

# MATLAB®で信号処理 ～各種フィルタ設計を題材として～

MathWorks Japan

アプリケーションエンジニアリング部

竹本佳充

# Agenda

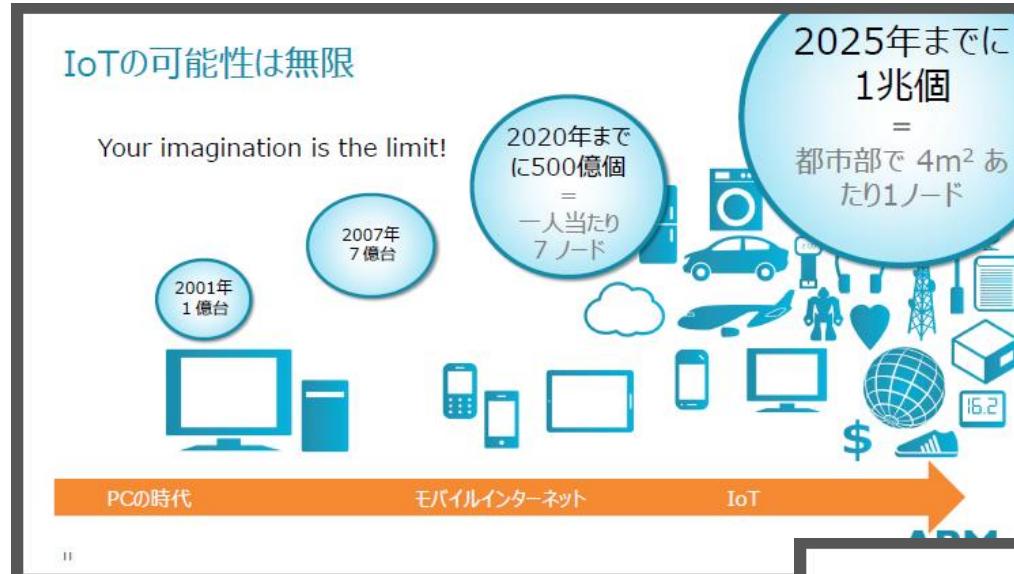
- **Section1:** フィルタとは？
- **Section2:** Case study
- **Section3:** フィルタの実装
- **Section4:** フィルタ設計FAQ
- まとめ

