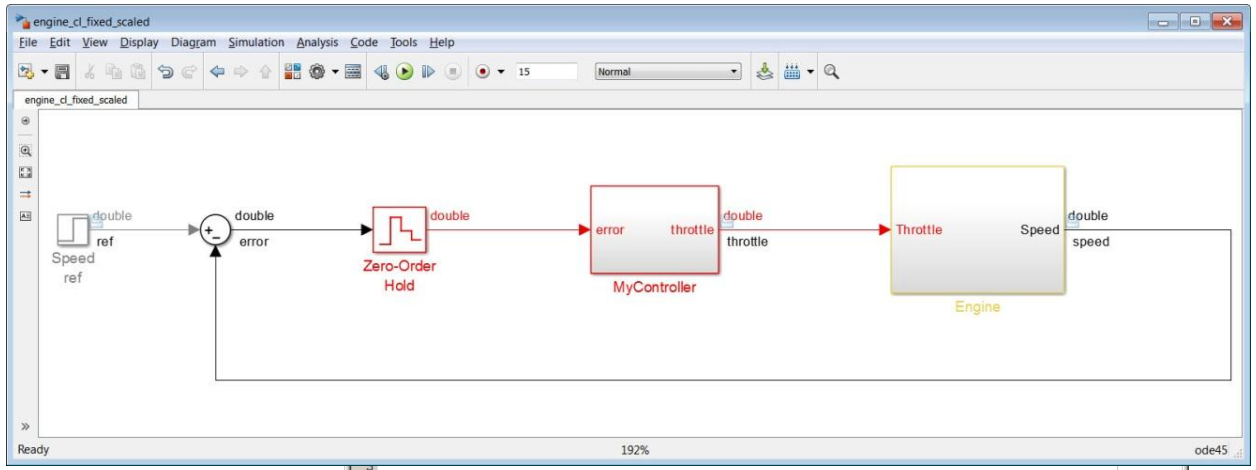
PID 制御を Simulink でより簡単に

MathWorks Japan

デモ: エンジン制御



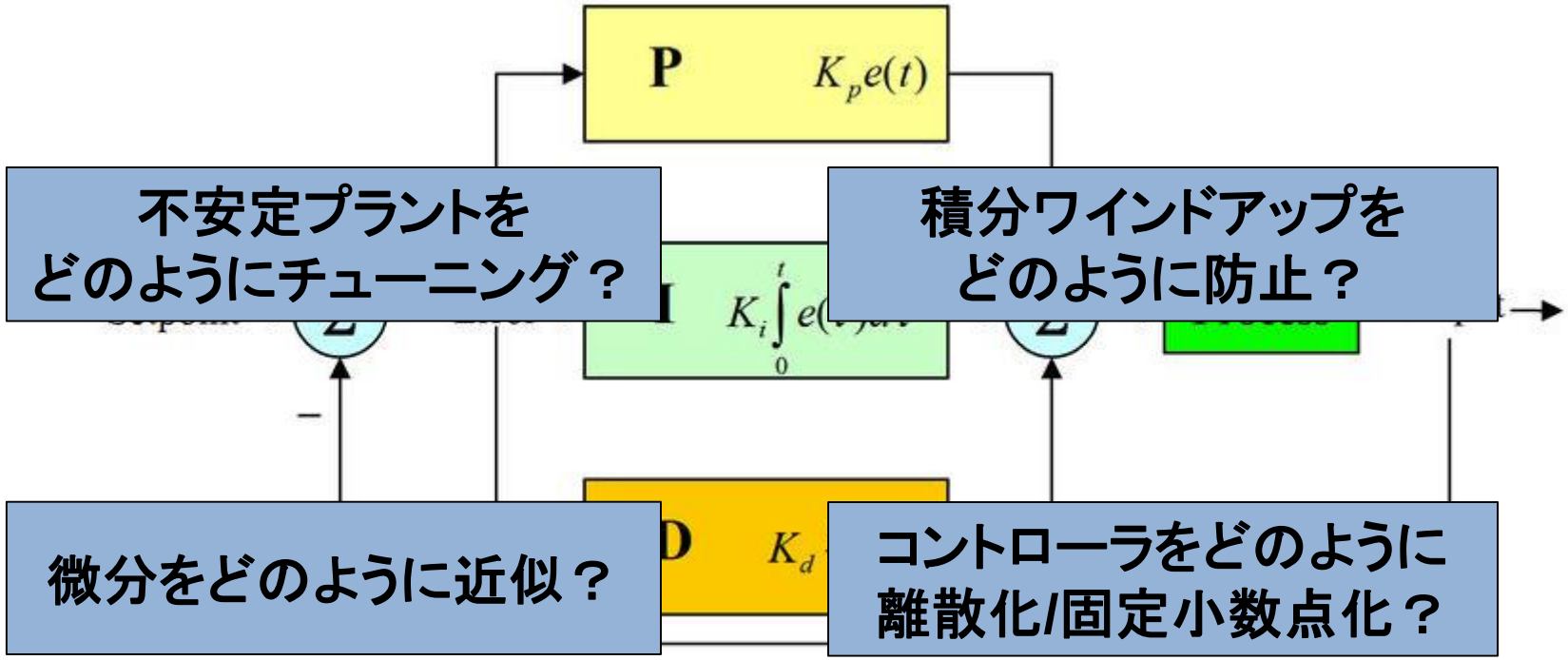

```
int16_T error;
int16_T throttle;
int16_T Kd = 29577;
int16_T Ki = 27387;
int16_T Kp = 25689;
D_Work DWork;
void Fixed_step(void)
{
  int16_T FilterCoefficient;
  FilterCoefficient = (int16_T)((int32_T)((int16_T)((int32_T)Kd * (int32_T)error
    >> 15) - DWork.Filter_DSTATE) * 2637L >> 12);
  throttle = (((int16_T)((int32_T)Kp * (int32_T)error >> 15) >> 2) +
              (DWork.Integrator_DSTATE >> 1)) + (FilterCoefficient >> 2);
  DWork.Integrator_DSTATE = (int16_T)((int32_T)(int16_T)((int32_T)Ki * (int32_T)
    error >> 15) * 5243L >> 19) + DWork.Integrator_DSTATE;
  DWork.Filter_DSTATE = (int16_T)(5243L * (int32_T)FilterCoefficient >> 15) +
    DWork.Filter_DSTATE;
}

void Fixed_initialize(void)
{
  throttle = 0;
  (void) memset((void *)&DWork, 0,
                sizeof(D_Work));
  error = 0;
  DWork.Integrator_DSTATE = 18432;
}
```

C / CPP source or header file Ln 1 Col 1 OVR

PID制御とは?

