MATLABコードからの組み込み用Cコード生成のワークフローと最適化のコツ

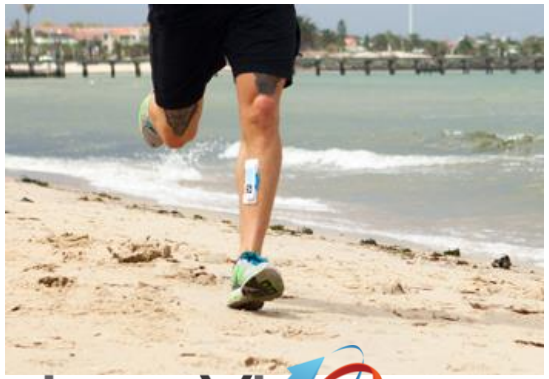
MathWorks Japan
アプリケーションエンジニアリング部
プリンシパルアプリケーションエンジニア
松本　充史

MATLAB Coder™ユーザ事例

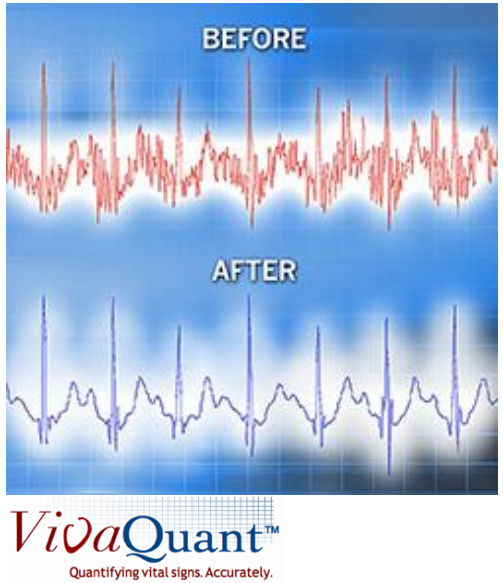
- MATLAB®を使うことで・・・
  - 新しいアイデアを素早く試す
  - テスト、解析によりシステムの特性評価
  - 短期間で高品質なCコード生成

https://jp.mathworks.com/company/user_stories/delphi-develops-radar-sensor-alignment-algorithm-for-automotive-active-safety-system.html

https://jp.mathworks.com/company/user_stories/respiri-develops-mobile-app-for-wheeze-detection-and-asthma-management.html



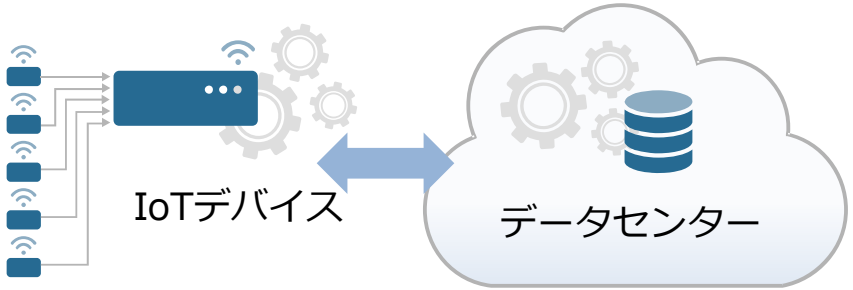
https://jp.mathworks.com/company/newsletters/articles/developing-motion-analysis-algorithms-for-medical-sports-and-occupational-safety-applications.html



https://jp.mathworks.com/company/user_stories/vivaquant-accelerates-development-and-validation-of-embedded-device-for-ambulatory-ecg-sensing.html

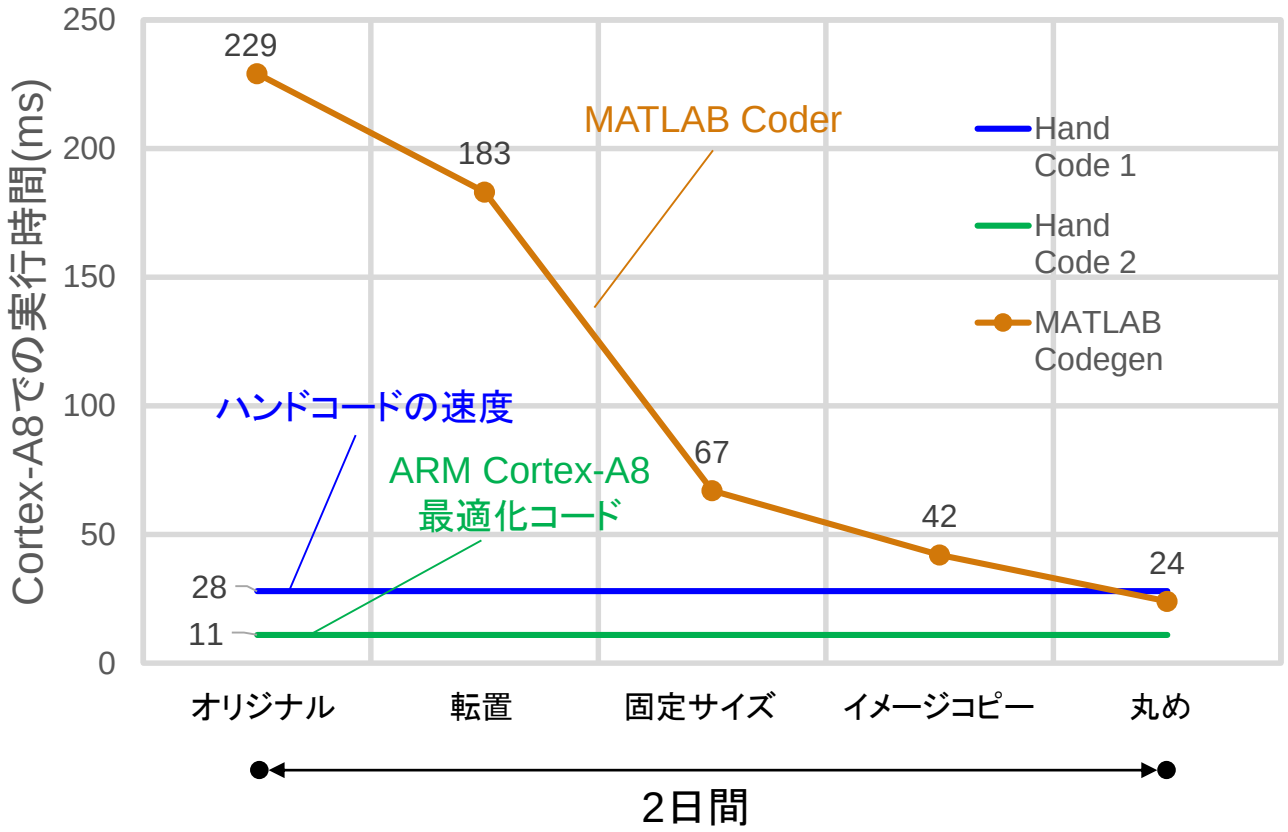
国内メーカーでのMATLAB Coder導入事例
携帯型IoTデバイス

- Cコードの生成効率が悪い➔MathWorksサポート
  - > バッテリー駆動IoT機器の信号処理アルゴリズム実装（Low Power←→高度な信号処理）
  - > ターゲットプロセッサコア：Cortex-M4
  - > オプション設定方法
  - > Embedded Coder®追加導入
- 更なる効率化のサポート
  - > Cortex-M4向けコード置換ライブラリ（CRL）適用
- MATLABアルゴリズム開発~実装の開発効率化を達成



MATLAB Coderを使った最適化事例
画像処理認識システム

- 画像処理アルゴリズムをARM Cortex-A8コアに実装
- MATLABコード最適化により、生成Cコードの実行時間が短縮
- 2日間の作業で手書きCコード超え