# Agenda

## ■ Section1: フィルタとは？

- 背景
- フィルタの用途・課題
- ユーザの声
- フィルタの種類
- MATLABによるフィルタ設計手法

# 背景: IoT時代のセンサ信号処理



- センサの組み込み製品への応用
- ノード数の飛躍的な増加
- 2020年東京オリンピック

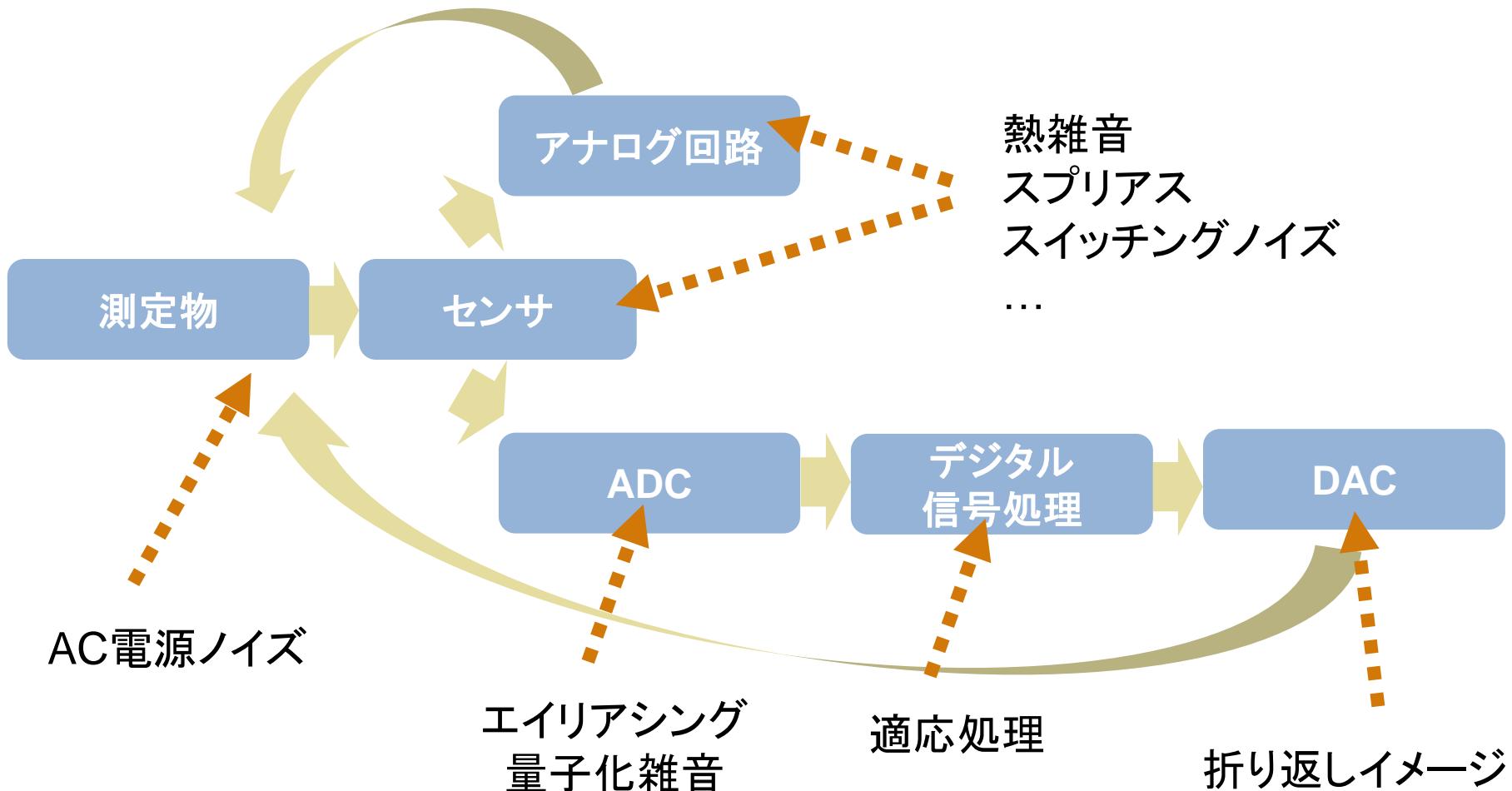


センサ信号処理は  
キー技術ノロジー

モデルベースで実践！組み込みDSP開発セミナー

(2014/9/10実施、ARM, STMicroelectronics, MathWorks三社共催セミナー) ARM社資料抜粋

# 背景: センサシステムにおけるフィルタリング



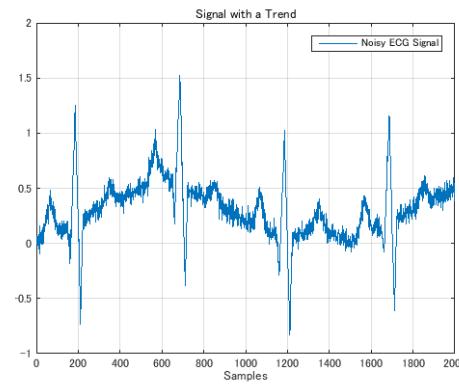
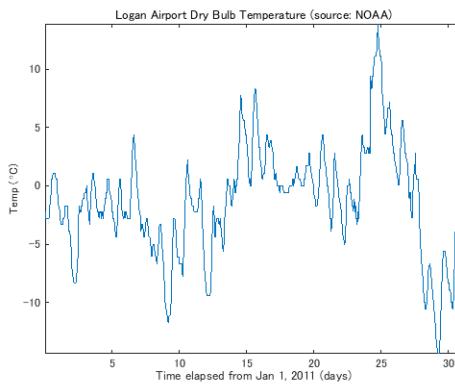
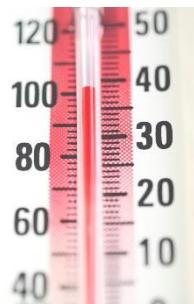
主にフィルタ処理を題材とします

