実用的なCコード自動生成ツール導入の手引き
～事例＆FAQから学ぶEmbedded Coder®レッスン～

MathWorks Japan
アプリケーションエンジニアリング部
シニアアプリケーションエンジニア
山本　順久

本セミナーではR2018aを用いています

アジェンダ

- はじめに：本セミナーの内容・目的について
- よくあるトラブル要因＆その対策
- ケーススタディ：制御モデルからの組込みコード生成
- まとめ

はじめに：モデルベースデザイン（MBD）のご利益のおさらい

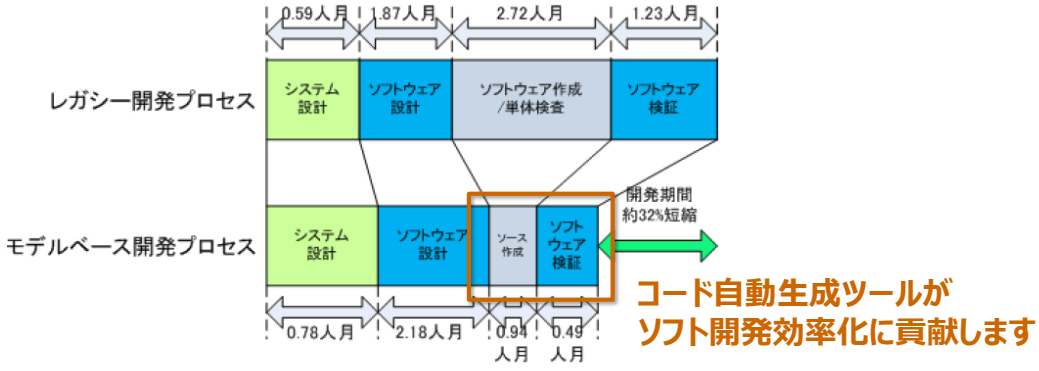


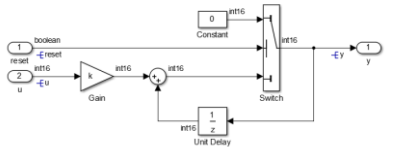
図 31 モデルベース開発とレガシー開発の工数

※ 2013/02 IPA モデルベース開発ツールを活用した際のコストの効果・検証 実施報告書から抜粋
（半導体製造装置内のボンディング制御コントローラソフト開発を対象に評価した例）

http://www.ipa.go.jp/files/000026867.pdf

Embedded Coder®：
モデル/M-ファイルから量産/組込み用C/C++コードを自動生成します

モデル（ロジック）



コード自動生成

```
void rst_cntr_step(void)
{
  if (reset) {
    y = 0;
  } else {
    y += (int16_T)(k * u);
  }
}
```

ソフト仕様

- 関数
  - 名前、引数
- 変数/定数属性
  - 名前、記憶クラス、データ型
- コード書式

- コーディングレス
  ただし、モデリングやコード生成準備作業は必要
- モデルの誤解釈混入リスクを解消
- モデル・コード間の乖離を防止
- ソフト屋さん以外の開発者も記述・理解しやすい

一方ではこんな声を頂くことも・・・

みなさま

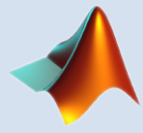
導入評価

MBD＆コード自動生成で
ソフト開発改善目指そう！

あれっ、思ったようなコード
にならない・・・どう適用し
たらいいんだろう・・・

マスワークス

技術支援



よく頂く質問やコード生成
のためのチップス・おススメ
情報を紹介します

本セミナーの内容

5

どんなトラブル要因でつまづいている事が多い？ 1/3

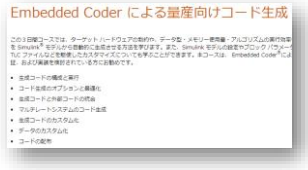
要因

ツールの機能・使い方を把握していない。我流で評価・検討されている。

対策

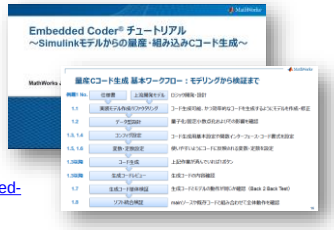
評価・運用の際に不明な点があればまずはマスワークスに問合せを。
マニュアル以外にも有償トレーニング、無料のチュートリアル・チップス集を提供しています。

