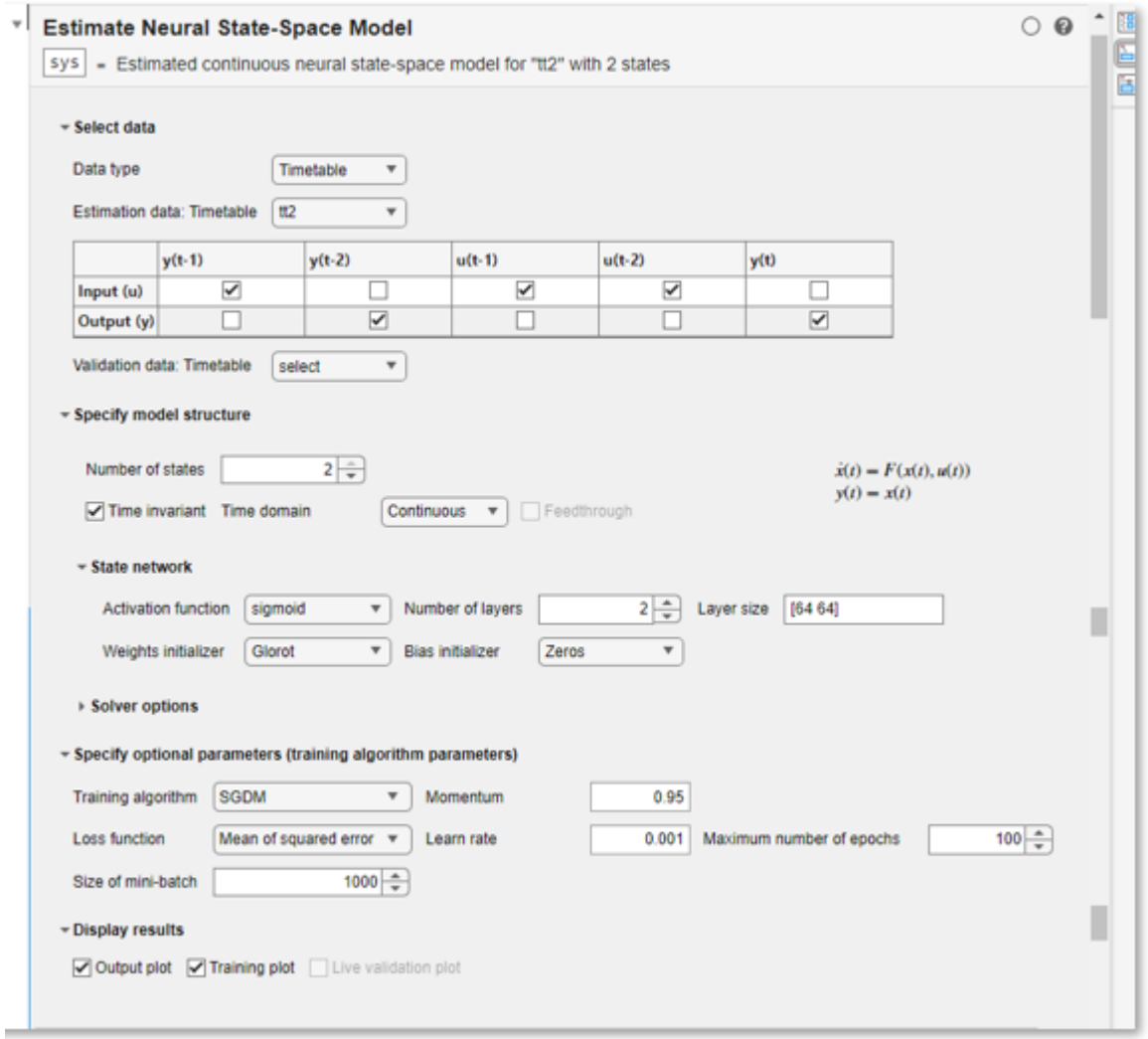

**Neural networks for nonlinear identification**

# Neural State Space Live Task

R2023b

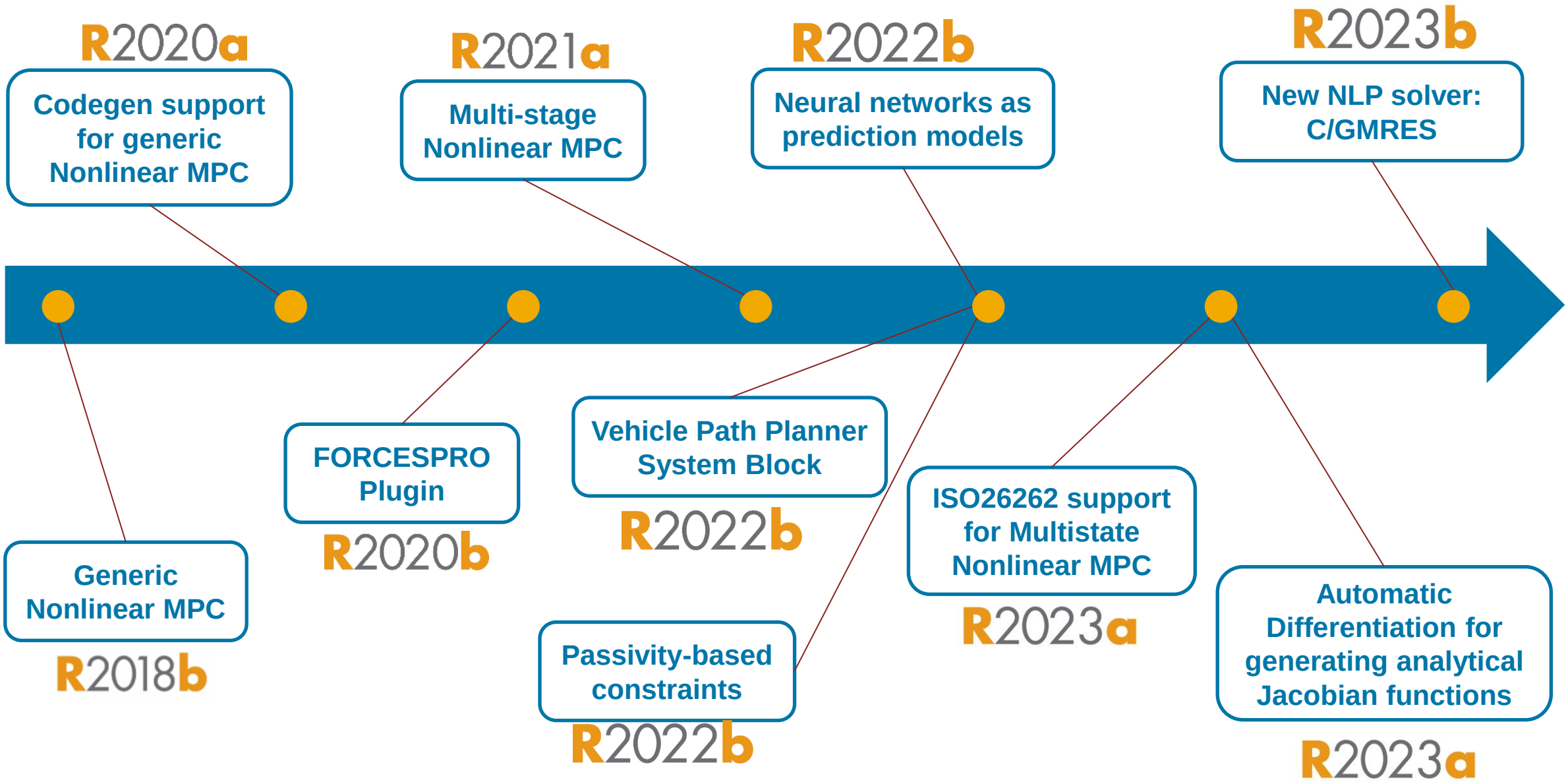
**インタラクティブにNeural state spaceモデルを設計可能に**

- **Estimate Neural State-Space Model** ライブタスクを用いることで入出力データの選定、モデル構造の選定、ハイパーパラメーターの調整、そして結果のプロットといったタスクをエディタ上で完結可能に
- ライブタスクからMATLABコードの生成もサポート



# Model Predictive Control Toolbox

# 非線形MPCが主な注力領域に



# 非線形MPCの解法

**非線形MPCの問題**

非線形な予測モデル

$$\begin{cases} \dot{x} = f(x,u) \\ y = g(x,u) \end{cases}$$

任意な非線形コスト関数

$$J = cost(x,u)$$

任意な非線形制約

$$c = ineq(x,u)\ ceq = eq(x,u)$$

非線形計画問題
(直接法)