# フィルタの用途・課題

トレンド成分

⇒トレンド成分のみ取得したい

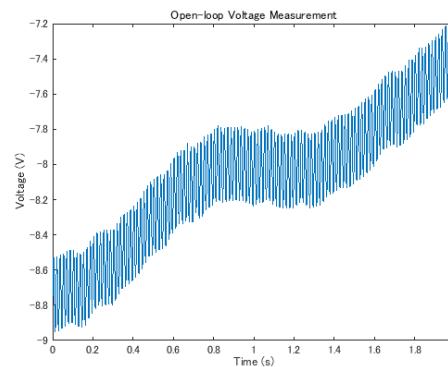
⇒トレンド成分は除去したい



電源ノイズ

取得データに重畳

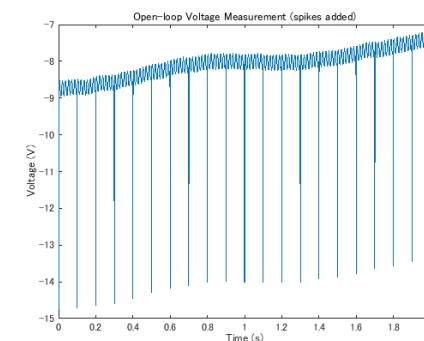
⇒信号処理で切るしかない



スパイクノイズ

通常のフィルタでは除去しきれない

⇒選択するフィルタに工夫が必要



# フィルタの用途・課題(cont'd)

- データ解析
  - 多様なフィルタ手法を検討・比較したい
  - 試行錯誤を減らして効率を上げたい
  - 既存技術はツールにまかせ、独自アルゴリズム検討に集中したい
  - 大規模データを扱いたい
- システム設計
  - テストベンチ作成に時間がかかる
  - 量子化の影響を検証したい
  - 安定性を評価・確保したい
- 実装
  - 次数が高すぎてH/Wに実装できない
  - シミュレータで計算した係数を他言語にマニュアルで移植している

すべてMATLAB/Simulinkで解決・実現可能

# ユーザーの声

関数ベース

設計手法

仕様ベース

オブジェクト  
ベース

FDATool

filterbuilder

スクリプト

設計環境

designfilt

## ユーザーの声(cont'd)

```
>> Hs = fdesign.lowpass
```

```
Hs =
```

```
    Response: 'Lowpass'  
    Specification: 'Fp,Fst,Ap,Ast'  
    Description: {4x1 cell}
```

```
NormalizedFrequency: true
```

```
    Fpass: 0.45  
    Fstop: 0.55      何の略称?  
    Apass: 1  
    Astop: 60
```

```
>> H = design(Hs)
```

```
H =
```

```
    FilterStructure: 'Direct-Form FIR'  
    Arithmetic: 'double'  
    Numerator: [1x43 double]
```