比例　　積分　　微分

Proportional - Integral - Derivative



不安定プラントを
どのようにチューニング？

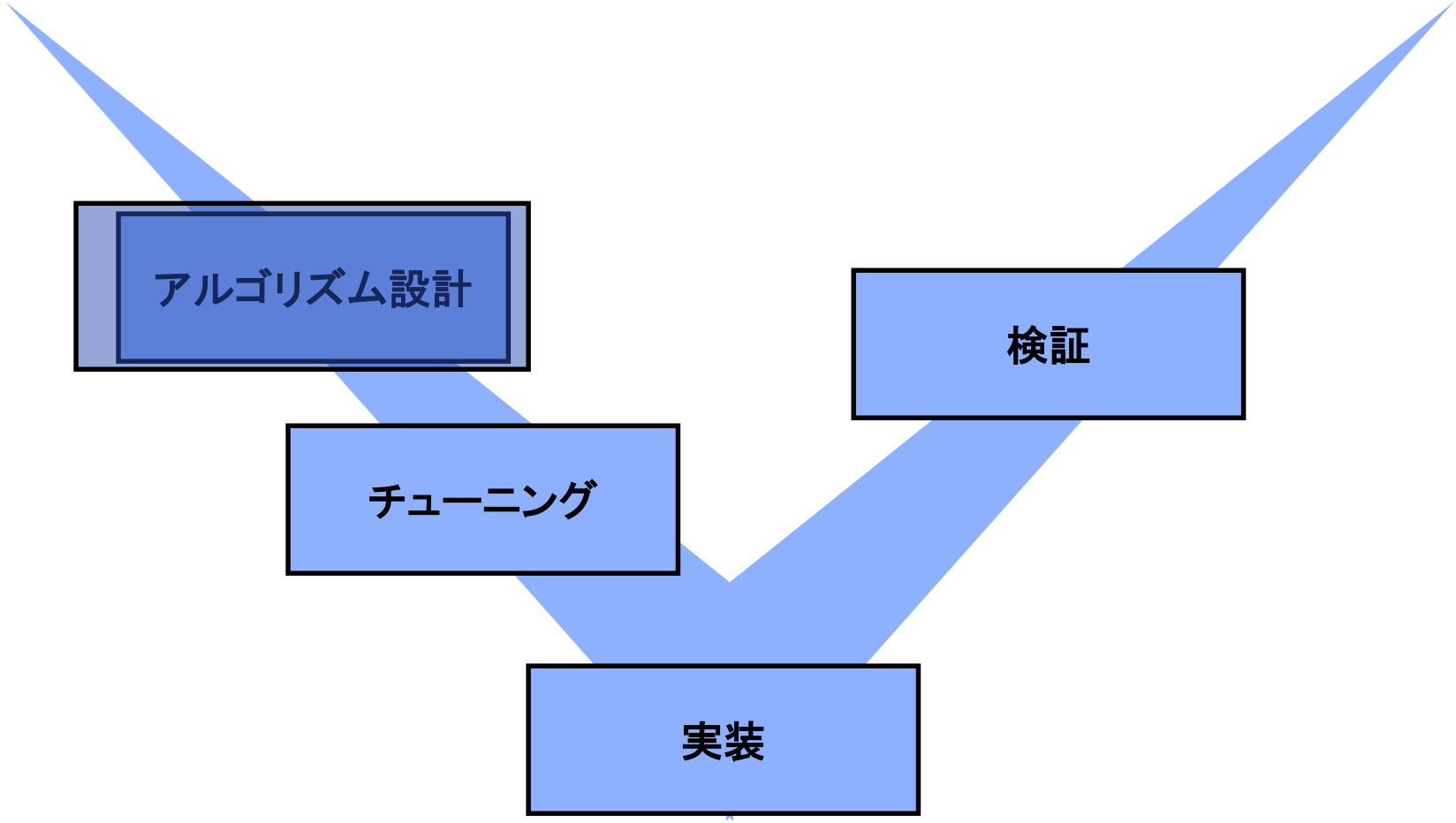
積分ワインドアップを
どのように防止？

微分をどのように近似？

コントローラをどのように
離散化/固定小数点化？

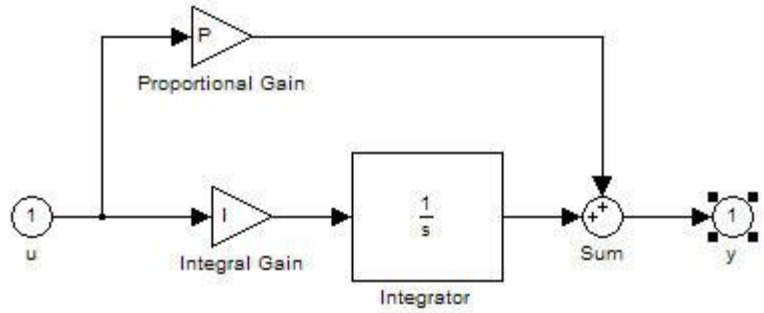
Source: Wikipedia

PIDコントローラ設計のワークフロー

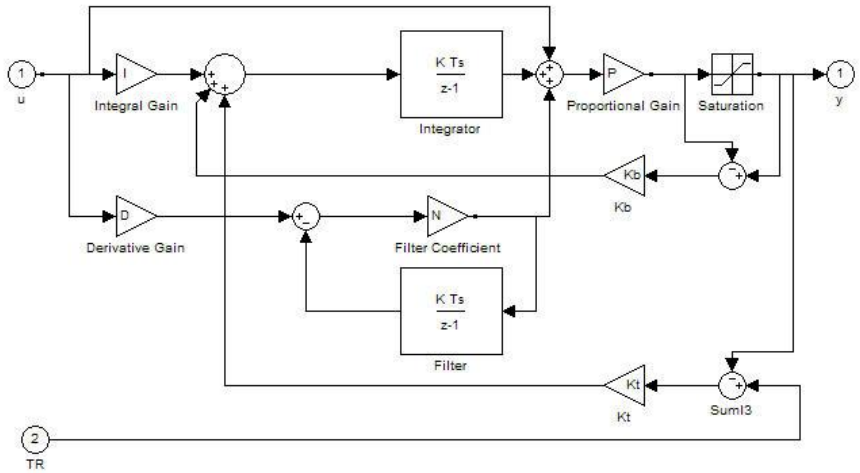


アルゴリズム設計の課題

- コントローラタイプの選択 ― P, PI, PD, PID
- コントローラ形式の選択 ― 標準 or 並列
- 出力飽和制限
- アンチ・ワインドアップ
  - ワインドアップ現象を防ぐ対策が必要
- バンプレス移行（切替）
  - 複数のコントローラ間の切替時に
    操作量が急変しない対策が必要