このセッションのポイント

MATLAB Coderの
基本的な利用方法

効率的なC生成のための
コーディングテクニック

コード置換ライブラリと
System Object

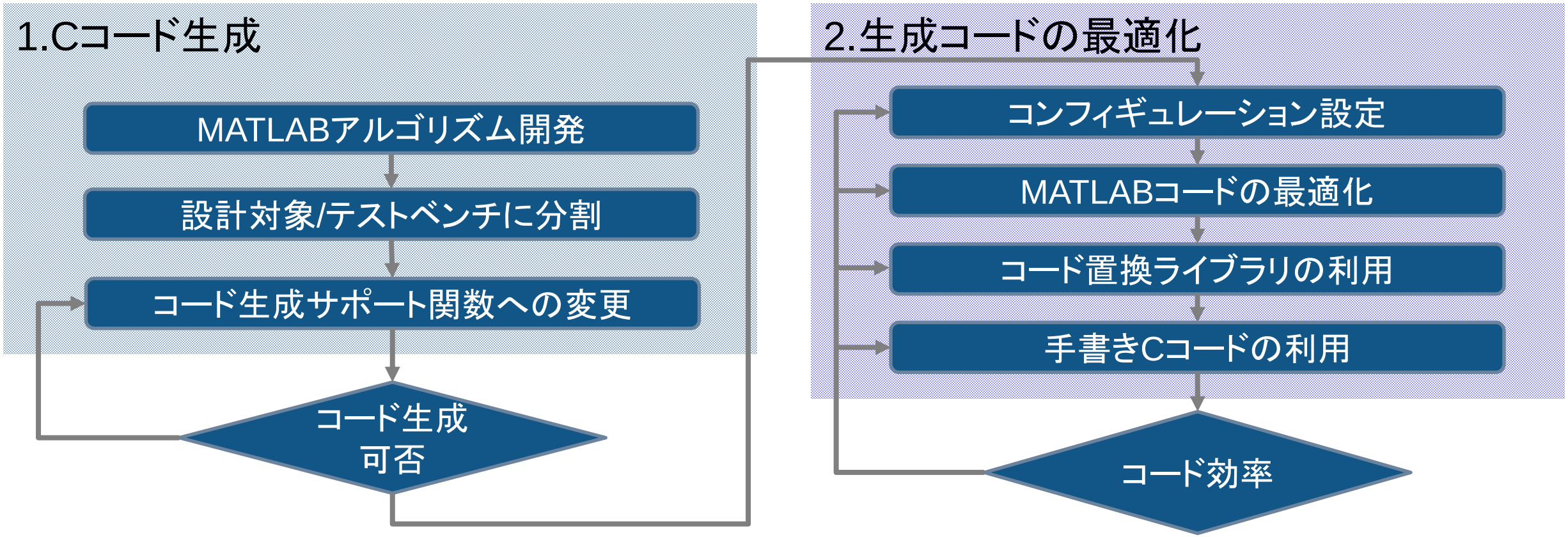
このセッションのポイント

MATLAB Coderの
基本的な利用方法

効率的なC生成のための
コーディングテクニック

コード置換ライブラリと
System Object

MATLABプログラムからのCコード生成ワークフロー概要



MATLABコードの構成は
Cコード生成対象をFunctionにして、テストベンチを用意

オリジナル
（信号生成+設計対象+可視化）

テストベンチ
- 記述内容
  - 信号生成
  - 生成対象呼び出し
  - 可視化・解析
- コードの制限
  - なし

生成対象
- 記述内容
  - 引数有りの function MATLABファイル
- コードの制限
  - コード生成対応関数・文法

Cソースコード

MATLAB Coderプロジェクトの作成

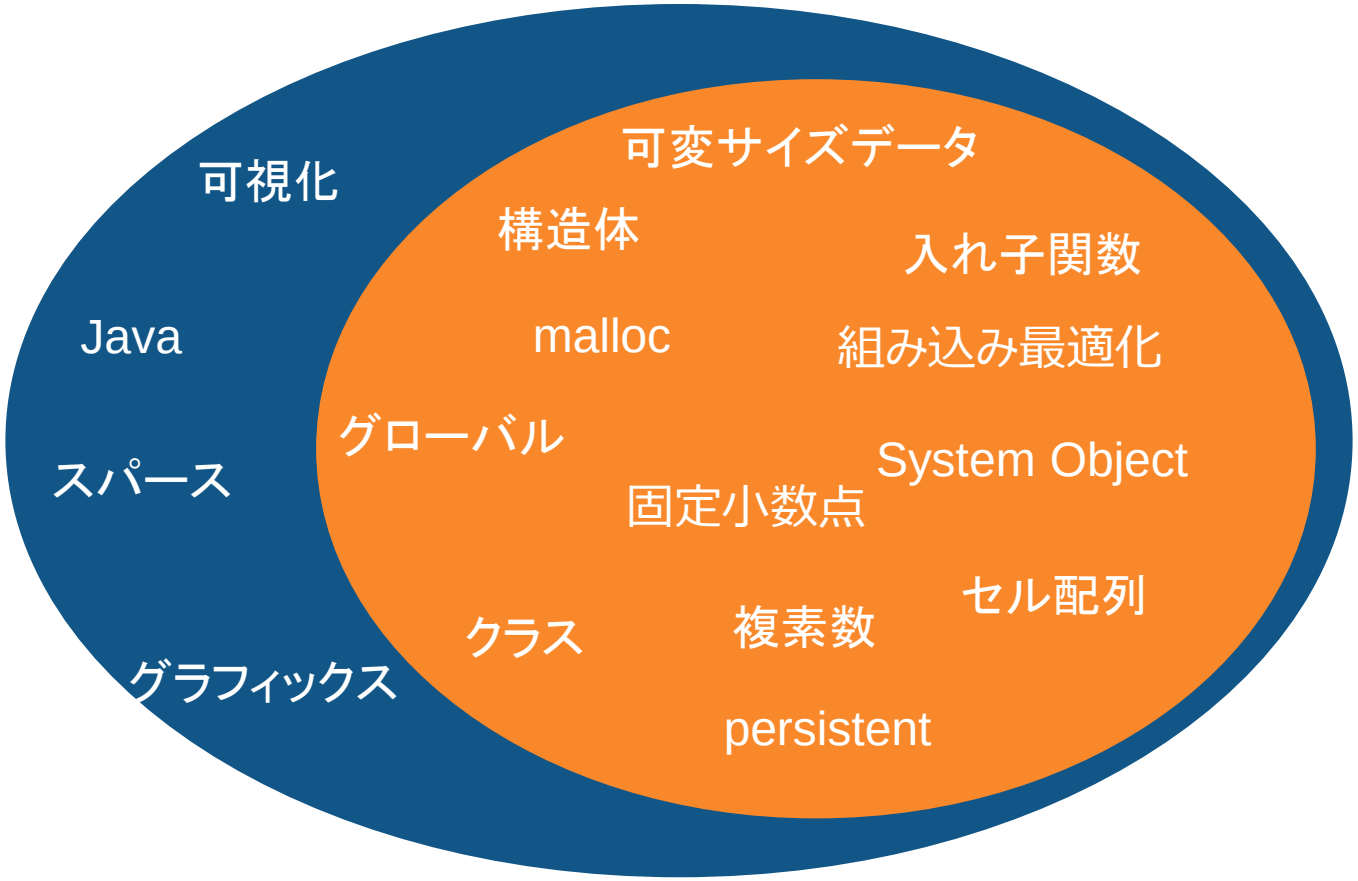
- アプリとコマンドでコード生成の設定および実行
  - アプリ：初心者、設定に慣れない人向け➔アプリ/MATLAB Coderまたは `>> coder`
  - コマンド：経験者、設定が固まっている人向け `>> codegen filename -options`

詳細設定画面

コード生成に対応するMATLAB関数

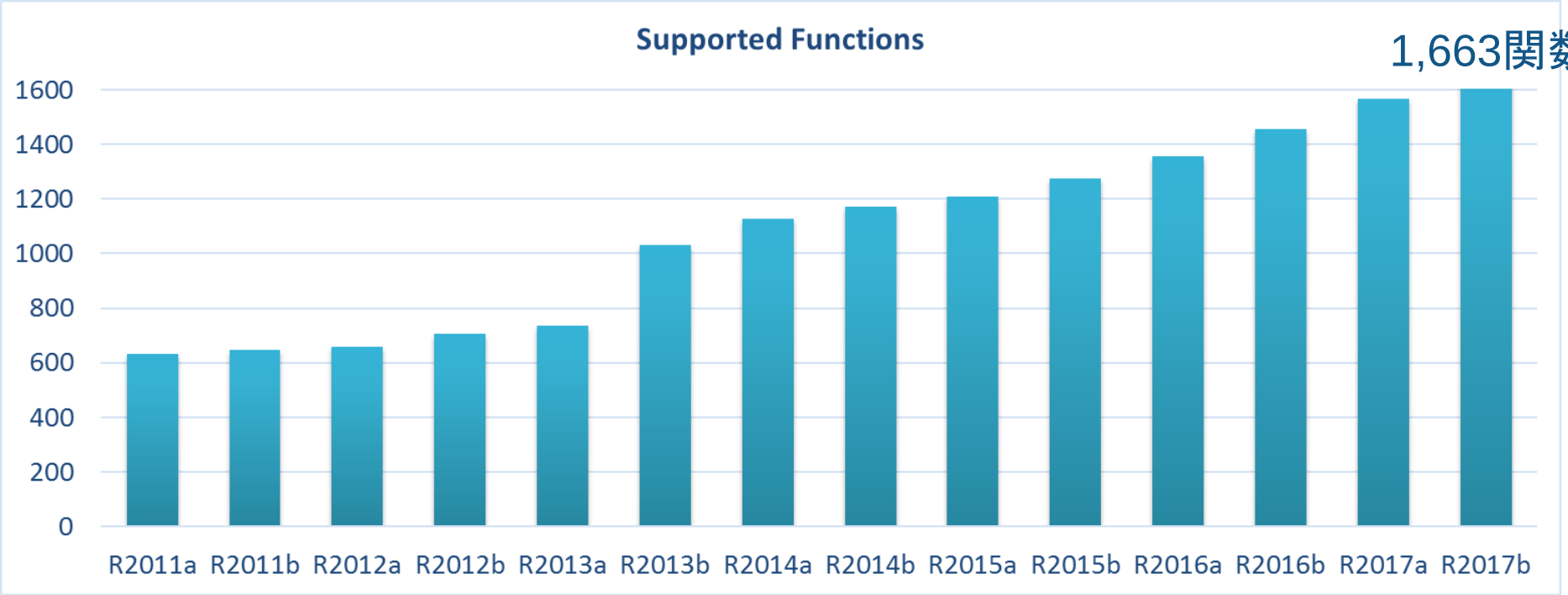
- MATLAB Coderドキュメント
  >コード生成のためのMATLABプログラミング
  >言語、関数、オブジェクトのサポート

- Webドキュメント
  http://www.mathworks.com/help/coder/language-supported-for-code-generation.html