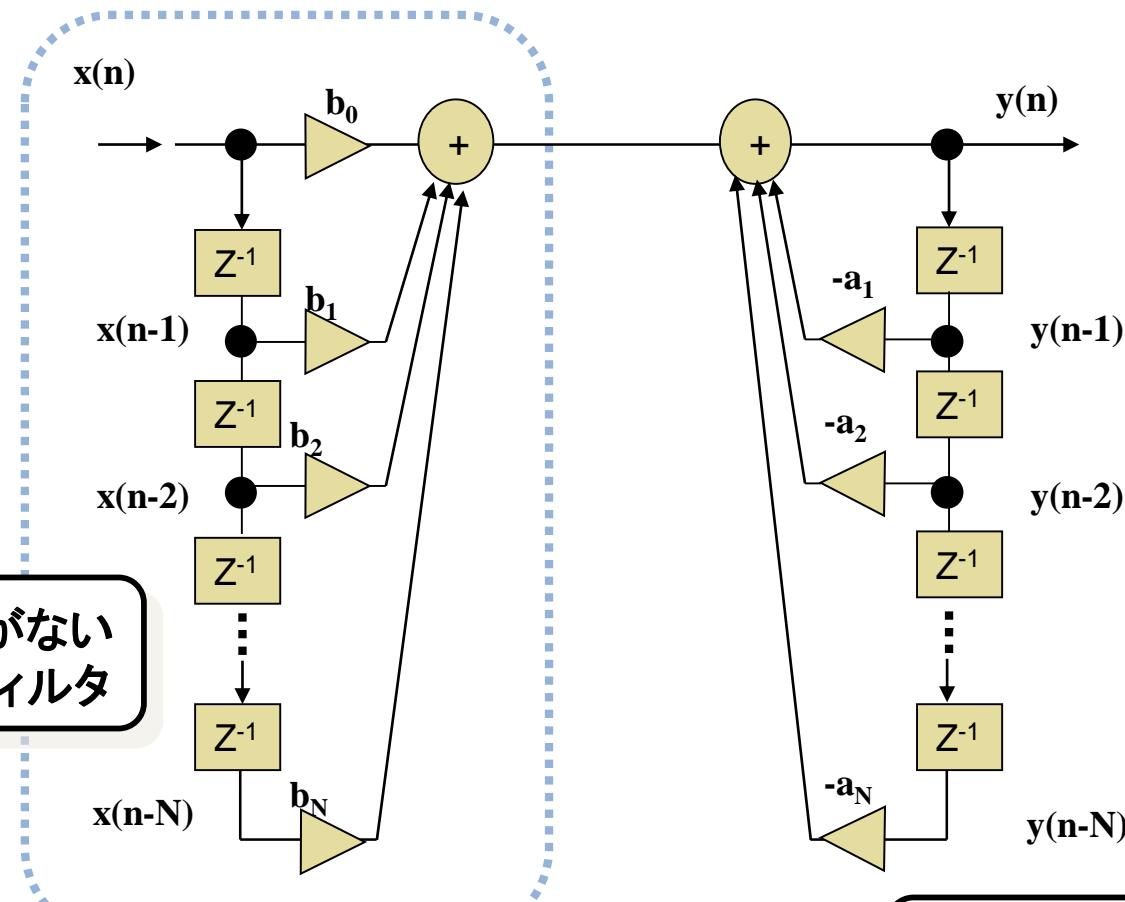
```
PersistentMemory: false
```

2種類のオブジェクトを  
管理する必要がある

```
>> SOS = H.sosMatrix;  
>> y = sosfilt(SOS, x);
```

フィルタの構造によって、  
適用する関数が違う

# MATLABによるフィルタ設計: フィルタの構造

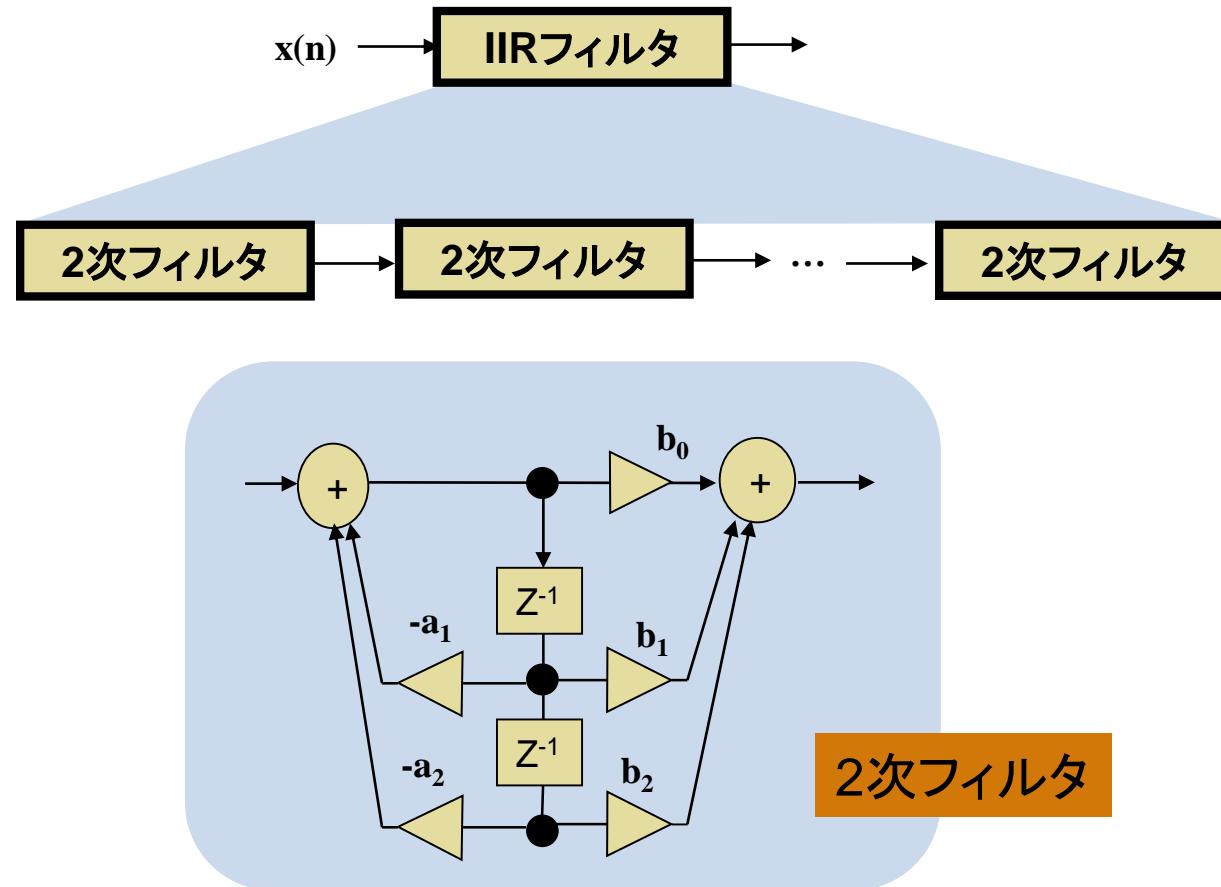


フィードバックがない  
タイプ⇒FIRフィルタ

フィードバックをもつ  
タイプ⇒IIRフィルタ

フィルタの設計⇒係数を決めるこ

# MATLABによるフィルタ設計:IIR(SOS型)



SOS (Second Order Section, Biquad)型  
⇒IIRフィルタの係数感度低減の効果

# FIRフィルタとIIRフィルタとの比較

項目	FIR (Finite Impulse Response)	IIR (Infinite Impulse Response)
インパルス応答	有限	無限
次数	多い	少ない
フィードバック構造	なし	あり
安定性	安定	不安定
群遅延特性	線形	非線形
H/Wコスト	高い	低い
速度	遅い	早い

# IIRフィルタの種類と特徴

項目	バタワース	チェビシェフⅠ型	チェビシェフⅡ型	楕円
通過域リップル	なし	あり	なし	あり
遮断域リップル	なし	なし	あり	あり
群遅延特性	○	△	△	×
特徴	リップル特性を持たないため、 <b>平坦な周波数特性</b> が得られる	通過域リップルを大きくすることで、急峻なロールオフ特性が得られる	遮断域の減衰量を小さく指定することで、急峻なロールオフ特性が得られる	通過域と遮断域双方のリップルを制御可能ため、 <b>最も急峻なロールオフ特性</b> が得られる
波形イメージ				