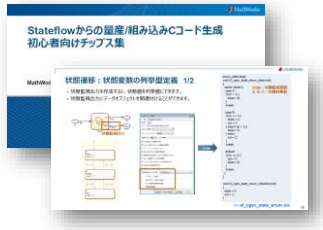
有償トレーニング



https://jp.mathworks.com/training-schedule/embedded-coder-for-production-code-generation.html

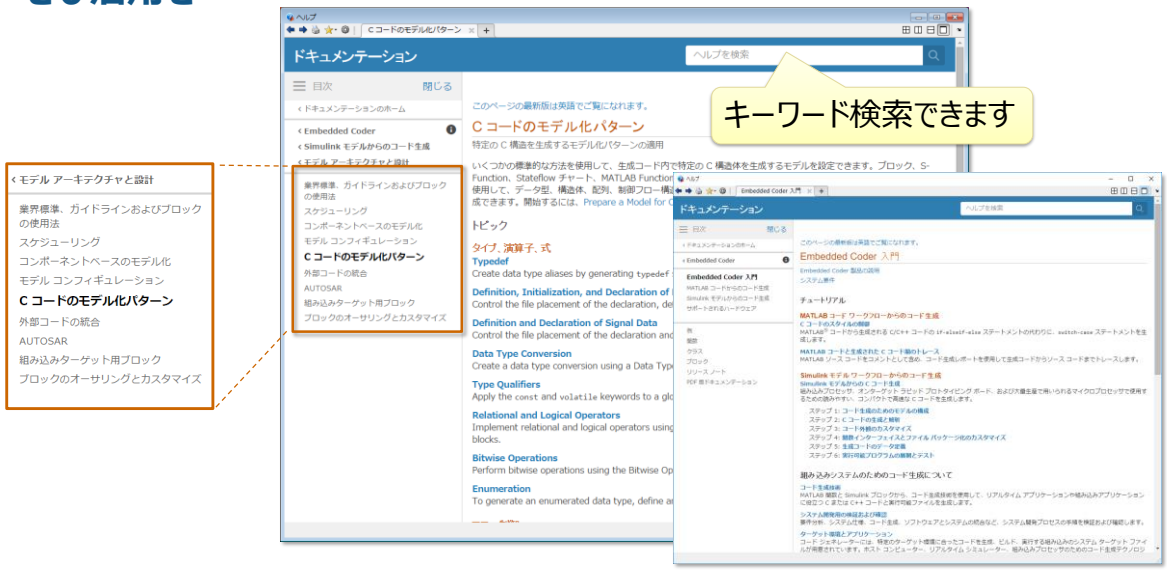
チュートリアル・チップス集





6

MATLABヘルプマニュアルにも豊富な参考情報・例題が用意されています ぜひ活用を



7

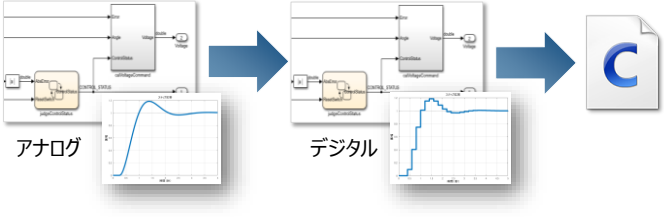
どんなトラブル要因でつまづいている事が多い？ 2/3

要因

コード生成用途としては不適・不十分なモデルを対象としている。

対策

上流工程のモデルはシミュレーション可能でもコード生成には向かない、というケースが多々あります。量産/組込み用途では、まずモデル構造・部品設計やI/O・データ設計を実施して、その成果をモデルに反映することを推奨します。コード改善のためにモデルの手直しを行うこともあります。