Cコード生成対応関数はバージョンアップごとに増加

数年前の未対応関数も現在は対応しているかも

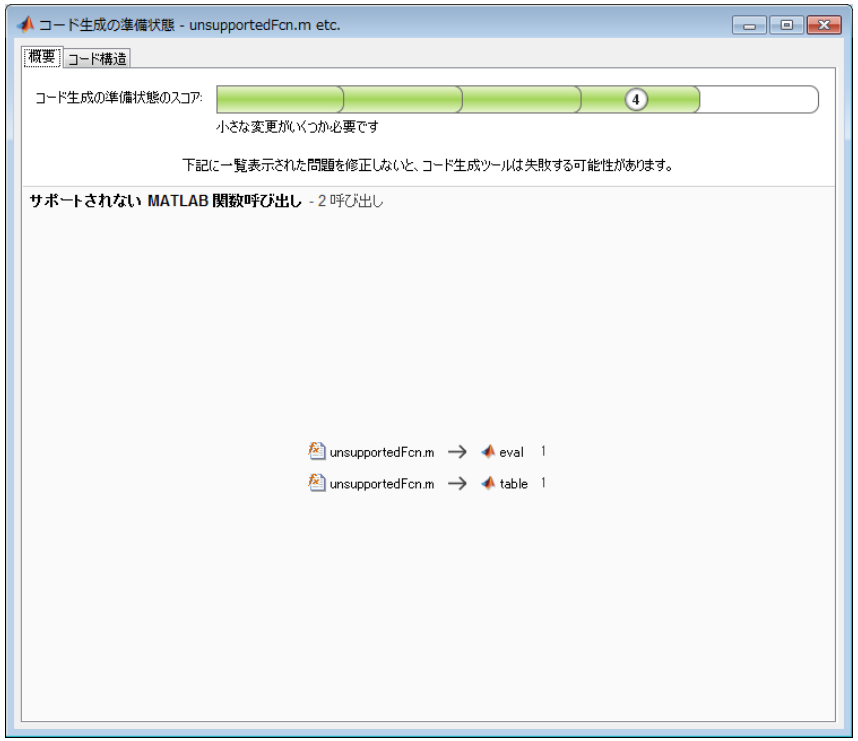


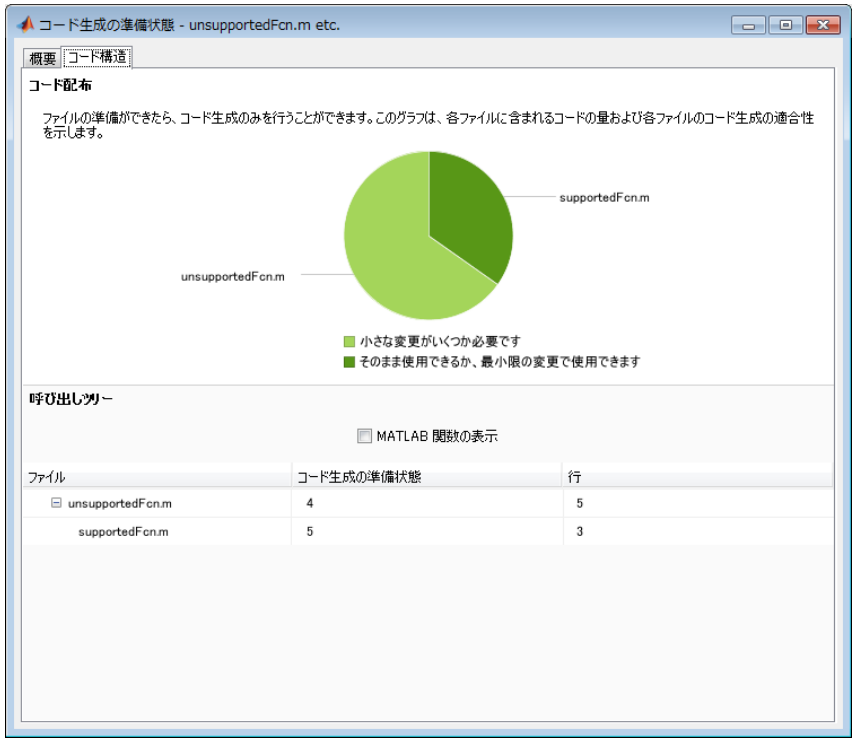
近年対応関数が増加しているToolbox

Statistics and Machine Learning Toolbox™, Fixed-Point Designer™
Image Processing Toolbox™, Computer Vision System Toolbox™
Wavelet Toolbox™, DSP System Toolbox™, Audio System Toolbox™など

コード生成対応の事前簡易チェック

- coder.screenerによるコードのチェック
  - >> coder.screener('filename') により、MATLAB Coderアプリ実行前に確認可能
  - ユーザ関数ごとにチェックして問題箇所の表示

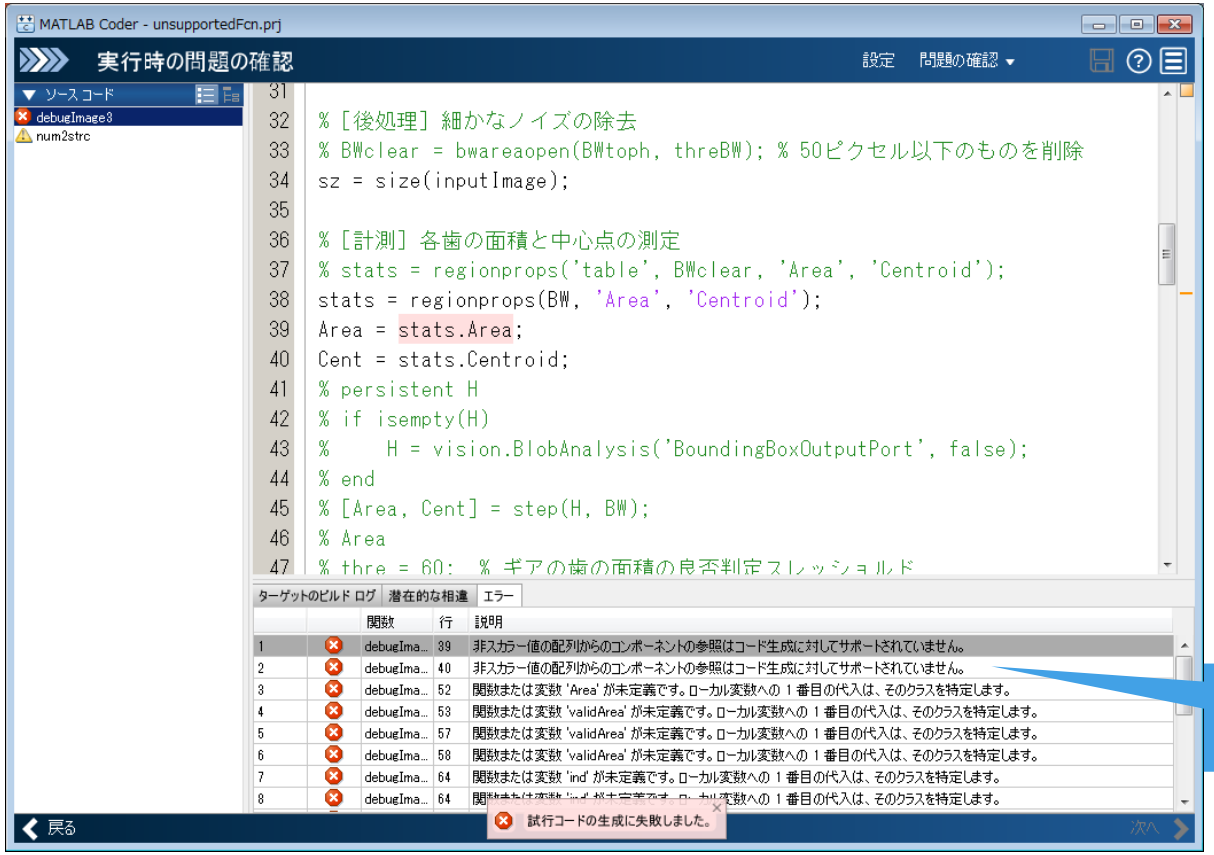




coder.screenerの確認結果例

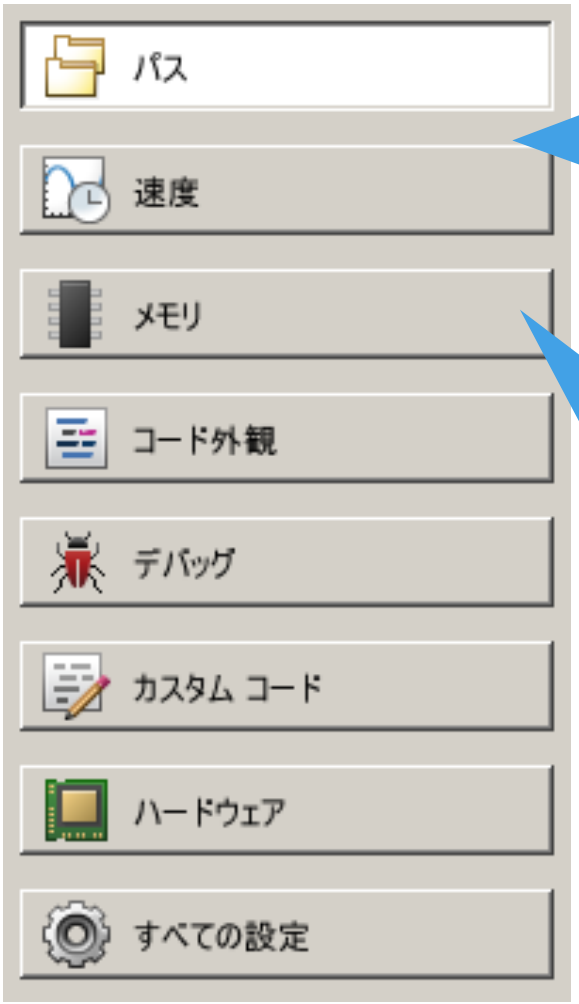
コード生成対応の詳細チェック方法

- MATLAB Coderアプリの「実行時の問題の確認」画面でチェック
  - 未対応関数、文法の検出
  - 該当行と説明のレポート表示

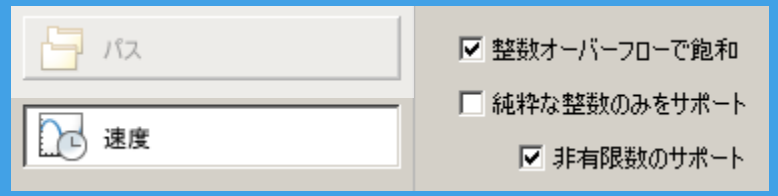


エラー内容の表示

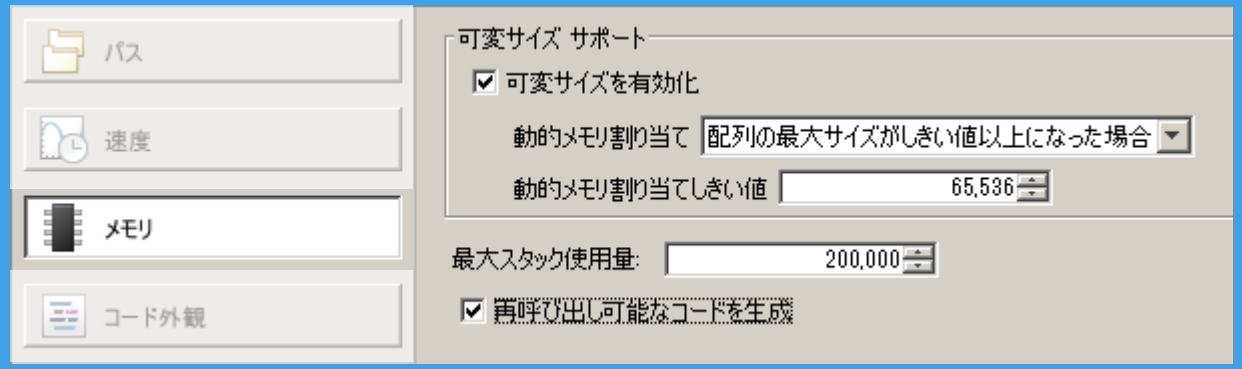
コード生成/詳細設定の例



- 「整数オーバー･･･」を有効化すると飽和処理が追加され、速度低下
- 固定小数点プロセッサ向けは浮動小数点コードを生成しないよう「純粋な整数･･･」を有効化（Embedded Coderのオプション）