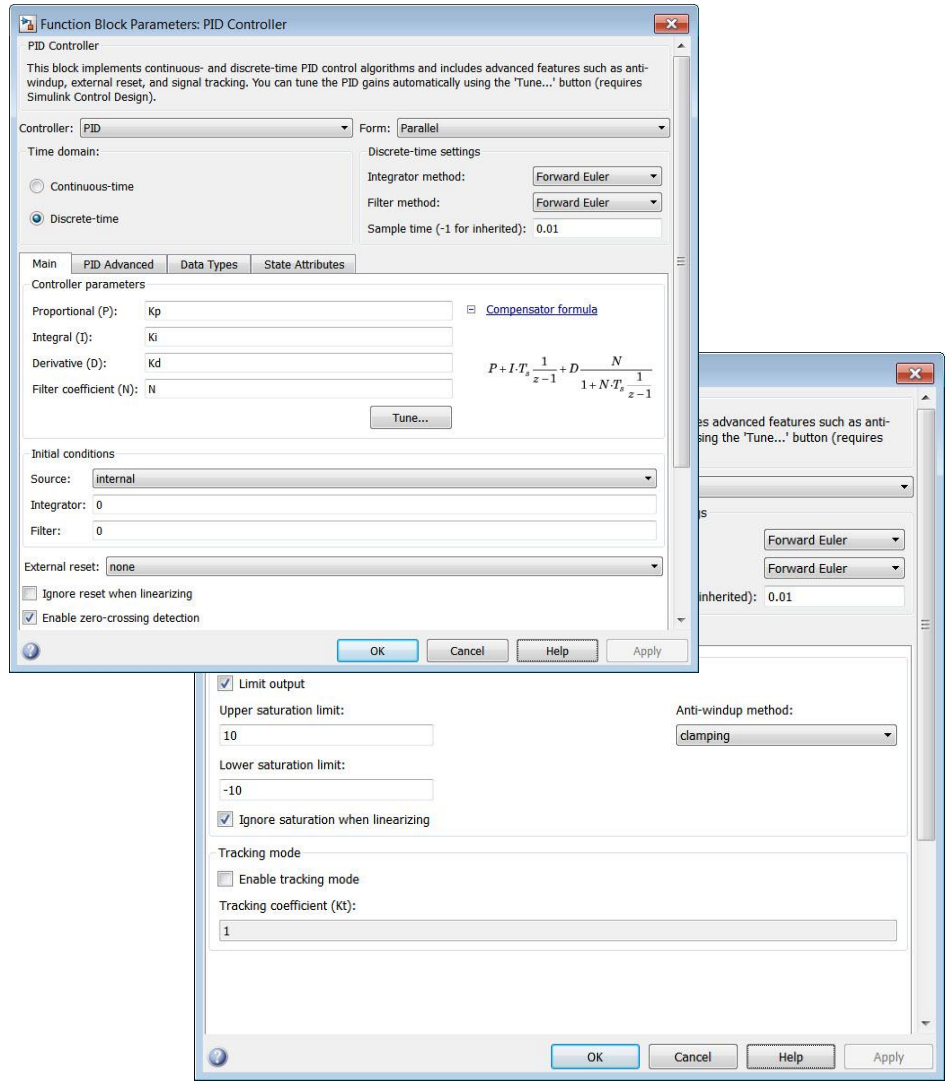


コントローラ設計の
変更は手間がかかる・・・

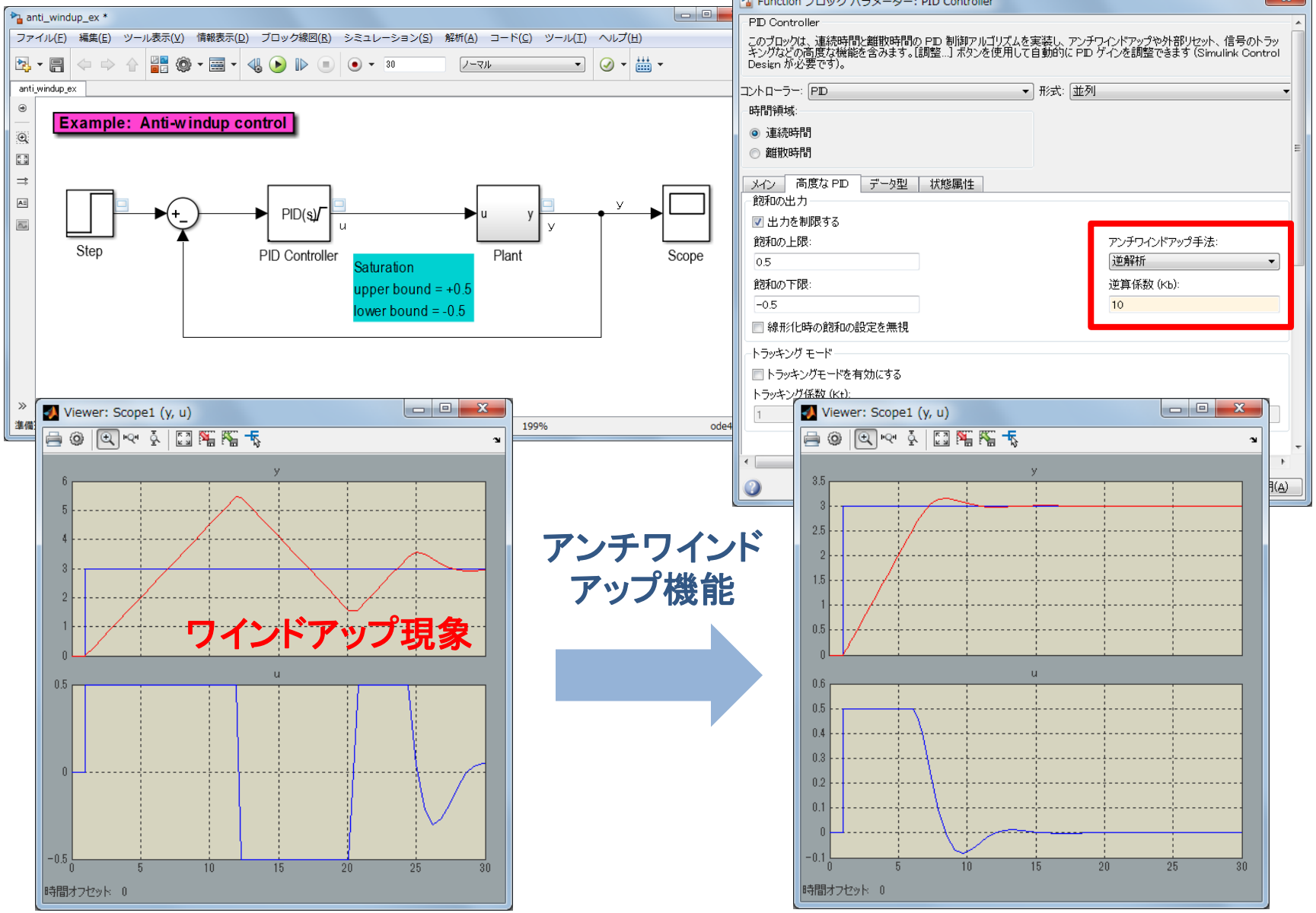


ソリューション： Simulink PID Controller ブロック

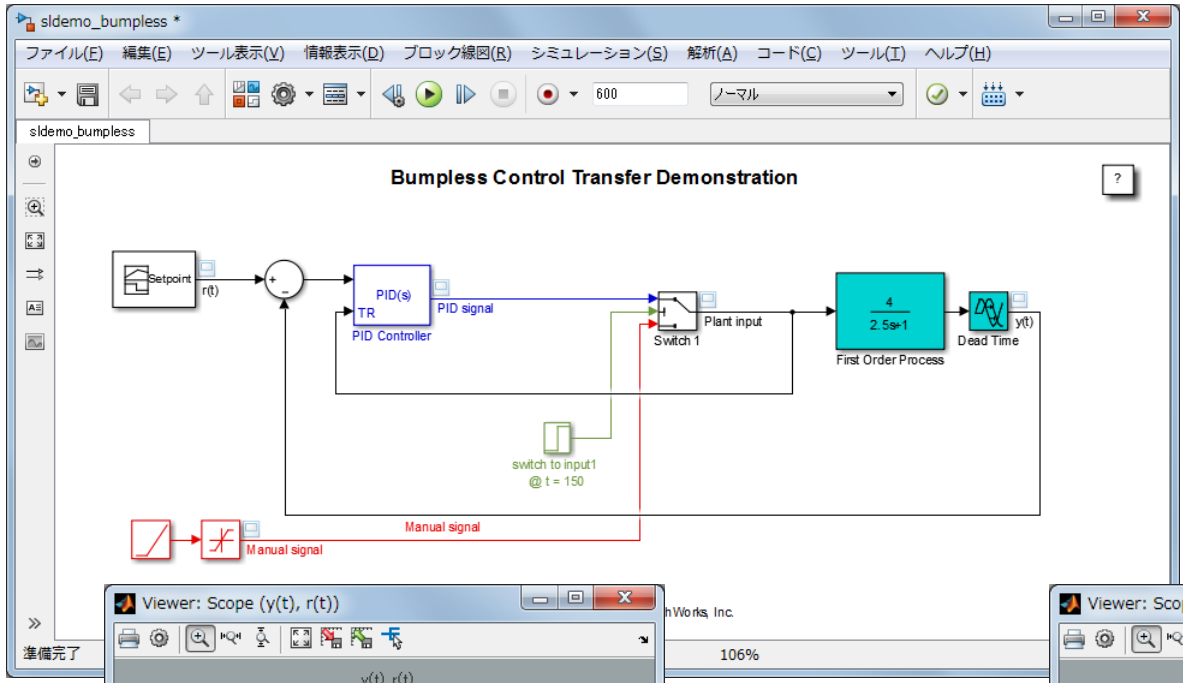
- I, P, PD, PI, PIDを選択
  （Dは近似微分）
- 並列 or 標準形式を選択
- 出力飽和制限の設定
- アンチ・ワインドアップ機能
- トラッキングモードによるバンプレス移行