- モデル構造/モジュール設計は十分？
- 量産コードに不適なブロックはない？
- データ型は適切？
- コード生成用コンフィグ設定は適切？

8

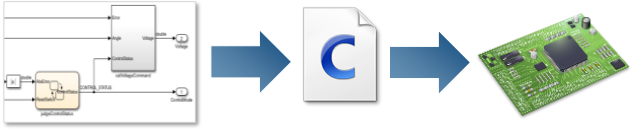
どんなトラブル要因でつまづいている事が多い？ 3/3

要因

モデルとコードの切り分け / 階層設計が考慮されていない

対策

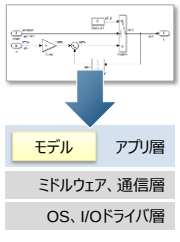
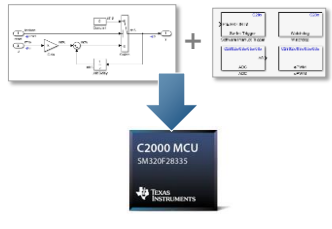
ソフト全体（アプリ層＋OS/HW層等）をモデルに記述できない / すべきでないケースが多々あります。コード生成の前にまずはモデルで記述すべき処理とコードとして利用すべき / 残すべき処理を整理する必要があります。



- ソフト全体のどこをモデルで書く？
- 周期処理 or 非周期処理？
- 他ソフトとのデータのやりとりは？
- mainルーチン/OS/ファームウェアとの関係は？
- デバドラやAPI呼び出しは必要？

9

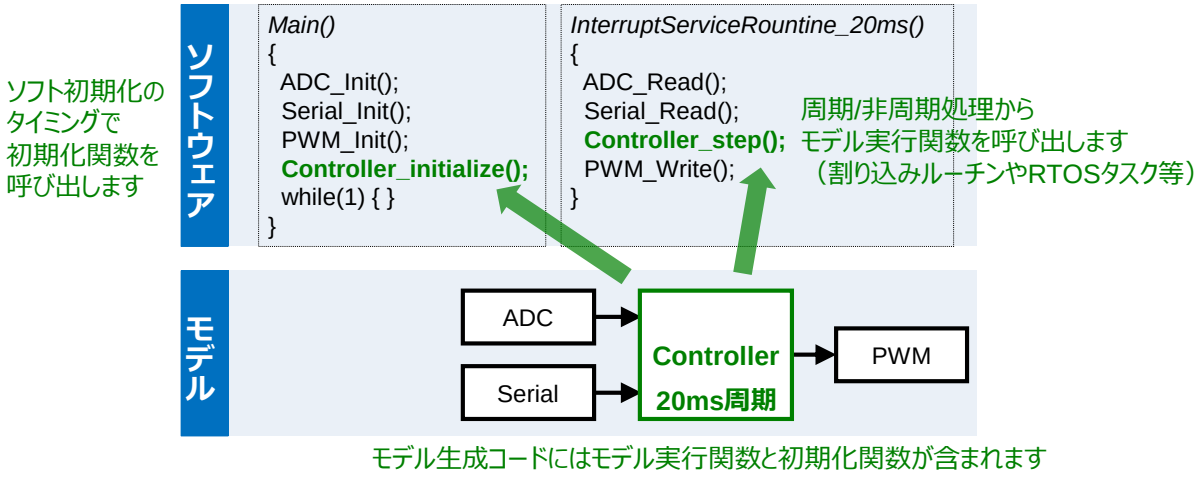
Embedded Coder 生成コードの利用には主に2つのアプローチがあります