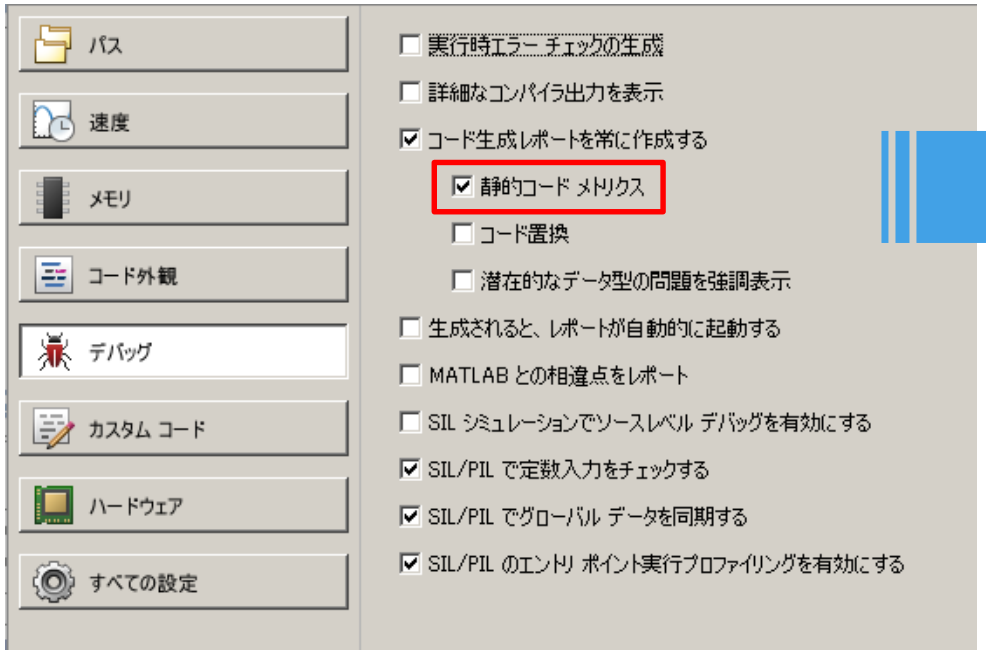


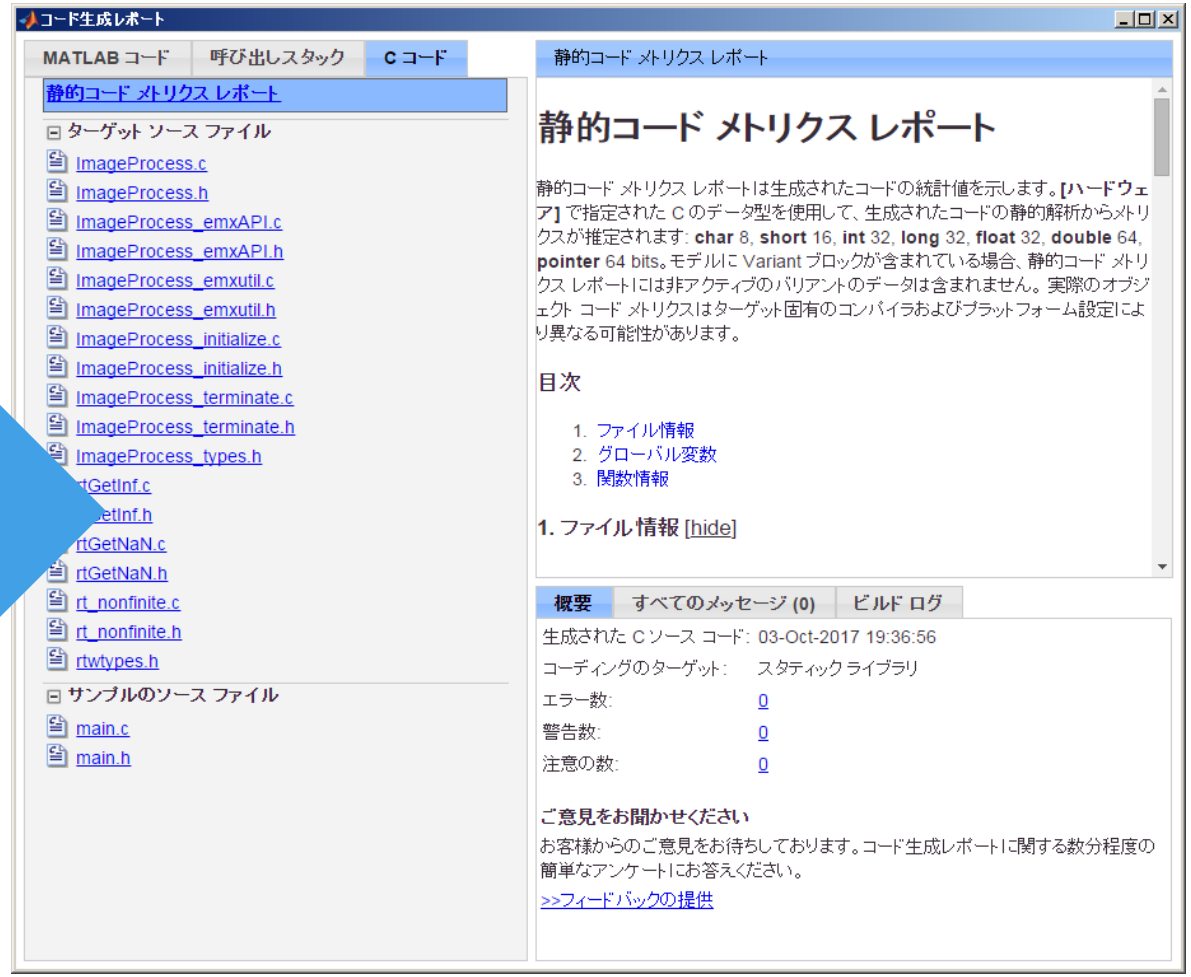
- 「可変サイズ･･･」有効化➔動的メモリ割り当てにより大きなデータセットではスタックオーバーフローの危険性、パフォーマンス低下
- 「再呼び出し･･･」を有効にすると、関数の引数によってデータ受取り、複数のプロセスから呼び出し可能



コード生成レポート/静的コードメトリクスの情報

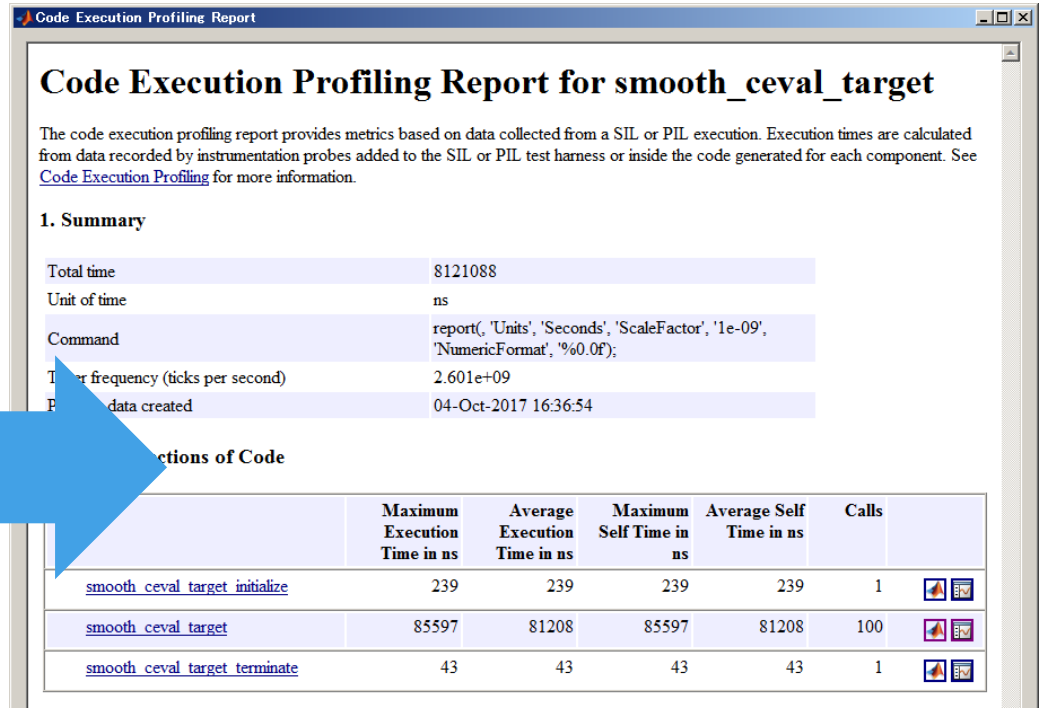
- ファイル情報
  - ファイルごとのコード行数
  - グローバル変数、サイズ（バイト）
  - スタックサイズ





パフォーマンスチェックと等価性検証
SIL/PIL検証（Embedded Coder機能）

- 等価性検証機能
  - SIL(Software In the Loop) : 生成Cコードを PC上で実行
  - PIL(Processor In the Loop) : 生成Cコードを組み込みプロセッサ/エミュレータで実行
- 生成コードのパフォーマンスチェック
  - 実行時間の測定

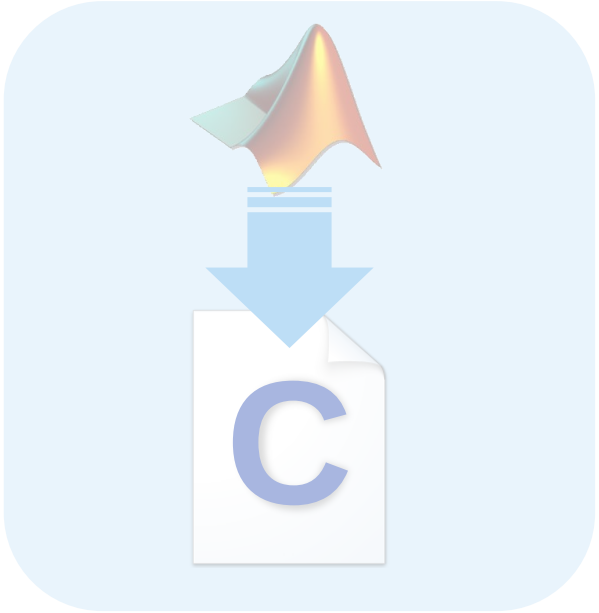


基本利用におけるTips

- %#codegenを付けることでエディターのコードアナライザーを有効化
- plot, disp, figureなどの表示系関数は無視されるので削除不要
- 非対応関数はcoder.extrinsic('func')を使ってコード生成無効化
  （コード生成準備段階で問題がある関数を除外したいときにも便利）
- 生成対象コードにassert命令を入れることで、データ型や行列サイズを指定
  例：
  - assert(isa(param,'single'));　　% パラメータはsingle型
  - assert(all(size(param) == [3, 4]));　　% 行列サイズは3x4
  - assert(isscalar(param));　　% スカラー