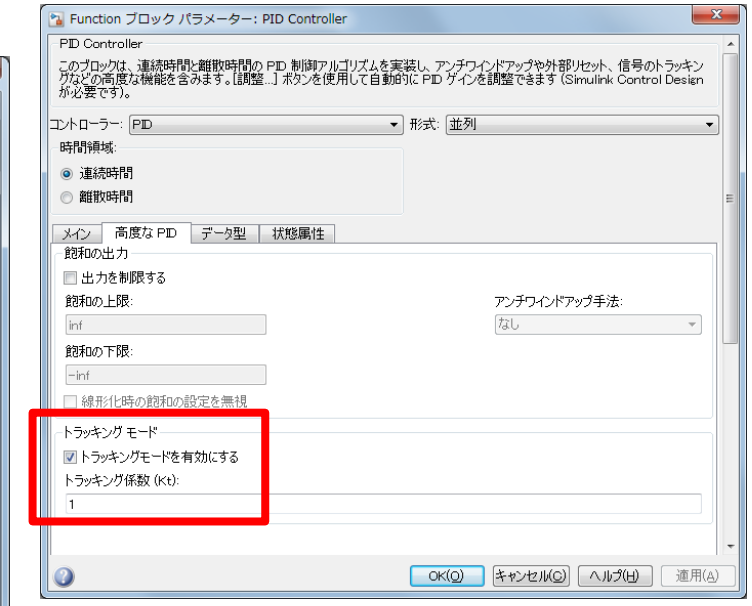


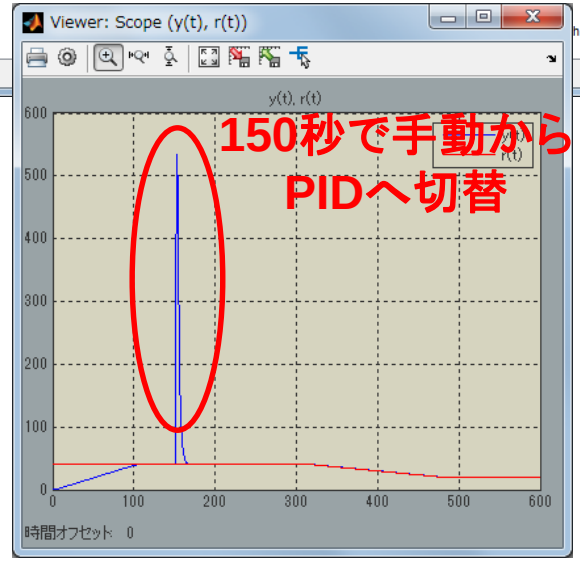
アンチワインドアップ制御



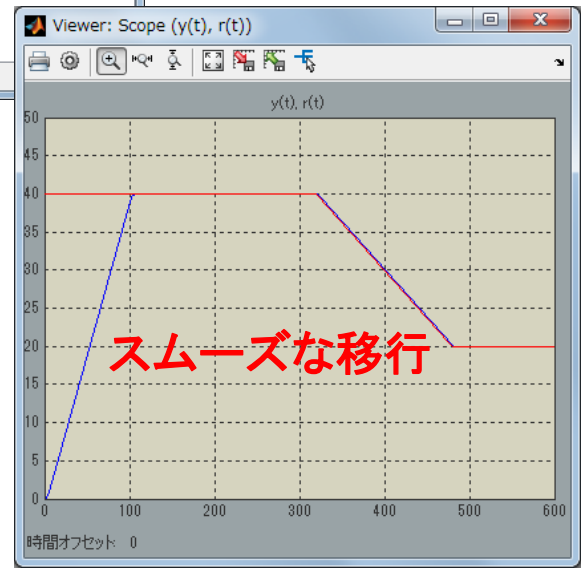
バンプレス移行（切替）





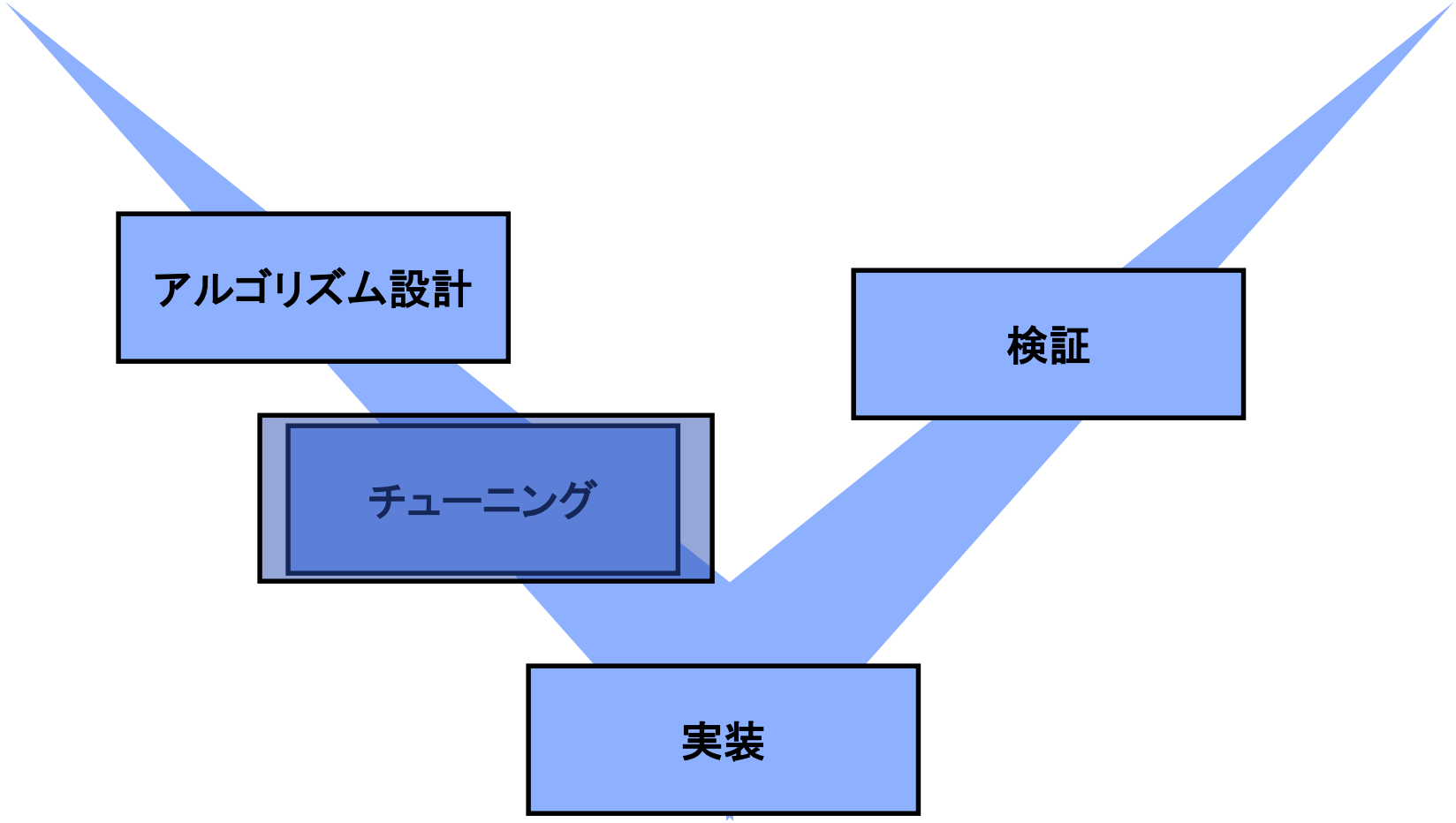


トラッキング
モード



>> sldemo_bumpless

PIDコントローラ設計のワークフロー



チューニングの課題

- チューニングとは、応答時間やオーバーシュート（位相余裕）などの仕様を満たすゲインを探すこと

マニュアル チューニング

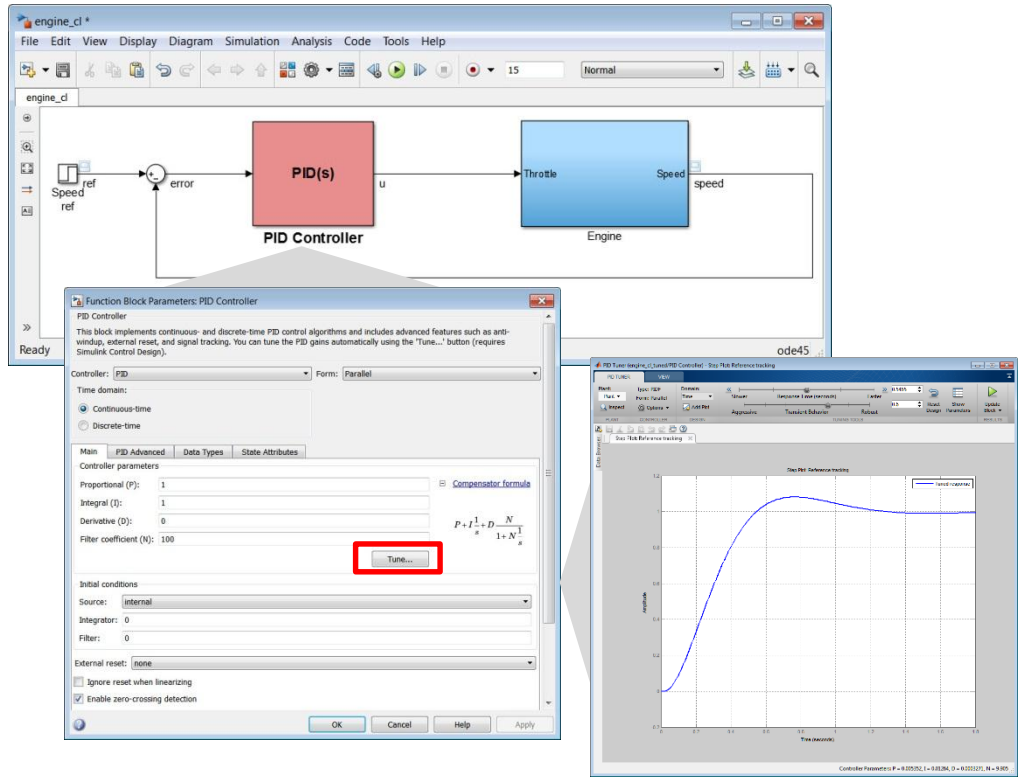
- 試行錯誤のため多くの時間を要する/システマティックではない
- 最適な設計が難しい
- 実機ベースの試行錯誤は、危険な状況を引き起こす可能性も