# MATLABによるフィルタ設計: フィルタの実行

出力                    フィルタ係数                    入力

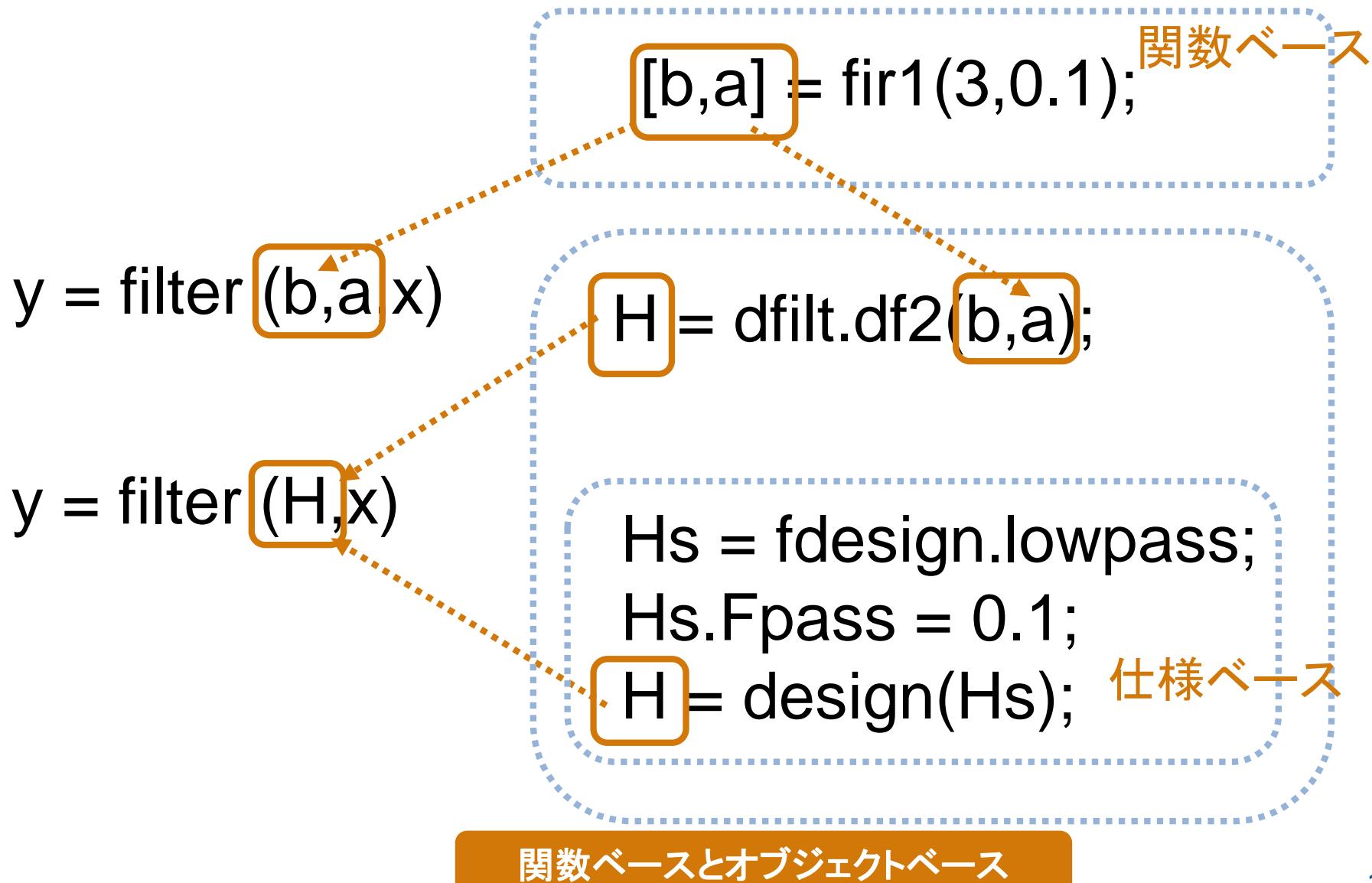
$y = \text{filter}(b, a, x)$

$y = \text{filter}(H, x)$

フィルタオブジェクト

filter関数でフィルタリング

# MATLABによるフィルタ設計: 設計



# MATLABによるフィルタ設計: 手法

## ■ 関数ベース

- 従来手法
- 記述がシンプル
- 直感的

## ■ オブジェクトベース、仕様ベース

- フィルタ属性の管理
- メソッド選択の試行錯誤削減
- 仕様オブジェクト作成、フィルタオブジェクト作成の2ステップが必要

## ■ `designfilt`

- R2014a新機能
- 仕様ベースの設計手法の手順を簡略化  
(オブジェクト生成作業を1ステップ化、プロパティ名の明確化)
- フィルタ設計フローのアシスト(足りない引数の候補推定等)

# MATLABによるフィルタ設計: オプション

信号処理用  
Simulinkブロックライブラリ

- FFT
- 各種フィルタ
- スペクトラムアナライザ
- ライブ音声処理



フィルタ設計用  
MATLAB関数ライブラリ

- 適応フィルタ
- マルチレートフィルタ
- 固定小数点フィルタ

R2011aより統合

- Filter Design Toolboxの  
フィルタ設計機能を踏襲
- 旧Signal Processing Blocksetの  
機能を、MATLAB環境で実行可能

DSP System Toolboxに  