このセッションのポイント

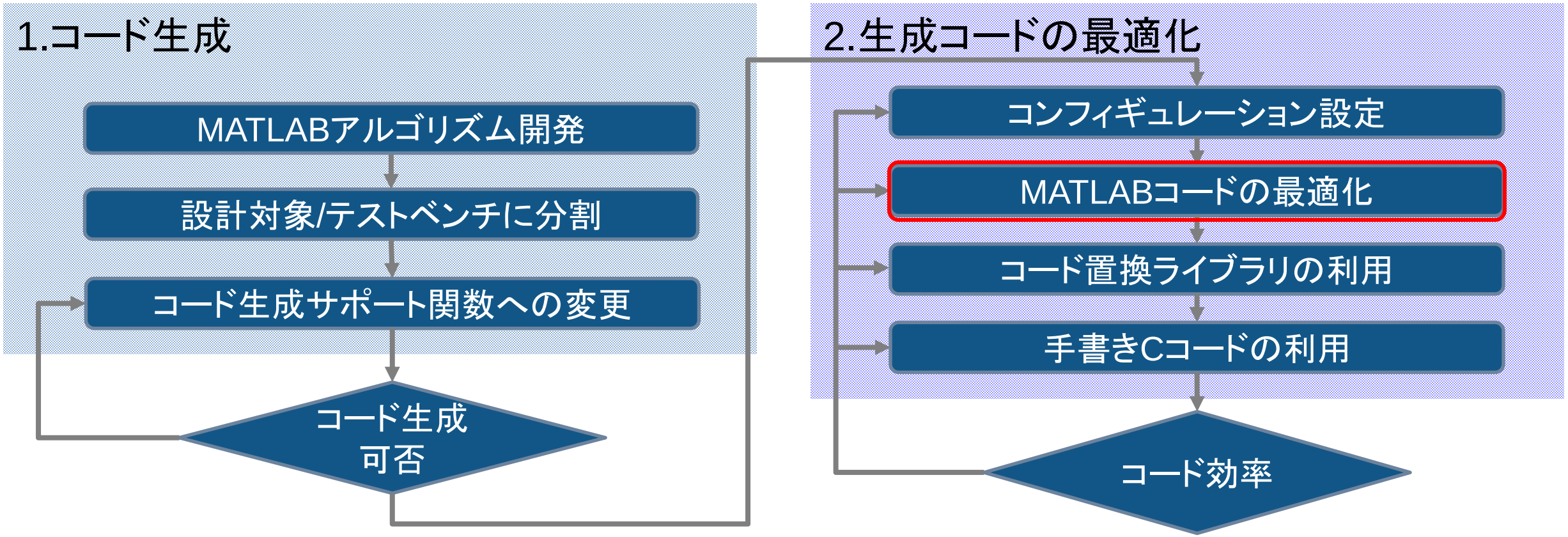
MATLAB Coderの
基本的な利用方法



効率的なC生成のための
コーディングテクニック

コード置換ライブラリと
System Object

MATLABプログラムからのCコード生成ワークフロー概要



なぜMATLABコードの最適化が必要なのか？

MATLABコードは
多様な入出力に対応する言語

```
function a= foo(b,c)
a = b * c;
```

- 要素ごとの乗算
- 内積演算
- 行列演算

整数/浮動小数点数
/固定小数点数
実数/複素数
…

```
void foo(const double b[15],
                const double c[30], double a[18])
{
  int i0, i1, i2;
for (i0 = 0; i0 < 3; i0++) {
    for (i1 = 0; i1 < 6; i1++) {
      a[i0 + 3 * i1] = 0.0;
      for (i2 = 0; i2 < 5; i2++) {
        a[i0 + 3 * i1] += b[i0 + 3 * i2] * c[i2 + 5 * i1];
      }
    }
  }
}
```

Cコードでは表現が異なる

```
double foo(double b, double c)
{
        return b*c;
}
```

MATLABコードの最適化
未実行分岐パスや変数の削減(coder.Constant)

- 未実行分岐パスの削減➔Cコード実行速度の向上
- 変数の削減➔メモリの削減

```
>> codegen MF -args {coder.Constant(1), coder.Constant(4), zeros(10,1)}
```

```
function out = MF(flag, gain, in)
%# codegen
switch flag
    case 1
        out=gain * sin(in);
    case 2
        out=gain * cos(in);
    otherwise
        out=gain * sqrt(in);
end
```

入力引数flag, gainが変数だと
生成コードでも変数として処理

入力引数flag, gainが定数だと
分岐削除、define定義

```
#define gain      (4.0)

void MF(const double in[10], double out[10])
{
  int k;
  for (k = 0; k < 10; k++) {
    out[k] = gain * sin(in[k]);
  }
}
```

MATLABコードの最適化
ループ内演算の最小化

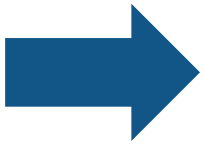
- ループ内の冗長演算の最小化➔Cコード実行速度の向上

改良前

```
function C=SeriesFunc(A,B,n)

C=zeros(size(A));

for i=1:n
    C=C+inv(B)*A^i*B;
end
```



改良後

```
function C=SeriesFunc(A,B,n)

C=zeros(size(A));

Bi = inv(B);

for i=1:n
    C = C+Bi*A^i*B;
end
```

ループ変数に依存しないinv(B)の処理をループ外に移動

MATLABコードの最適化
OpenMPを使ったマルチスレッドコード生成

- マルチスレッド化➔パフォーマンス向上

改良前

```
function a = test_for(r)    %#codegen
a=ones(10,256);

for i=1:10
    a(i,:)=real(fft(r(i,:)));
end
```

改良後

```
function a = test_for(r)    %#codegen
a=ones(10,256);

parfor i=1:10
    a(i,:)=real(fft(r(i,:)));
end
```

forを並列実行するparfor

```
void test_for(const double r[2560], double
a[2560])
{
  ....
#pragma omp parallel for ¥
 num_threads(omp_get_max_threads()) ¥
 private(i0) ¥
 firstprivate(dcv0,b_r)

  for (i = 0; i < 10; i++) {
  .....
}
```

OpenMPのプラグマ追加

※ parforはParallel Computing Toolboxの並列演算用のコマンド
※ OpenMPはC/C++並列コンピューティング用API

参考：シミュレーション用コードとコード生成用コードの切り替え

coder.targetオプションで、目的ごとに使用するMATLABコードを切り替え

```
function a = switch_fcn(r)    %#codegen
a=ones(10,256);

if coder.target('Rtw')
    % for code generation
    out = myCodeGenFcn(input);
else
    % for MATLAB and etc.
    out = mySimFcn(input);
end
```

Cコード生成時

MATLAB実行時など

オプション	ターゲット
MATLAB	MATLABでの実行
MEX	MEX関数生成時
Sfun	Simulinkシミュレーション
Rtw	LIB, DLL, EXE生成時
HDL	HDL生成時
Custom	カスタムターゲットの生成

MATLABコードの最適化
関数入力のデータコピーの削減

- 変数の再利用➔RAM使用量の削減、パフォーマンス向上

改良前

```
function y = foo( A, B )
%#codegen

y = A * B;

end
```

```
double foo(double A, double B)
{
  double y;
  y = A * B;
  return y;
}
```

改良後

```
function A = foo( A, B )
%#codegen

A = A * B;

end
```

```
void foo(double *A, double B)
{
      *A *= B;
}
```

参照渡し➔メモリ使用量の削減

MATLABコードの最適化
関数入力のデータコピーの削減(サブ関数使用時)

- 変数の再利用➔RAM使用量の削減、パフォーマンス向上
  ※入力がベクタ/行列のときのみ有効

改良前

```
function y1 = foo1(u1) %#codegen
    x1 = u1+1;
    y1 = foo2(x1);
end
function y2 = foo2(u2)
    x2 = u2.*2;
    y2 = [x2,x2];
end
```

```
void foo0(const double u1[5], double b_y1[10])
{
  int i;
  double x2;
for (i = 0; i < 5; i++) {
    x2 = (u1[i] + 1.0) * 2.0;
    b_y1[i] = x2;
    b_y1[5 + i] = x2;    ...
```

改良後

```
function y1 = fooOpt(u1) %#codegen
  u1 = u1+1;
  [y1, u1] = foo2(u1);
end
function [y2, u2] = foo2(u2)
    u2 = u2.*2;
    y2 = [u2,u2];
end
```

```
void fooOpt(double u1[5], double b_y1[10])
{
  int i0;
for (i0 = 0; i0 < 5; i0++) {
    u1[i0]++;
    u1[i0] *= 2.0;
    b_y1[i0] = u1[i0];
    b_y1[5 + i0] = u1[i0];    ...
```

中間変数x2削減

MATLABコードの最適化
coder.nullcopyによる変数初期化

- coder.nullcopyによる変数初期化➔不要な初期化の削減、パフォーマンス向上

改良前

```
function X = fcn(I) %#codegen
N = size(I);
X = zeros(N);
for i = 1:numel(N)
    if mod(i,2) == 0
        X(i) = i;
    elseif mod(i,2) == 1 ....
```

Xを初期化しないと代入できない

```
void fcn(const double I[64000], double X[64000])
{
  int i;
  (void)I;
  memset(&X[0], 0, 64000U * sizeof(double));
  for (i = 0; i < 2; i++) {
    if (b_mod(1.0 + (double)i) == 0.0) {
      X[i] = 1.0 + (double)i;
    } else { ....
```