ルールベース チューニング

- ステップ応答法や限界感度法
- 適用できないケースもあり（不安定系）
- 微調整が難しい

ソリューション： PID 調整器

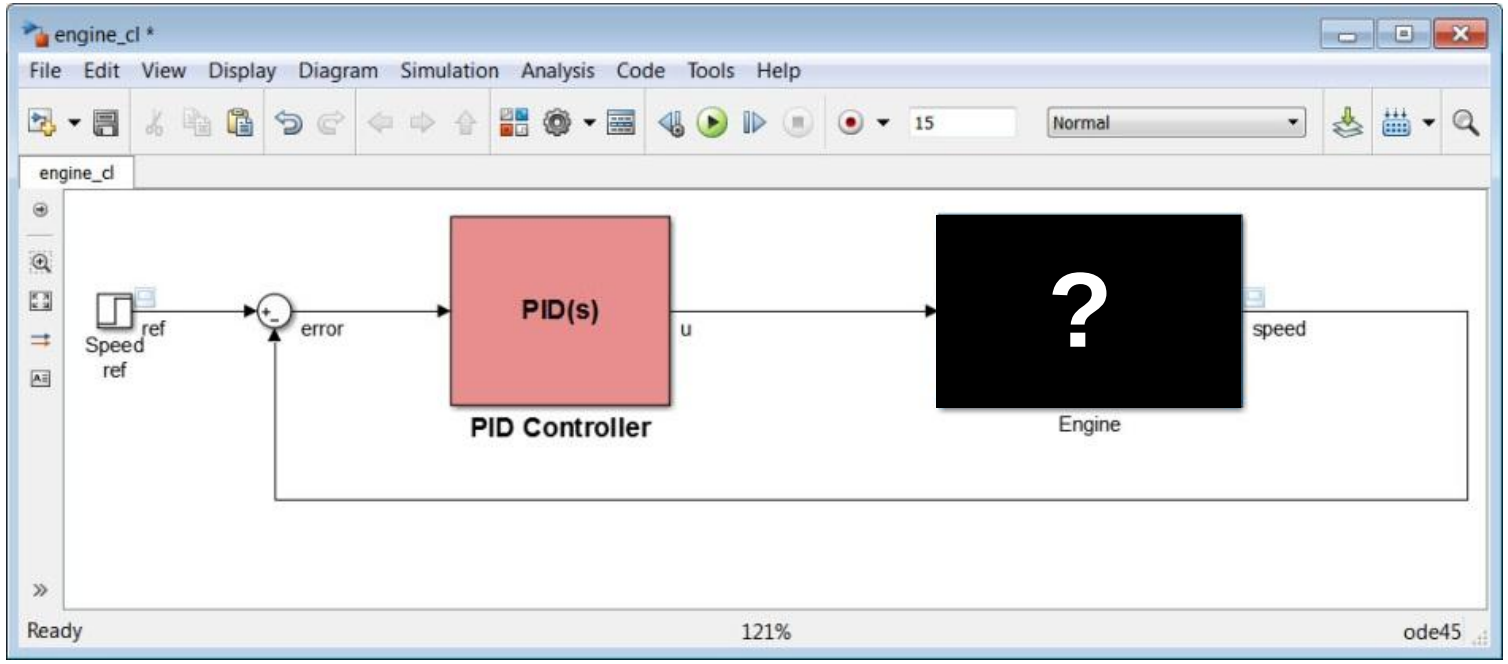
- PIDゲインを自動的に探索
- スライダーによる簡単な微調整
- すべてのタイプのプラントをサポート