各種応用的フィルタ設計機能

# MATLABによるフィルタ設計: オプション構成

## ■ MATLAB

- filter処理
- 多項式フィッティング(トレンド抽出)

## ■ Signal Processing Toolbox

- ベーシックなフィルタ設計
  - FIR, IIR(バタワース、チェビシェフ...)
  - メディアンフィルタ
- フィルタ設計GUI(FDATool, filterbuilder)

## ■ DSP System Toolbox

- 応用的なフィルタ設計
  - 適応フィルタ
  - マルチレートフィルタ
- 各種可視化(スペクトラムアナライザ)
- 音声信号処理

## ■ Simulink

- ブロック線図環境

## ■ Embedded Coder

- 組み込みコード生成

# Agenda

- Section1: フィルタとは？
- Section2: Case study
- Section3: フィルタの実装
- Section4: フィルタ設計FAQ
- まとめ

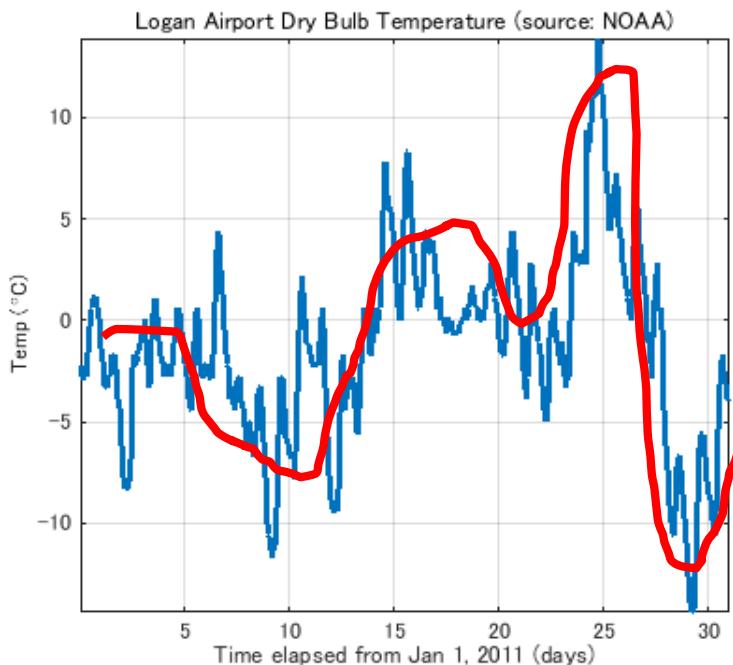
# Agenda

## ■ Section2: Case study

- Case1: 気温データ
  - (トレンド抽出、アベレージング)
- Case2: 心拍データ
  - (トレンド成分の除去、ピーク値検索、平滑化)
- Case3: センサーデータ
  - (リサンプリング、メディアンフィルタ)
- Case4: 変位データ・加速度データ
  - (微分・積分処理)
- Case5: ノイズキャンセリング
  - (適応フィルタ)
- Case6: ΔΣADコンバータ
  - (アンチエイリアシング、マルチレート処理)