	ソフトコンポーネント型（基本）	モデルビルド型（応用）
利用イメージ	**アプリの一部をモデル化、生成されたコードを既存ソフトに組み込む** 	**MCU/DSP用ブロックライブラリ利用、モデルを直接実装する** 
主な用途	既存ソフトとの差分開発 HW/OS非依存アプリの開発	MCU/DSPを用いた制御実験 （On Target RCP）
特徴	標準ブロックのみ使用	専用ブロックライブラリを一部使用
I / O	グローバル変数 / 引数	AD/DA, PWM, CAN等のI/Oブロック
注意点	ビルド・結合は手動で行う必要アリ	対応MCU/DSP限定、HW知識も必要
対応 MCU/DSP	生成コードはANSI/ISO-C準拠なので **ほぼ全てのMCU/DSPに対応可能**	TI C2000, STMicro STM32F4 等 （MathWorksまたはベンダから提供）

10

ソフトコンポーネント型の利用イメージ

- データ受け渡しはグローバル変数 or 引数
- 割り込みやポーリング処理でモデル実行関数を呼び出します



モデル生成コードにはモデル実行関数と初期化関数が含まれます

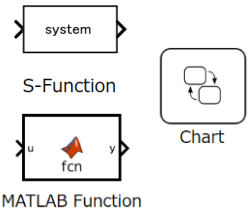
外部 / 手書きコードとモデルを組み合わせるには？

- 自作ライブラリ/MCU専用ライブラリ関数を呼び出したい
- モデルで書けない・書きづらいデバドラ/APIを呼び出したい
- コード→モデルベース開発移行期に既存ソフトを流用したい

解決策

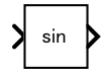
外部コード関数の取り込み

シミュレーション時は関数実行、コード生成時は関数呼び出し処理を生成します。シミュレーション/コード生成で呼び出す関数を別にすることも可能です。



特定処理の関数呼び出し置換

Code Replacement Library: CRL
四則演算、数学関数等をMCU専用/ユーザ定義の関数呼び出しに置換します。



y = mysin(x);

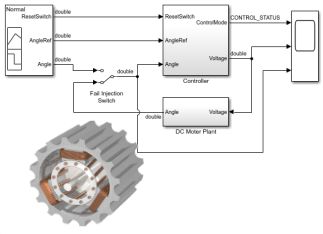
生成コードへの自作コード挿入

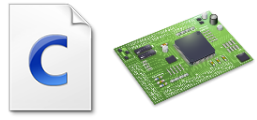
生成コード特定箇所に任意コードを挿入できます（シミュレーション時は無視されます）。

ケーススタディ：モータ制御モデルからの組込みCコード生成

このモデル、シミュレーションで
うまく動いたんで製品化まで
持っていきたい。できそう？

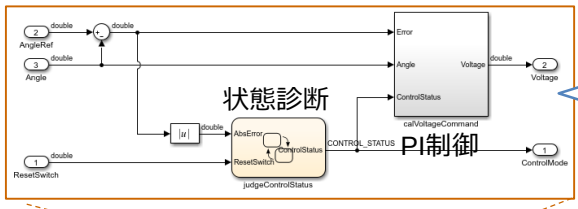
コード自動生成ツール使えば
ソフト化できると思いますよ。
もらったモデルで試してみます。