Simulink Control Design が必要

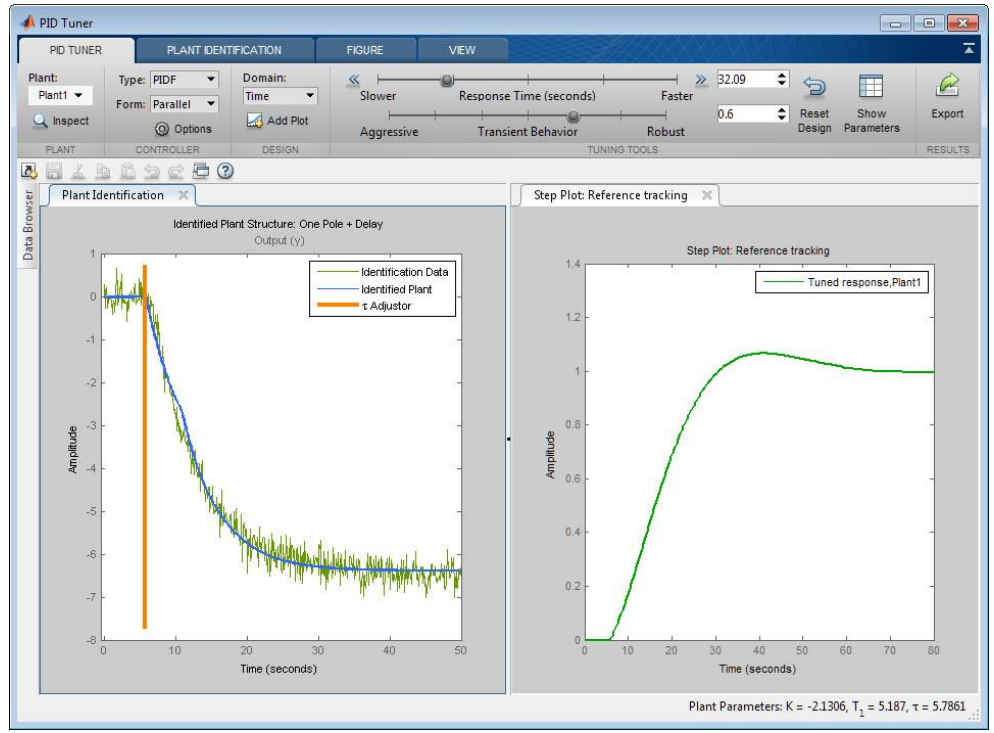
チューニングの大きな課題

- もしプラントモデルがない場合はどうなるか？



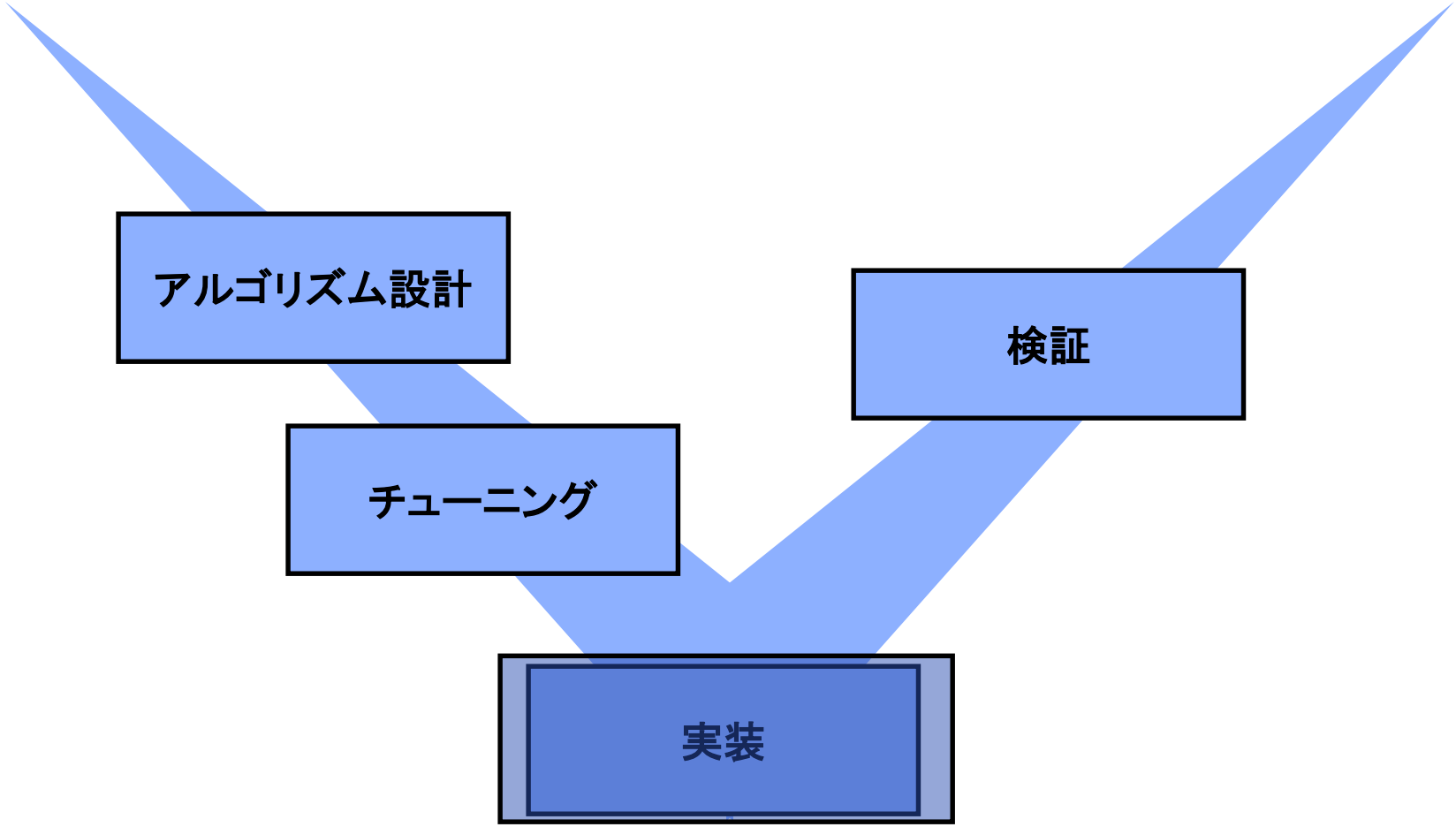
ソリューション： PID 調整器へのシステム同定の統合

- 入出力の測定データを PID 調整器に直接インポート
- プラントの伝達関数を対話的かつ自動的に同定
- PID コントローラゲインの自動チューニング



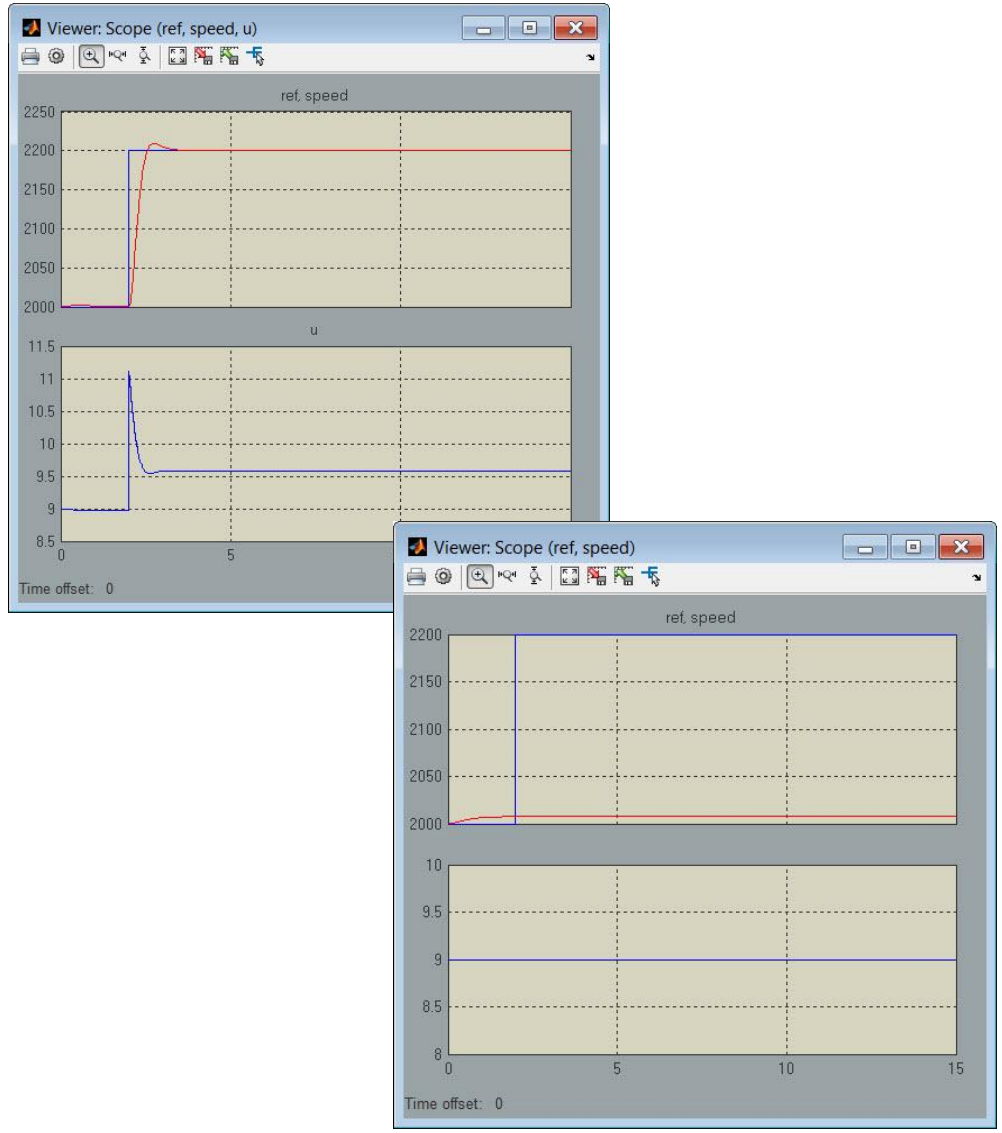
System Identification Toolbox が必要

PIDコントローラ設計のワークフロー



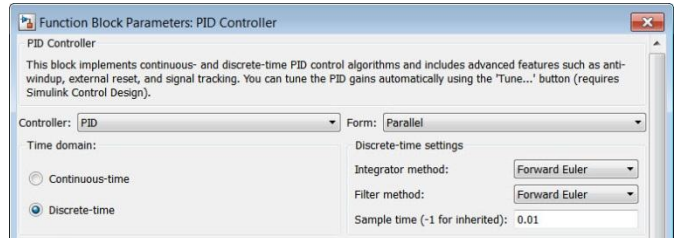
実装の課題

- 離散時間への変換
- 固定小数点プロセッサのための スケーリング
- 効率的なコード生成



ソリューション： PID Controllerブロック、チューニングアルゴリズム、固定小数点化ツール、およびコード生成ツール

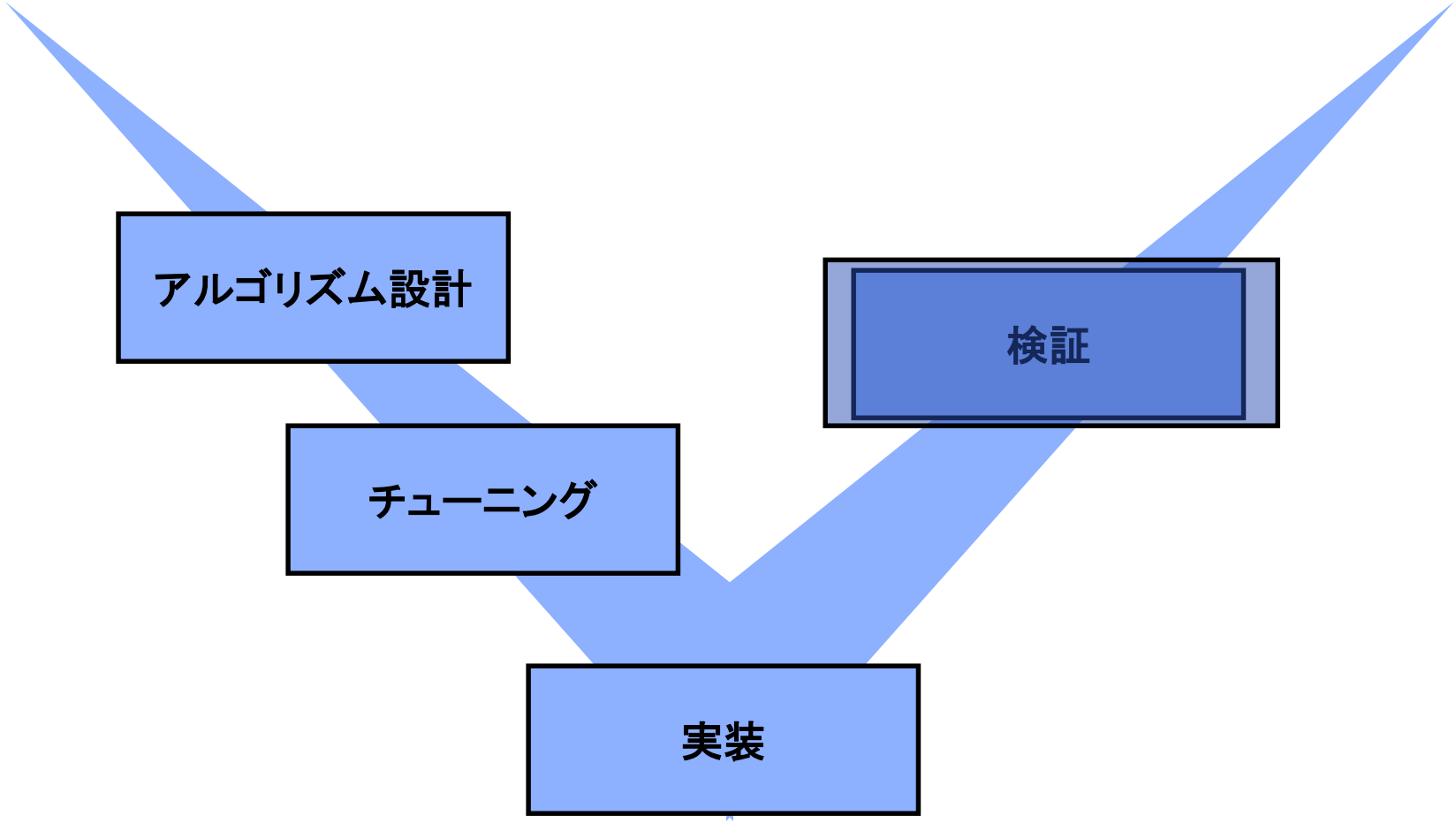
- 連続時間 or 離散時間による実装
- 離散時間コントローラを直接チューニング
- 信号とパラメータのデータ型を定義
- コード生成をサポート




```
#include "Fixed.h"
#include "Fixed_private.h"

int16_T error;
int16_T throttle;
D_Work DWork;
void Fixed_step(void)
{
  int16_T FilterCoefficient;
  FilterCoefficient = (int16_T)((int32_T)((int16_T)(29577L * (int32_T)error >>
    15) - (DWork.Filter_DSTATE >> 1)) * 2637L >> 11);
  throttle = (((int16_T)(25689L * (int32_T)error >> 14) >> 2) +
              (DWork.Integrator_DSTATE >> 1)) + (FilterCoefficient >> 2);
  DWork.Integrator_DSTATE = (int16_T)((int32_T)(int16_T)(27387L * (int32_T)error
    >> 14) * 5243L >> 19) + DWork.Integrator_DSTATE;
  DWork.Filter_DSTATE = (int16_T)(5243L * (int32_T)FilterCoefficient >> 15) +
    DWork.Filter_DSTATE;
}

void Fixed_initialize(void)
{
  throttle = 0;
  (void) memset((void *)&DWork, 0,
                sizeof(D_Work));
  error = 0;
  DWork.Integrator_DSTATE = 18432;
}
```