Solved by fmincon

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \le 0 \text{ for each } i \in \{1,\dots,m\} \\ & h_j(x) = 0 \text{ for each } j \in \{1,\dots,p\} \\ & x \in X. \end{aligned}$$

2点境界値問題
(間接法)

Solved by C/GMRES

$$\begin{cases} \dot{x} = \boldsymbol{f}(x(t),u(t),t), & x(t_0) = x_0 \\ \dot{\lambda}^T = -\dfrac{\partial \boldsymbol{H}}{\partial x}, & \lambda(t_f)^T = -\dfrac{\partial \boldsymbol{\varphi}}{\partial x}\Big|_{t=t_f} \\ \dfrac{\partial \boldsymbol{H}}{\partial u} = 0 \end{cases}$$

# 新しいNLPソルバー: C/GMRES

R2023b

**Multistage Nonlinear MPCにて実行効率に優れたC/GMRESソルバーを利用可能に**

- C/GMRESは連続法および一般化最小残差法に基づいた手法
- 計算効率が高いため、fminconと比較してリアルタイム性に優れる
- コード生成をサポート
- ハード制約はソフト制約に変換
- 現状はユーザーカスタムの等式、不等式制約は考慮できない

```
% Create a multistage nonlinear MPC controller with 6 states, 2 inputs and
% prediction horizon = 50.  By default, it uses the "fmincon" solver.
planner = nlmpcMultistage(50,6,2);
% Configure it to use the c/gmres solver and choose solver settings
planner.Optimization.Solver = 'cgmres';
planner.Optimization.SolverOptions.StabilizationParamater = 1/planner.Ts;
planner.Optimization.SolverOptions.Restart = 30;
planner.Optimization.SolverOptions.MaxIterations = 200;
planner.Optimization.SolverOptions.BarrierParamater = 1e-5;
% Compute optimal rocket landing trajectory
[mv, ~, info] = nlmpcmove(planner, x0, u0);
```

Automatica
Volume 40, Issue 4, April 2004, Pages 563-574

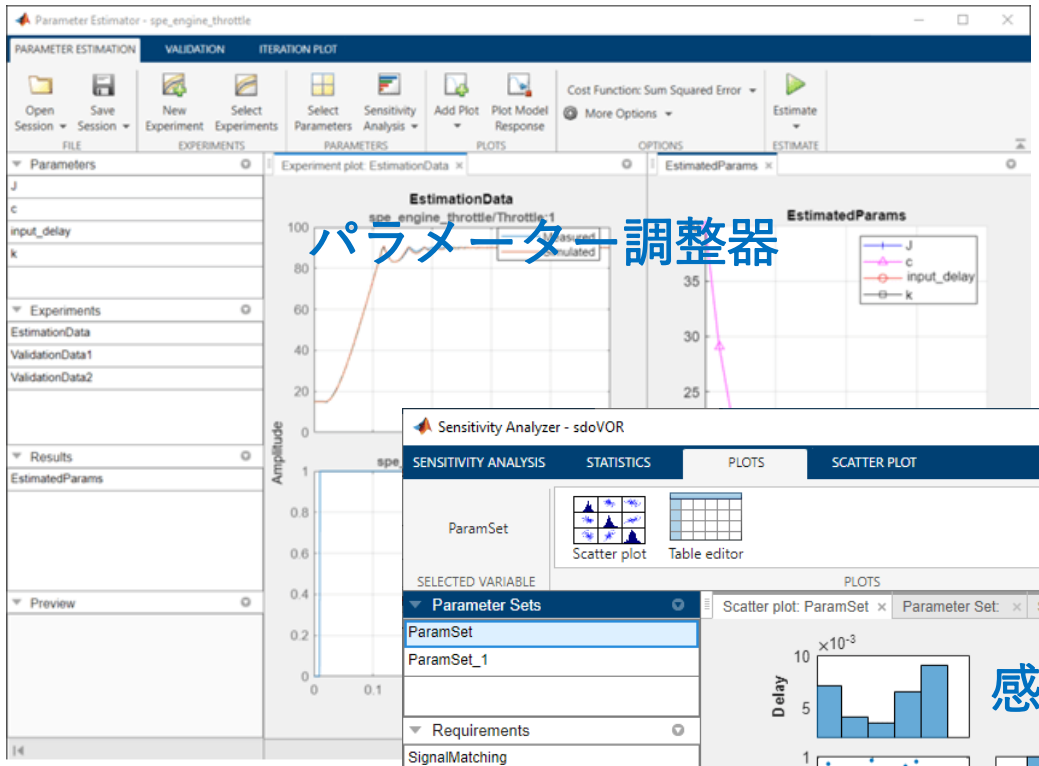
## A continuation/GMRES method for fast computation of nonlinear receding horizon control ☆