改良後

```
function X = fcn(I) %#codegen
N = size(I);
X = coder.nullcopy(zeros(N));
for i = 1:numel(N)
    if mod(i,2) == 0
        X(i) = i;
    elseif mod(i,2) == 1 ....
```

```
void fnc(const double I[64000], double X[64000])
{
  int i;
  (void)I;
  for (i = 0; i < 2; i++) {
    if (b_mod(1.0 + (double)i) == 0.0) {
      X[i] = 1.0 + (double)i;
    } else {  ....
```

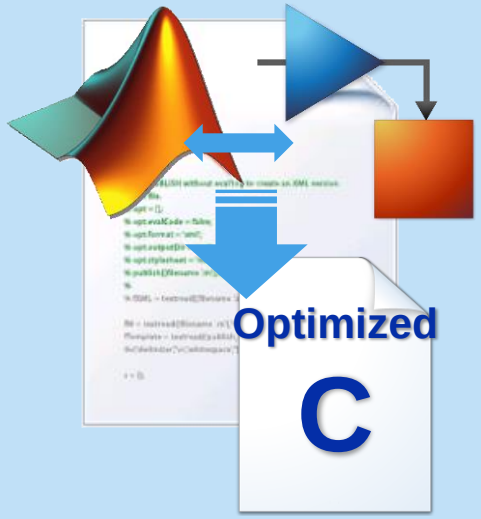
不要な初期化の削減

このセッションのポイント

MATLAB Coderの
基本的な利用方法

効率的なC生成のための
コーディングテクニック

コード置換ライブラリと
System Object



コード置換ライブラリ(Code Replacement Library=CRL)とは？

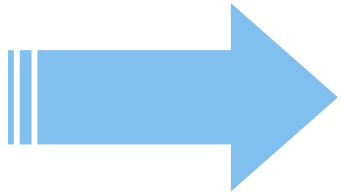
- ターゲットプロセッサ向けに最適化されたコードに置換機能 （Embedded Coderが必要）
- サポートプロセッサ：ARM Cortex-M/A, Intel x86(IPP), TI C55x/C62x/C64x/ C67xなど
- 四則演算、数学演算、信号処理演算などに対応（ターゲットプロセッサによって異なる）
  信号処理演算はSystem Objectが対応
- ユーザ登録が可能 **>> crtool**

MATLABプログラム

```
function [y1, y2, y3, y4

y1 = u1 + u2;
y2 = u1 - u2;
y3 = u1 .* u2;
y4 = sin(u3);
y5 = cos(u3);
y6 = sqrt(u3);
```

```
persistent h;
if isempty(h)
    h = dsp.FIRFilter('Numerator', fir1(63, 0.33));
end
y1 = step(h, u1);
```



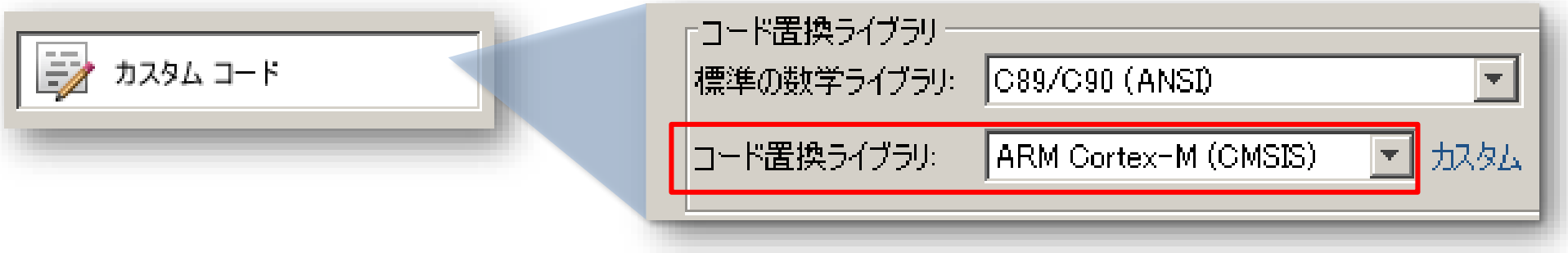
生成されたCortex-M用Cコード

```
void arm_EC_ops(const float u1[2], const floa
                float y2[2], float y3[2], flo
{
  mw_arm_add_f32(u1, u2, &b_y1[0], 2U);
  mw_arm_sub_f32(u1, u2, &y2[0], 2U);
  mw_arm_mult_f32(u1, u2, &y3[0], 2U);
  *y4 = arm_sin_f32(u3);
  *y5 = arm_cos_f32(u3);
  mw_arm_sqrt_f32(u3, y6);
```

```
/* System object Outputs function: dsp.FIRFilter */
arm_fir_f32(&b_obj->S, &U0[0], &b_y1[0], 75U);
```

コード置換ライブラリ適用方法

1. 演算（数式, System Object）、プロパティ、データ型、丸め条件を設定（置換条件はサポートパッケージの各ドキュメントに記載）
2. MATLAB Coderの詳細設定/カスタムコード

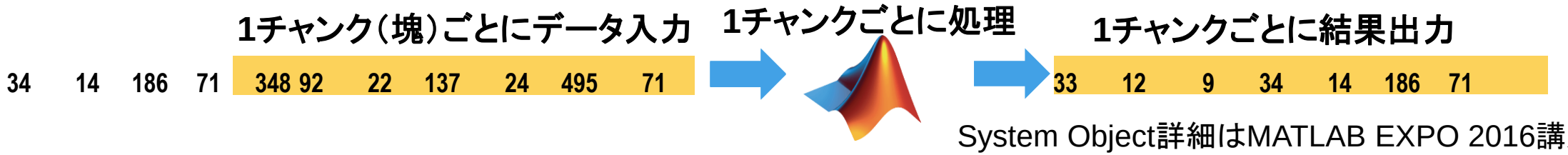


3. 置換の成功可否をレポートから確認

System Objectとは？

- 動的システムのストリーミング処理に特化したMATLABクラス
  - フィルタなどの遅延要素で必要な内部状態の保存に面倒な手続きが不要
- 共通のインターフェースを持つ（setup/reset/step/releaseメソッド）

	System Object	MATLAB関数
処理形態	ストリーミング	バッチ
メモリ消費	小	大
Simulink対応	ほぼ全て	一部
状態の管理	簡単	ユーザ設定
Cコード生成	ほぼ全て対応、コード置換ライブラリ	一部対応
HDLコード生成	高抽象度オブジェクト対応	関数非対応

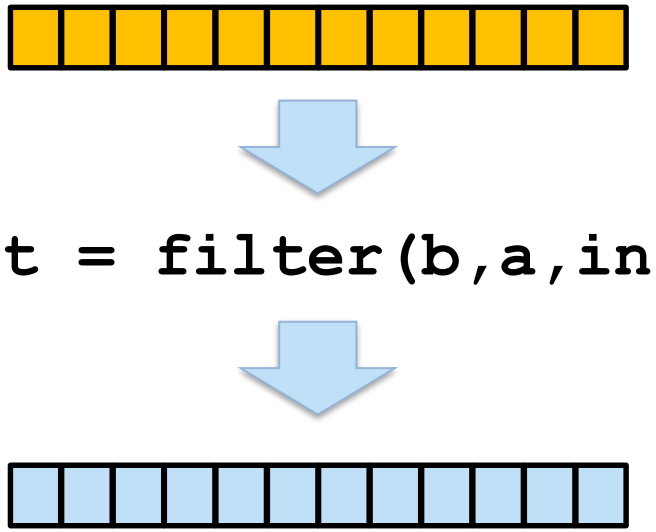


System Object詳細はMATLAB EXPO 2016講演資料参照
http://www.matlabexpo.com/jp/2016/proceedings/g3-system-object.pdf

MATLAB関数とSystem Object

MATLAB

フレームデータ
バッチ処理
コード

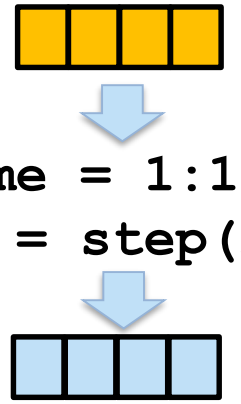


```
out = filter(b,a,in)
```

System Object

フレーム/サンプルデータ
ストリーミング処理
コード/ブロック

```
H = dsp.FIRFilter;
```



```
for time = 1:10
    out = step(H,in)
end
```

Simulink

フレーム/サンプルデータ
ストリーミング処理
ブロック

信号処理演算のコード置換ライブラリに対応するSystem Object
CMSIS/Ne10ライブラリサポート

System Object	Cortex-M CMSIS	Cortex-A Ne10
dsp.FIRFilter	✓	✓
dsp.FIRDecimator	✓	✓
dsp.FIRInterpolator	✓	✓
dsp.LMSFilter	✓	
dsp.BiquadFilter	✓	
dsp.FFT	✓	✓
dsp.IFFT	✓	✓
dsp.Convolver	✓	
dsp.Crosscorrelator	✓	
dsp.Mean	✓	
dsp.RMS	✓	
dsp.StandardDeviation	✓	
dsp.Variance	✓	

System Object	Cortex-M CMSIS	Cortex-A Ne10
dsp.VariableBandwidthFIRFilter	✓	✓
dsp.FIRHalfbandInterpolator	✓	✓
dsp.FIRHalfbandDecimator	✓	✓
dsp.CICCompensationDecimator	✓	✓
dsp.CICCompensationInterpolator	✓	✓
dsp.DigitalDownConverter	✓	✓
dsp.DigitalUpConverter	✓	✓
dsp.SampleRateConverter	✓	✓