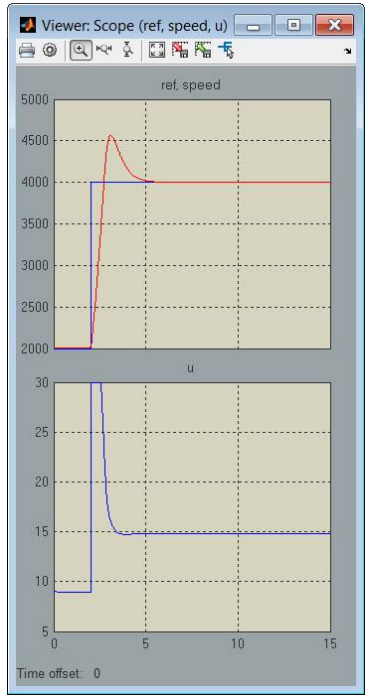
Fixed-Point Designer と
Embedded Coder が必要

PIDコントローラ設計のワークフロー



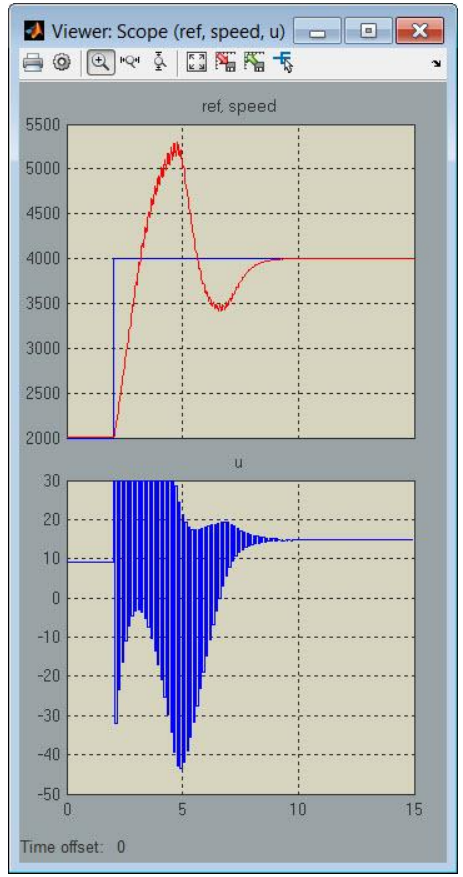
検証の課題

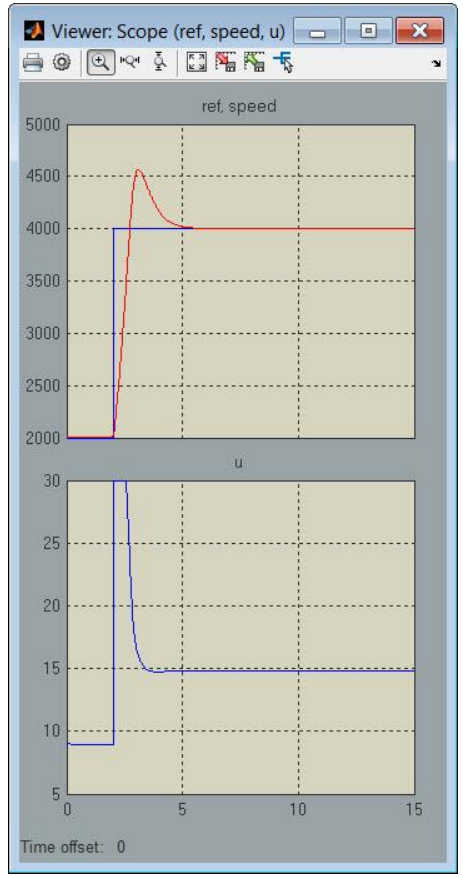
- 設計したコントローラの動作検証
- ハードウェア試作前のテスト
- 様々なシナリオを想定
  - 動作点外のテスト
  - 極限・異常テスト



ソリューション： シミュレーション

- デスクトップシミュレーション
  - 異なる動作条件でテスト
  - 設計フェイズ（精度）に合わせたテスト
- SILテスト
  - モデルと生成コードの等価性を検証
  - コントローラモデルから S-functionを生成し、Simulinkで生成コードをシミュレーション

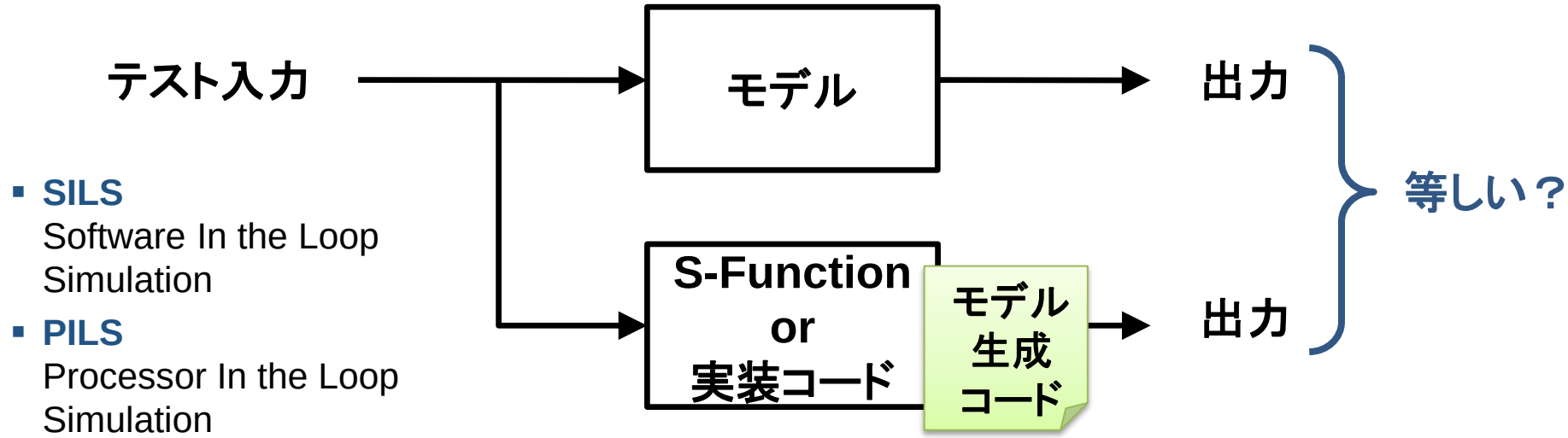




SIL: Software-in-the loop

コントローラモデルと生成コードの等価性検証

- モデルと生成コードに同じテスト入力を与えて等価性を検証
  - SILS： ホストPC上でコードを動作
  - PILS： ターゲットデバイス シミュレータ上でコードを動作



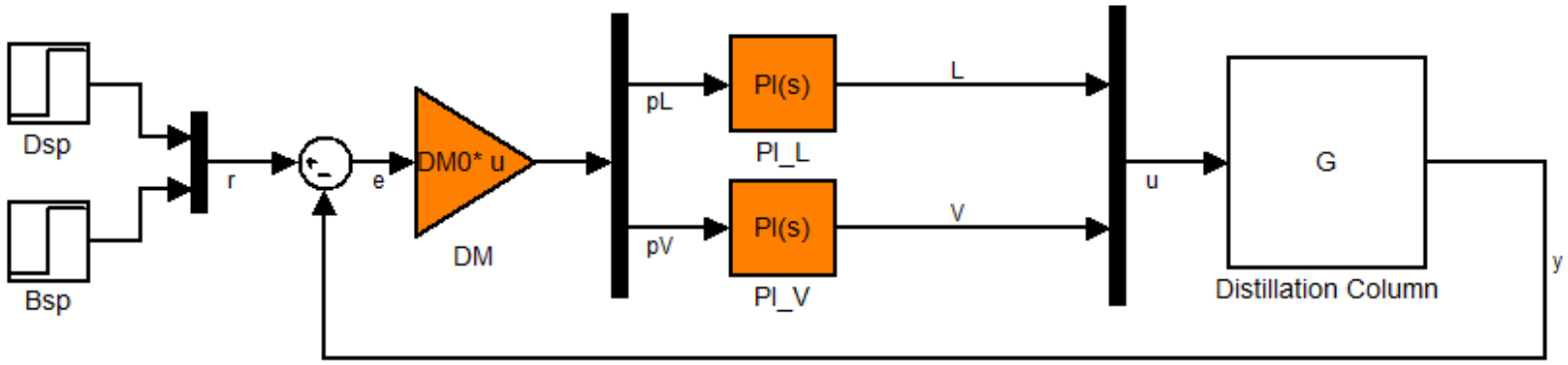
- **SILS**
  Software In the Loop Simulation
- **PILS**
  Processor In the Loop Simulation