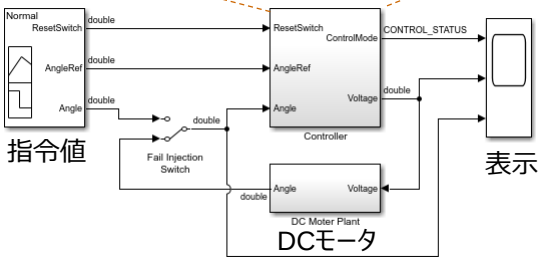


制御屋さん　　ソフト屋さん

モータ制御モデルの内容を見てみたら・・・



モータ制御ロジック



指令値　　表示

DCモータ

DCモータ制御システムモデル

この部分をソフト化したいけど、実行
周期は不明だしデータ型もdouble。
このままじゃソフト化できないよ・・・
どう手直しすればいい？

ちなみに・・・デフォルト設定で生成されるCコード例

```
void controlMotor_step(void)
{
  real_T rtb_error;
  real_T rtb_Abs1;
  boolean_T rtb_Compare;
  real_T tmp;
  rtb_error = controlMotor_U.AngleRef - controlMotor_U.Angle;
  rtb_Abs1 = fabs(rtb_error);
  if (controlMotor_DW.temporalCounter_i1 < 255U) {
    controlMotor_DW.temporalCounter_i1++;
  }

  if (controlMotor_DW.is_active_c1_controlMotor == 0U) {
    controlMotor_DW.is_active_c1_controlMotor = 1U;
    controlMotor_Y.ControlStatus = WAITING;
  } else {
    switch (controlMotor_Y.ControlStatus) {
     case WAITING:
      if (rtb_Abs1 < 10.0) {
        controlMotor_Y.ControlStatus = NORMAL;
      } else {
        controlMotor_Y.ControlStatus = WARNING;
        controlMotor_DW.temporalCounter_i1 = 0U;
      }
      break;
```

- 関数名は
  モデル実行関数：モデル名_step
  初期化関数：モデル名_initialize
- グローバルデータはCoder独自の構造体
- データ型はモデル設定を反映
  auto設定だと通常doubleになります

正直言ってちょっと使いづらい・・・

仕様/機能開発モデルから量産/組込みコード生成までの道のり

①	ソフト仕様の設計	生成コードに対するソフト構造・関数・データ等を設計します
②	ソフト・システムモデルの修正	上記のソフト仕様に合わせてモデルを修正・設定します
③	ソフトモデルのリファクタリング	コード生成に不適なブロックやモデリングがあれば見直します
④	ソフトモデルのコード生成用設定	コード生成用の情報をモデルに設定・追加します
⑤	ソフトモデルからコード生成	上記準備作業が済めば1ボタンで生成できます
	生成コードの利用・動作検証	

① ソフト仕様の設計例：
生成コードに対するソフト構造・関数・データ等を設計します

項目	今回のソフト仕様
実行周期	20ms周期
関数名	controlMotor / controlMotor_Init
関数引数	void-void（引数無し）
関数利用形態	モデル関数をmainソース / スケジューラから呼び出す
データ型	制御演算は単精度浮動小数点型、 必要に応じて整数型に設定
データ名	モデル内シンボルをそのまま使用
データ記憶クラス	適宜設定

※コード生成ツール機能を考慮してソフト仕様を設計する必要があります

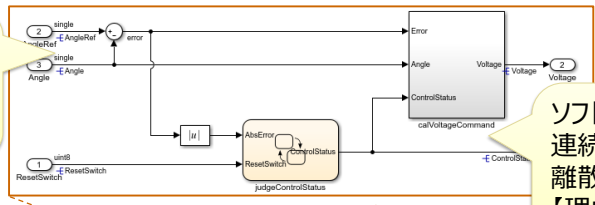
```
float Voltage;

void controlMotor(void)
{
  float error;
  error = AngleRef - Angle;

  …

  Voltage = …;
}

void controlMotor_Init(void)
{
  …
}
```