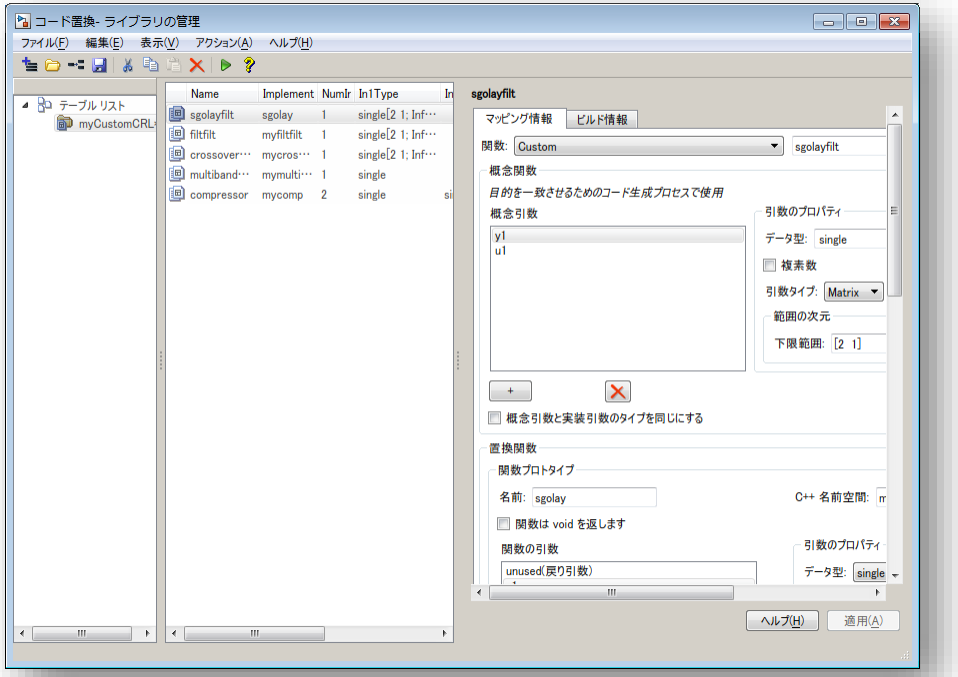
コード置換ライブラリー覧の表示

```
>> crviewer
```

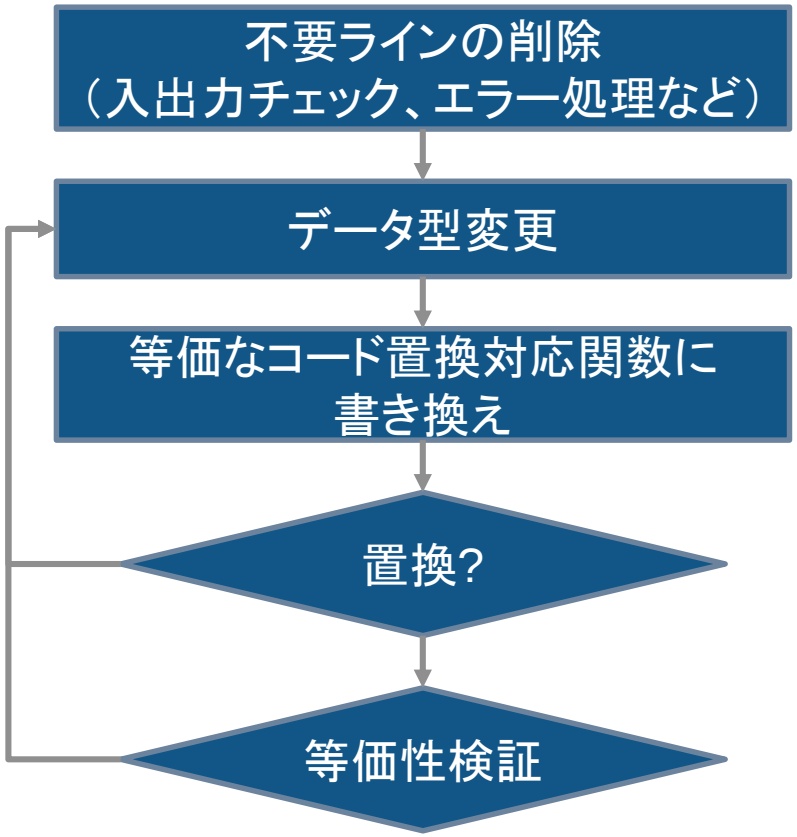
コード置換未対応の関数を対応させるには？

コード置換ライブラリのカスタマイズ
>> crtool

- MATLAB関数と対応するC関数をライブラリ登録
  - 入力/出力、データ型、次元の範囲など

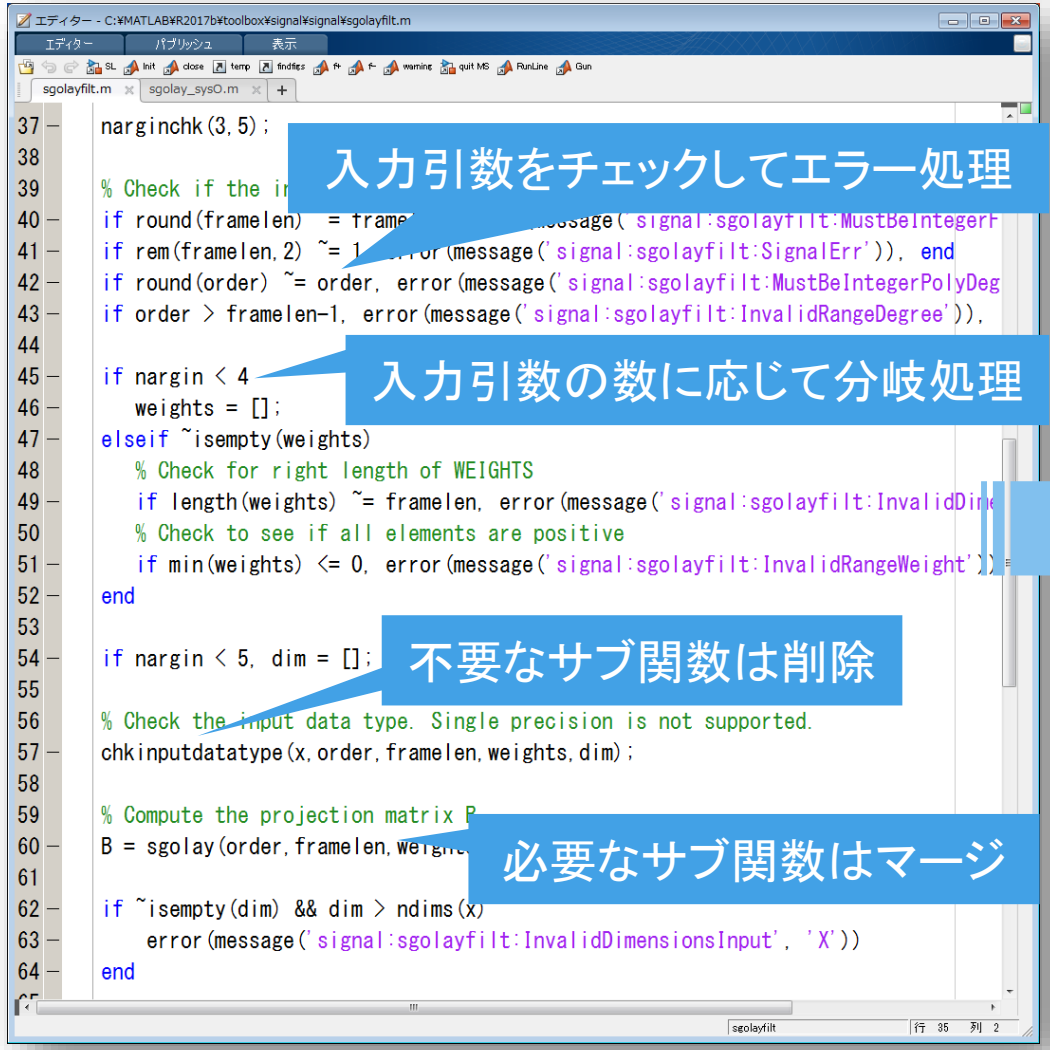


MATLAB関数をカスタマイズ

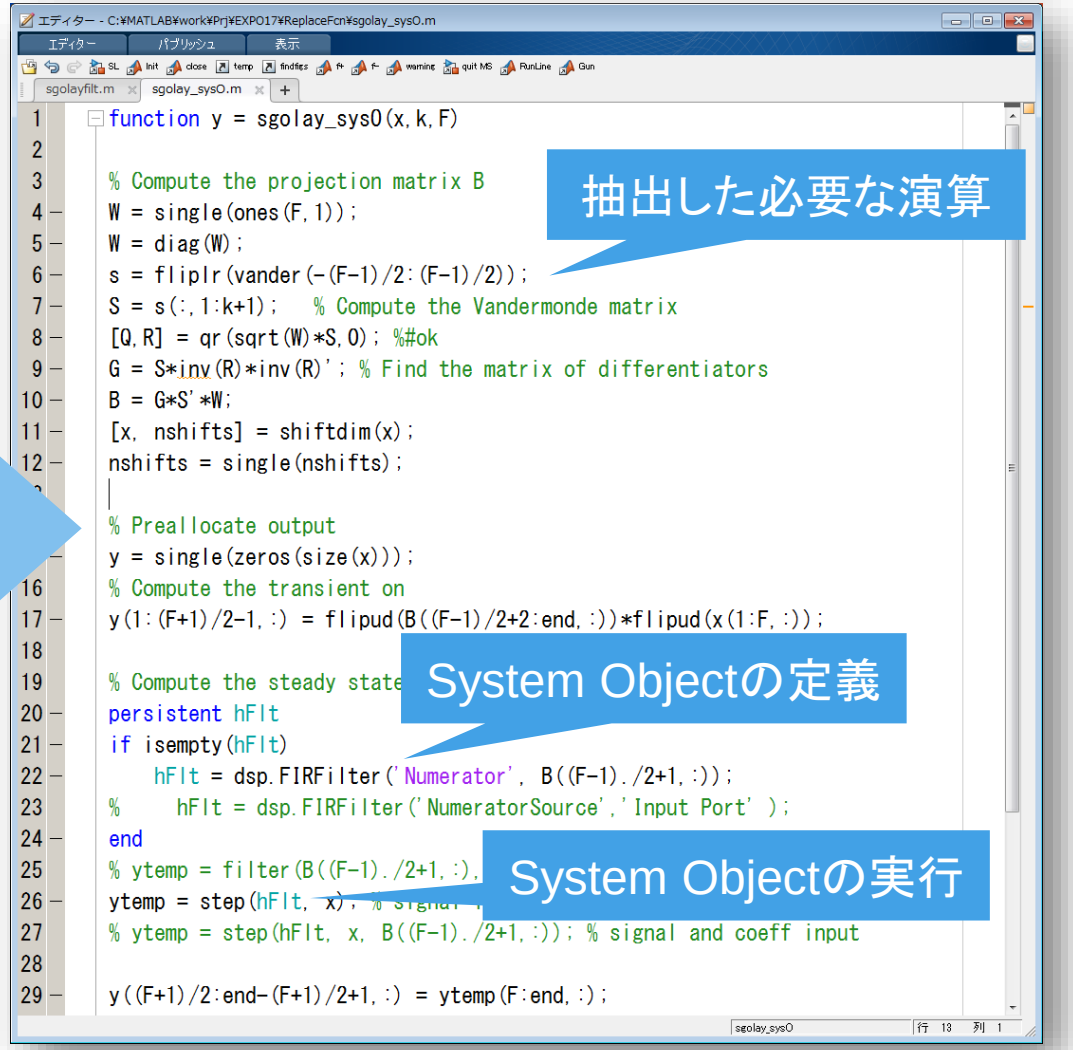


MATLAB関数のカスタマイズ例
Signal Processing Toolbox/sgolayfilt.m

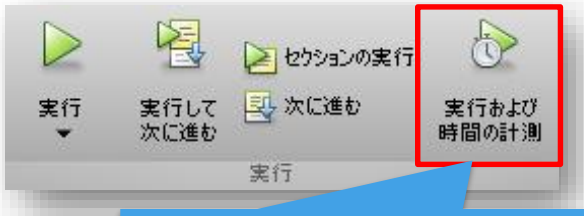
オリジナル（修正前）



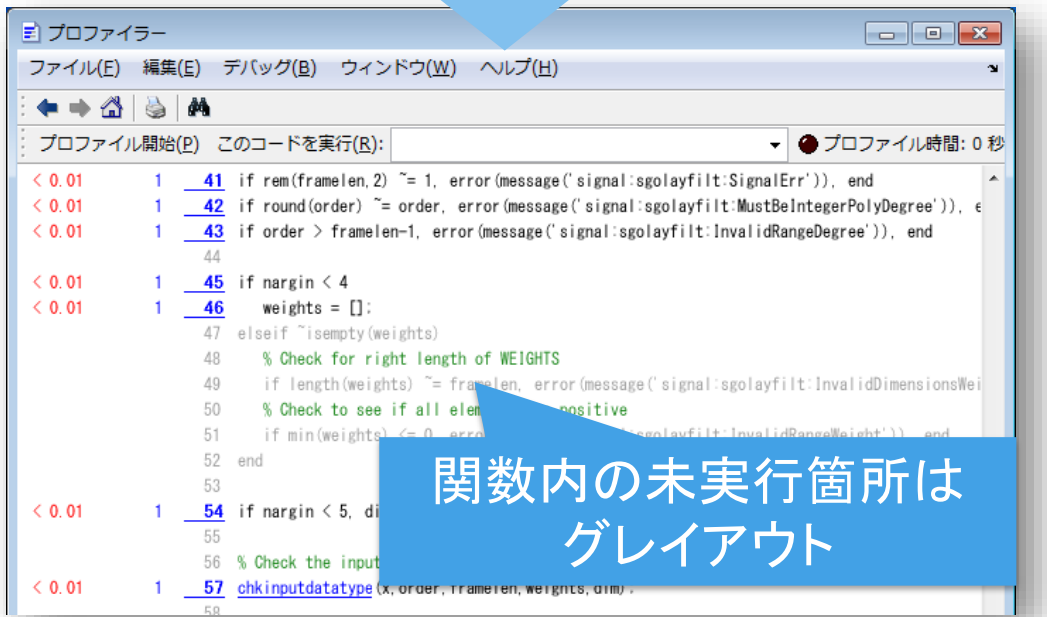
修正後



MATLAB関数の必要/不要箇所の判断に便利なツール

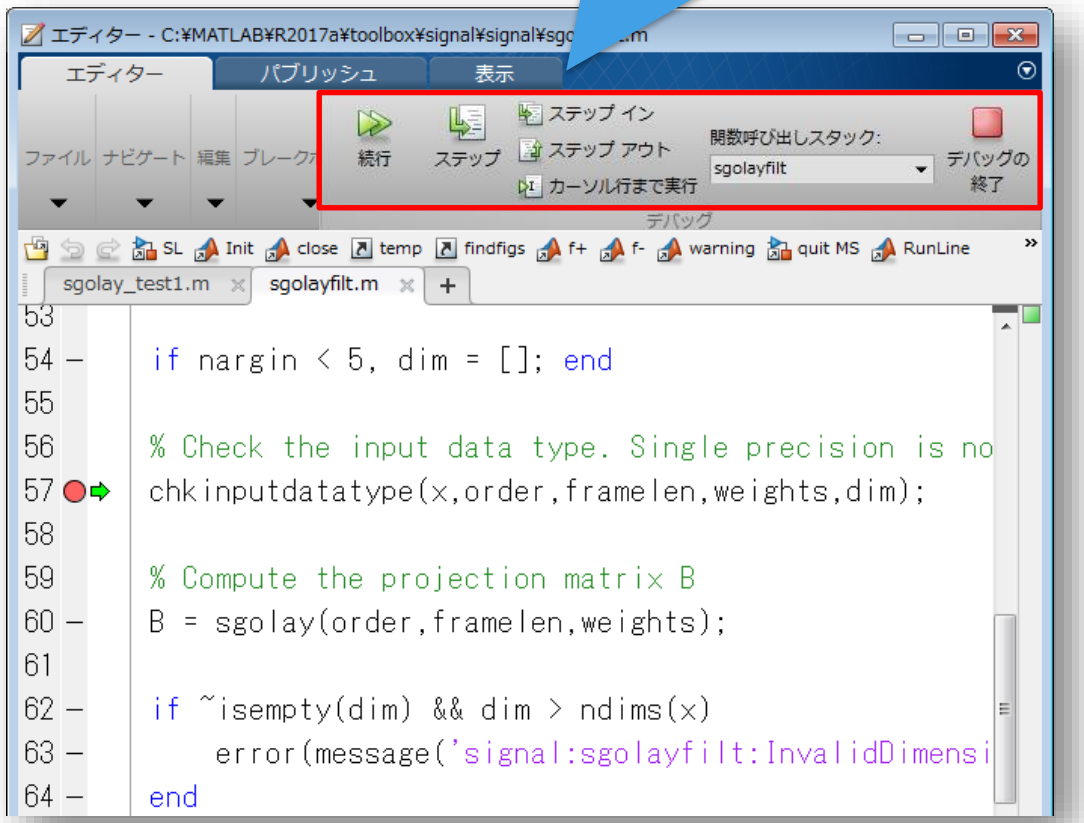


MATLABエディタ上の
[実行および時間の計測]をクリック
してプロファイルレポート生成



関数内の未実行箇所は
グレイアウト

デバッグ/ステップ実行により
不要箇所を判断



まとめ

MATLAB Coderの
基本的な利用方法

MATLABコードの
コーディングテクニック

コード置換ライブラリと
System Object

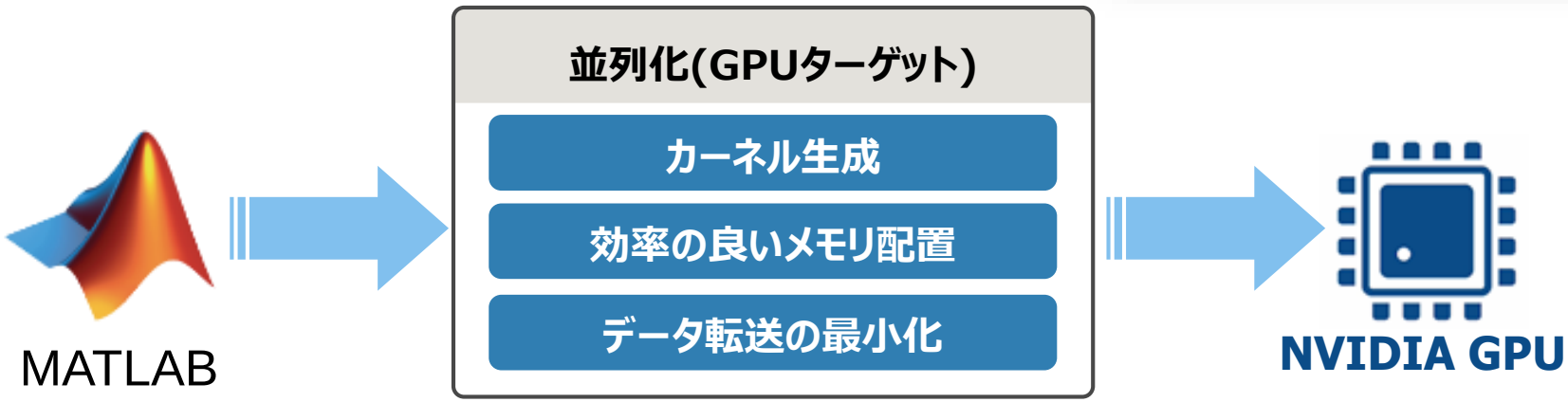
MATLAB Coder利用に役立つリソース

- Embedded Coderのオプション
  https://jp.mathworks.com/help/ecoder/gs/about-the-tutorials-1.html#buildlm