次の2通りの方法を提供

1. モデル生成コードを呼び出すS-Functionブロックを生成する
2. モデル全体、またはModelブロックの動作モードをSIL/PILに設定する

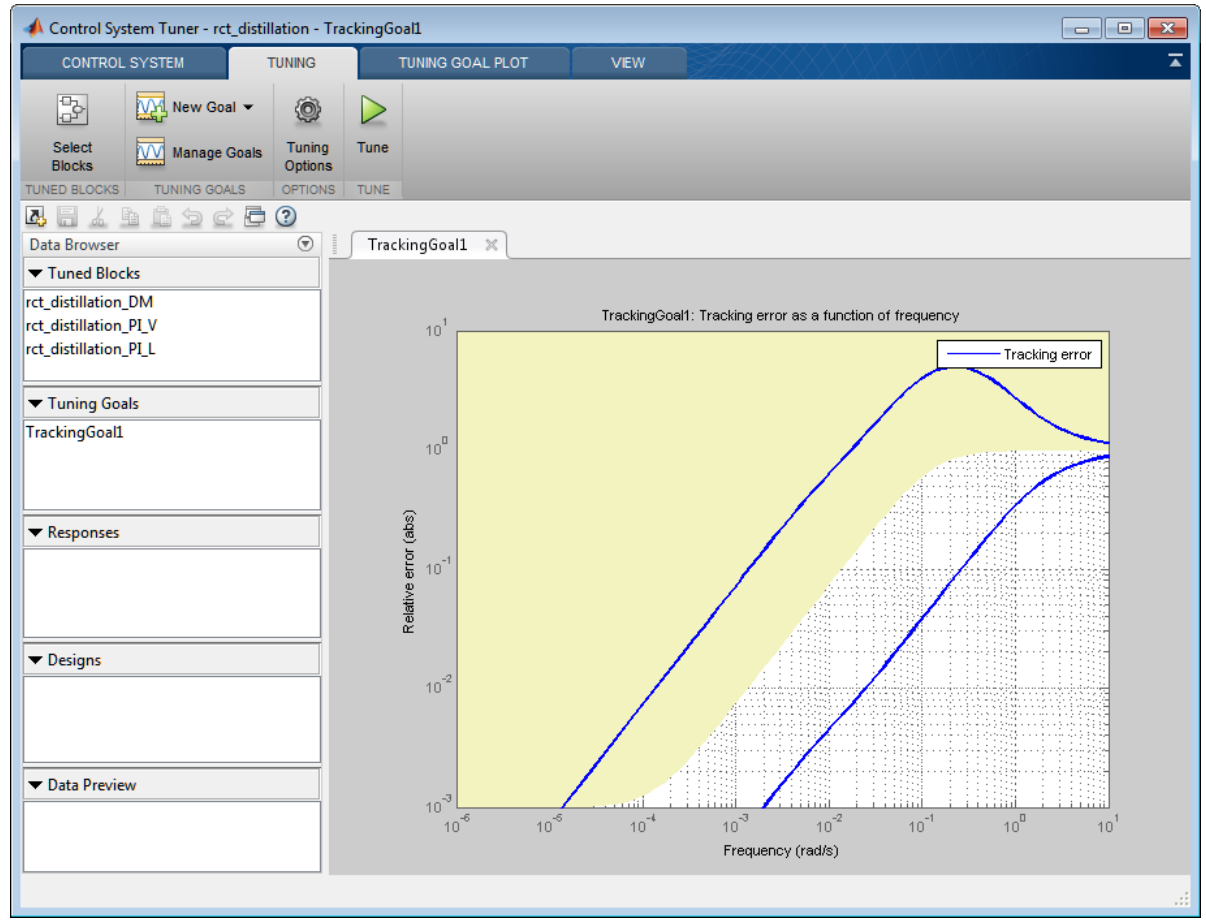
さらなる課題： 複雑なコントローラ

- マルチループや多入出力のような複雑な構造のPIDコントローラをどのようにチューニングするか？
  - ループ間の干渉によりチューニングが難しく、多くの時間と高い専門性を要する
  - 制御システム全体の最適な設計が難しい



ソリューション: 制御システム調整器（ Control System Tuner ）

- 複雑なコントローラを自動チューニング
- チューニングするSimulinkブロックと設計要求を指定



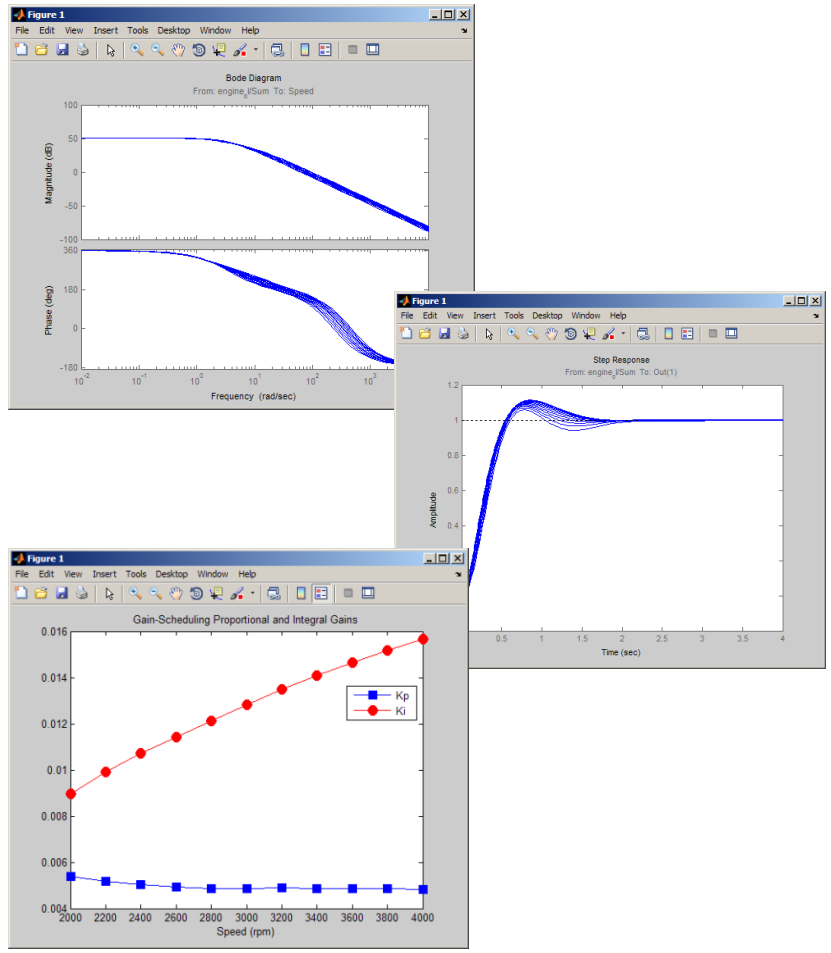
Robust Control Toolbox が必要

さらなる課題： ゲインスケジューリング

- 動作レンジの広い非線形なプラントをどのように制御するか？

ソリューション： ゲインスケジューリング

- 各動作点でプラントを線形化
- 各動作点で線形なコントローラを設計
- スケジューリングの仕組みを実装



まとめ

PIDコントローラの設計、チューニング、実装のワークフローを迅速かつ簡単に実現

- Simulink の PID Controller ブロック
- Simulink Control Design の PID チューニング法と PID 調整器
- Control System Toolbox の PID オブジェクトと解析・設計ツール
- Fixed Point Designer の固定小数点ツール
- Embedded Coder の自動 C コード生成機能
- Robust Control Toolbox の多入出力制御システムの自動チューニング機能

不安定なプラントをどのようにチューニング？

積分ワインドアップをどのように防止？

微分をどのように近似？

コントローラをどのように離散化/固定小数点化？

関連情報

- 製品情報
  http://www.mathworks.co.jp/products/
  - Simulink
  - Control System Toolbox
  - Simulink Control Design
  - Simulink Fixed Point
  - Embedded Coder
  - Robust Control Toolbox

- 次のステップ
  - 評価版ダウンロード
  - お問い合わせ先
    http://www.mathworks.co.jp/contact

- セミナー情報
  - ライブWebセミナー
    http://www.mathworks.co.jp/webinars
  - オンデマンドWebセミナー
    http://www.mathworks.co.jp/recordedwebinars
  - セミナー
    http://www.mathworks.co.jp/seminars