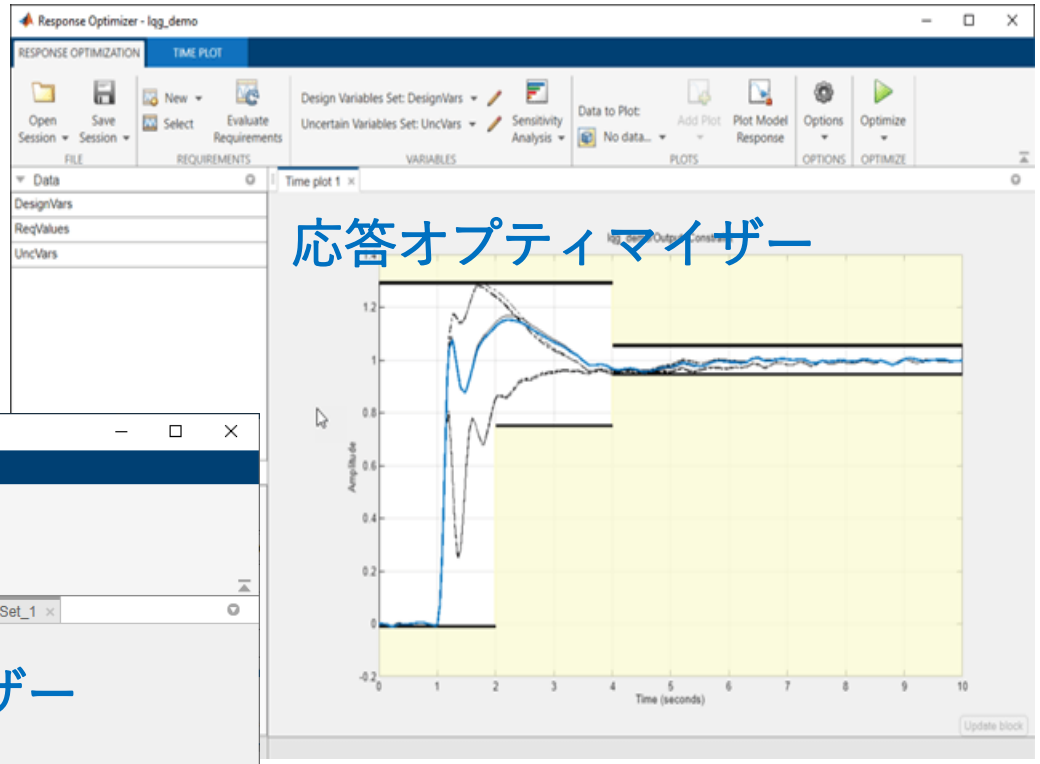
Toshiyuki Ohtsuka

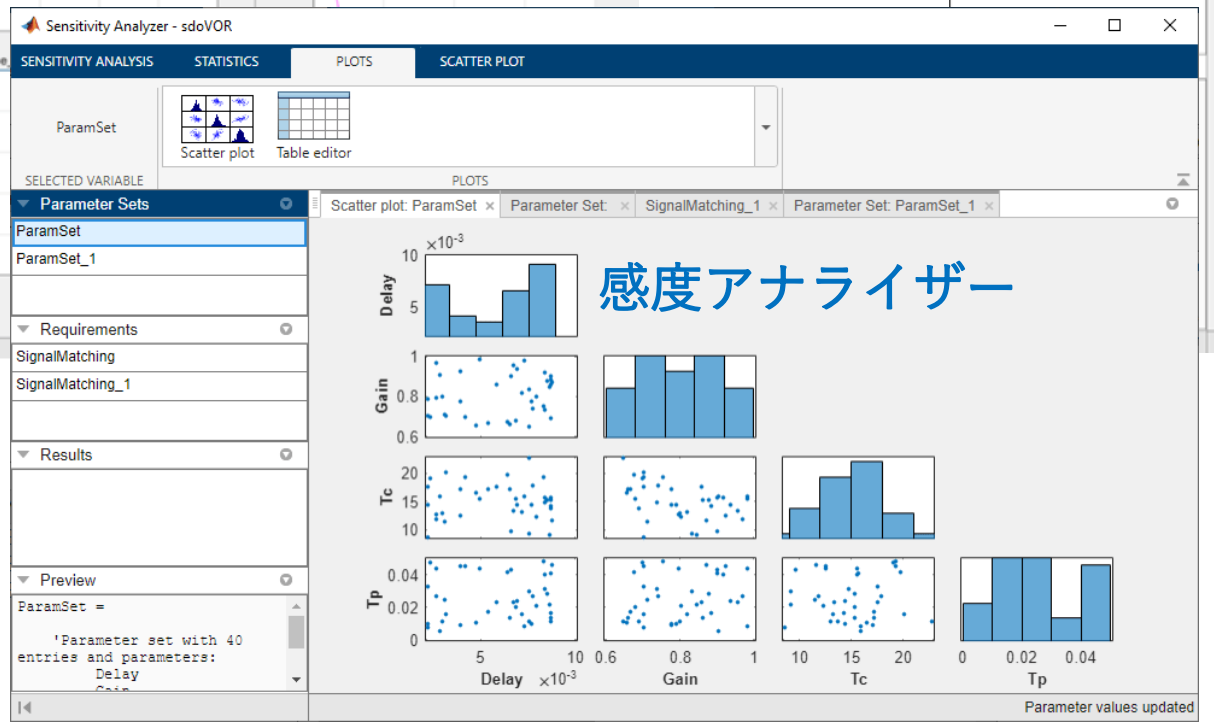
# Simulink Design Optimization

# Simulink OnlineがSimulink Design Optimizationのアプリをサポート

R2023b



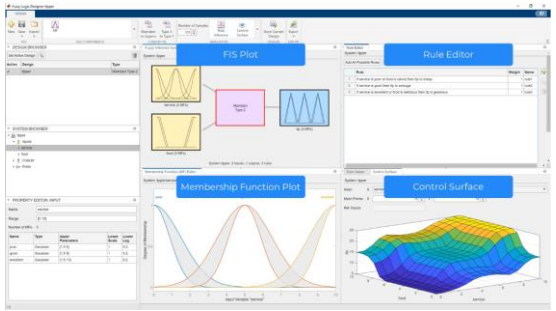