こんなイメージのコードを生成したい

17

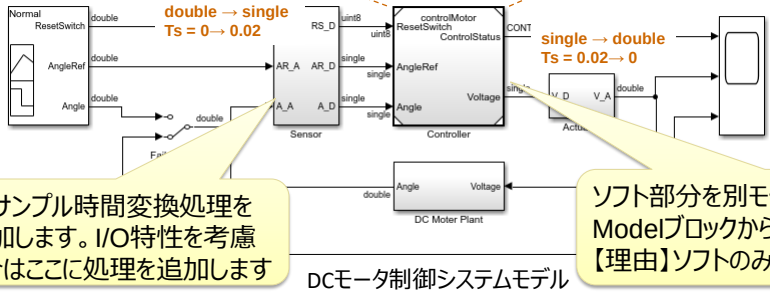
② ソフト・システムモデルの修正例：ソフト仕様に合うようモデルを修正します

ソフトモデル内のデータ型を
所望の型に設定します
【理由】データ型がそのまま
コードに反映されるため



モータ制御モデル

ソフトモデルが可変ステップ/
連続時間の場合、固定ステップ/
離散時間に変換します
【理由】可変ステップでは
コード生成できないため



データ型・サンプル時間変換処理を
前後に追加します。I/O特性を考慮
したい場合はここに処理を追加します

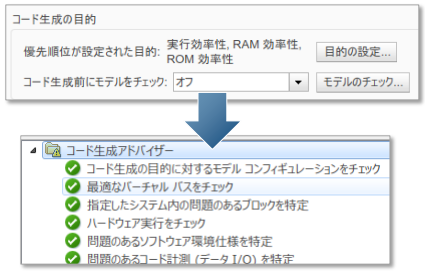
DCモータ制御システムモデル

ソフト部分を別モデルファイルにして
Modelブロックから呼び出します
【理由】ソフトのみをコード対象とするため

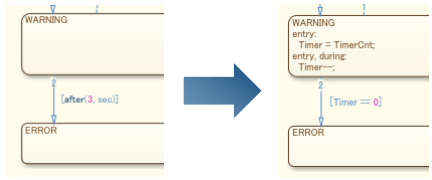
18

③ ソフトモデルのリファクタリング例：
コード生成に不適なブロックやモデリングがあれば見直します

コード生成適応可否の事前チェック
コード生成アドバイザーで簡単に
チェック＆修正できます
※全チェックをパスする必要は必ずしもありません



コード生成に向けたモデル修正例
afterアクション記述を減算カウンタ処理に変更
【理由】カウンタ変数のシンボル・記憶クラスを
設定可能にするため



19

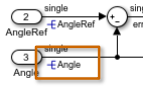
④ ソフトモデルのコード生成用設定例：
コード生成用の情報をモデルに設定・追加します

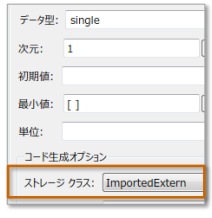
システムターゲットファイル＆言語の設定
ert.tlc＆言語を設定します（他のtlcを用いる場合もあります）

関数仕様を設定
インターフェース設定で関数名や引数を設定します

データ仕様を設定
モデル内の信号・状態・パラメータに対するコード内データ名や記憶クラスを設定します

extern real32_T Angle;





20

コード生成用設定適用前後の生成コード比較

```
void controlMotor_step(void)
{
  real_T rtb_error;
  real_T rtb_Abs1;
  boolean_T rtb_Compare;
  real_T tmp;
  rtb_error = controlMotor_U.AngleRef - controlMotor_U.Angle;
  rtb_Abs1 = fabs(rtb_error);
  if (controlMotor_DW.temporalCounter_i1 < 255U) {
    controlMotor_DW.temporalCounter_i1++;
  }

  if (controlMotor_DW.is_active_c1_controlMotor == 0U) {
    controlMotor_DW.is_active_c1_controlMotor = 1U;
    controlMotor_Y.ControlStatus = WAITING;
  } else {
    switch (controlMotor_Y.ControlStatus) {
      case WAITING:
        if (rtb_Abs1 < 10.0) {
          controlMotor_Y.ControlStatus = NORMAL;
        } else {
          controlMotor_Y.ControlStatus = WARNING;
          controlMotor_DW.temporalCounter_i1 = 0U;
        }
        break;
```

```
void controlMotor(void)
{
  real32_T rtb_error;
  boolean_T rtb_Compare;
  real32_T Abs1;
  real32_T tmp;
  rtb_error = AngleRef - Angle;
  Abs1 = fabsf(rtb_error);

  switch (ControlStatus) {
    case WAITING:
      if (Abs1 < MaxError) {
        ControlStatus = NORMAL;
      } else {
        ControlStatus = WARNING;
        Timer = ((uint8_T)TimerCnt);
        Timer = (uint8_T)(Timer - 1);
      }
      break;
```