- クイックスタートガイド
  https://jp.mathworks.com/campaigns/products/offer/download_matlab-coder.html
- MATLABコードを固定小数点化するコツ（英語）
  https://jp.mathworks.com/company/newsletters/articles/best-practices-for-converting-matlab-code-to-fixed-point.html
- MATLAB and C/C++ Resources（英語）
  https://jp.mathworks.com/campaigns/portals/matlab-coder.html
- System Objectとは？
  https://jp.mathworks.com/help/matlab/matlab_prog/what-are-system-objects.html
- System Objectによるデバイスドライバブロックの作成（英語）
  https://jp.mathworks.com/help/supportpkg/raspberrypi/device-driver-blocks.html
- トレーニングコース：MATLAB CoderによるCコード生成
  https://jp.mathworks.com/training-schedule/matlab-to-c-with-matlab-coder.html

更なる高速化のために！　GPU Coder™

New in R2017b !!

- プラグマによる関数解析とカーネル生成
  - CUDAの文法を知らなくても利用できる
- 専用デザインパタンの利用も可能
  - より確実かつ効率の良いカーネル生成
- GPU Coder専用GUIを使ったコード生成
  - 初めてでも使いやすいGUI
- Simulinkへの統合
  - 生成したDLLを呼び出すAPIを作成可能



MATLABコードからCUDA Cを生成します