# Fuzzy Logic Toolbox

# Fuzzy Logic Designer アプリ

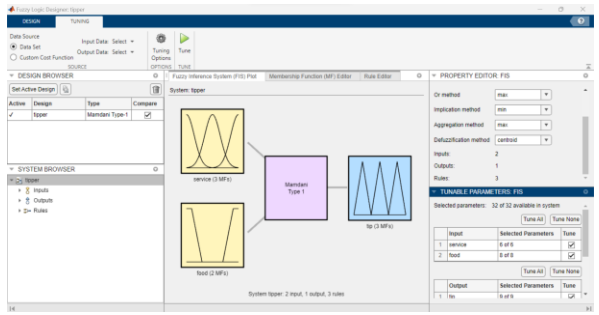
Fuzzy Logic Designer アプリ
の刷新

R2022b



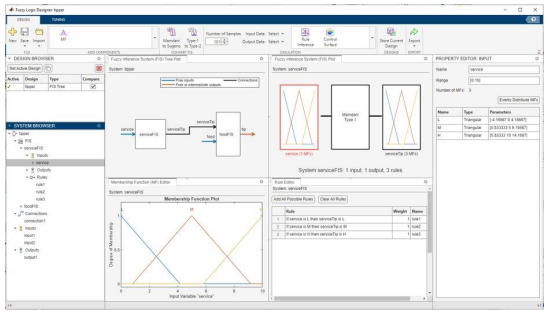

アプリでファジィチューニングをサポート

R2023a



アプリで FIS ツリーをサポート

R2023b

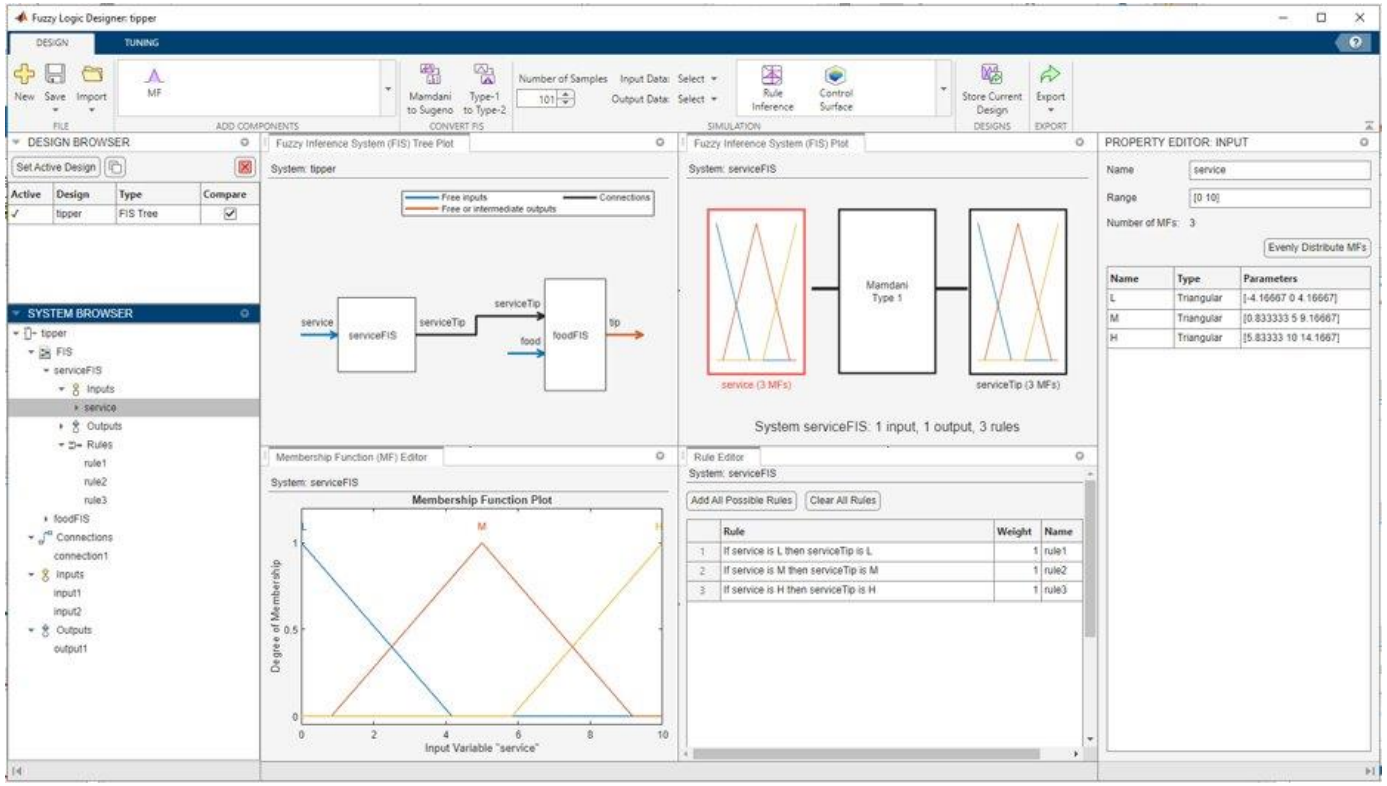


# Fuzzy Logic Designer アプリで FIS ツリーをサポート

R2023b

## FIS ツリーをグラフィカルに設計、シミュレーション、およびチューニング

- FIS ツリーを用いて、相互接続されたファジィシステムの集合をインタラクティブに構築・調整
- FIS ツリーシステムのメンバーシップ関数とルールを分析・評価・改良
- 設計した FIS ツリーシステムを MATLAB ワークスペースにエクスポートし、Simulink モデルや組込み展開に利用




Accelerating the pace of engineering and science