適用前コード
（主にdouble型）

適用後コード
（主にfloat型）

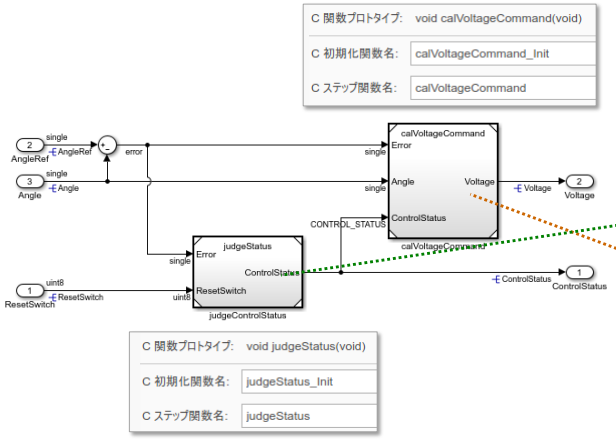
可読性・効率性が
大幅に改善されます

項目	適用前	適用後
ソース行数	289	257
グローバル変数RAMサイズ	55 byte	23 byte
ローカル変数スタックサイズ	53 byte	33 byte

※モデル生成コードレポートのコード静的メトリクスを抜粋

大規模モデル開発にはモデル＆生成コードのファイル分割が有効です

Modelブロックで別モデル参照化→生成コードも別関数・ソースファイルになります
変更がないモデルは自動的にコード生成処理を省略、差分/派生開発時のコード生成時間短縮にも貢献します




```
void controlMotor(void)
{
  real32_T rtb_error;
  rtb_error = AngleRef - Angle;
  judgeStatus(&rtb_error);        状態診断
  calVoltageCommand(&rtb_error);  PI制御
}
```

まとめ

みなさま

技術支援

マスワークス

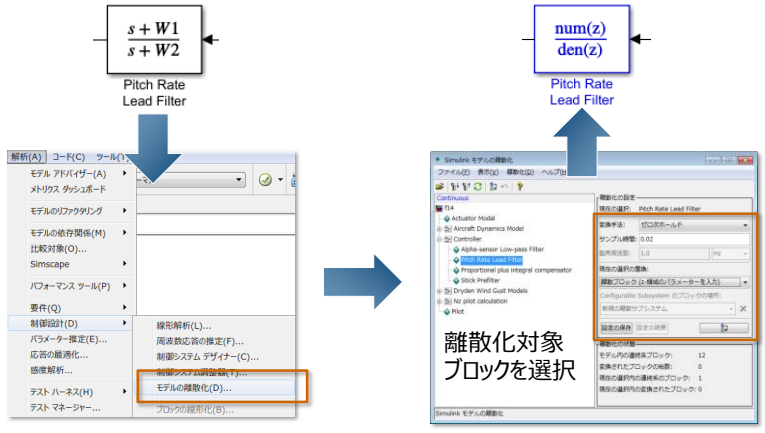
あれっ、思ったようなコードにならない・・・どう適用したらいいんだろう・・・

本セミナーでざっくりとコード生成までの流れやお役立ち情報、トラブル処方箋を紹介しました。
初心者/導入評価の方はまずはトレーニング・チュートリアルから始めることをおススメします。
「こんなコードを生成したい」等ご質問・ご要望のある方はぜひマスワークスにお問合せください。

23

連続時間ブロック→離散時間変換の際はモデルの離散化機能が便利です

Control System Toolbox™：モデルの離散化機能



※ 離散時間化すると一般に入出力特性が変化しますので、システム性能の再評価が必要となります
https://www.mathworks.com/help/releases/R2018a/simulink/ug/model-discretizer.html

24

本セミナーで紹介しているチュートリアル・チップス集をご希望の方は、
当社技術サポートまでお問い合わせください。
資料＆例題ファイルを提供いたします。

https://jp.mathworks.com/support/contact_us.html