# Case1: 気温データ

2011年1月のボストンの気温データ



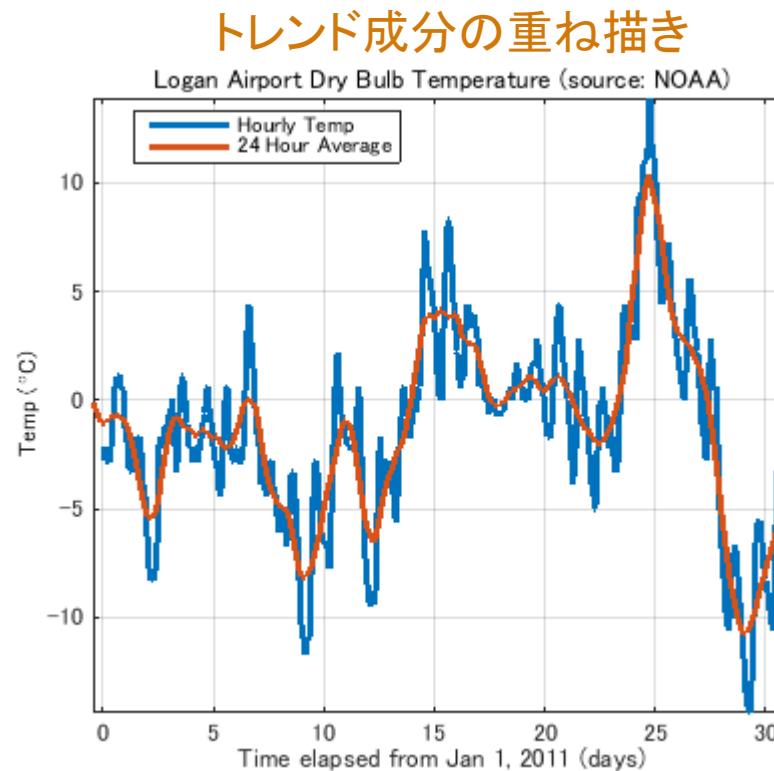
目的:

- 1ヶ月の気温変動傾向
- 24時間の気温変動傾向

課題:

- 時刻の影響を除去
- 24時間の変動平均

# Case1: 気温データ(トレンド抽出)

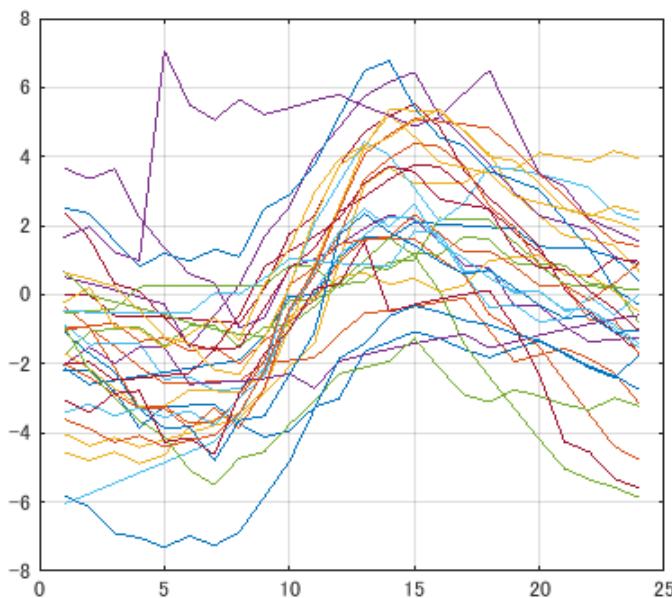


```
>>coeff = ones(1, 24)/24; フィルタ係数  
>>out = filter(coeff, 1, in); フィルタリング
```

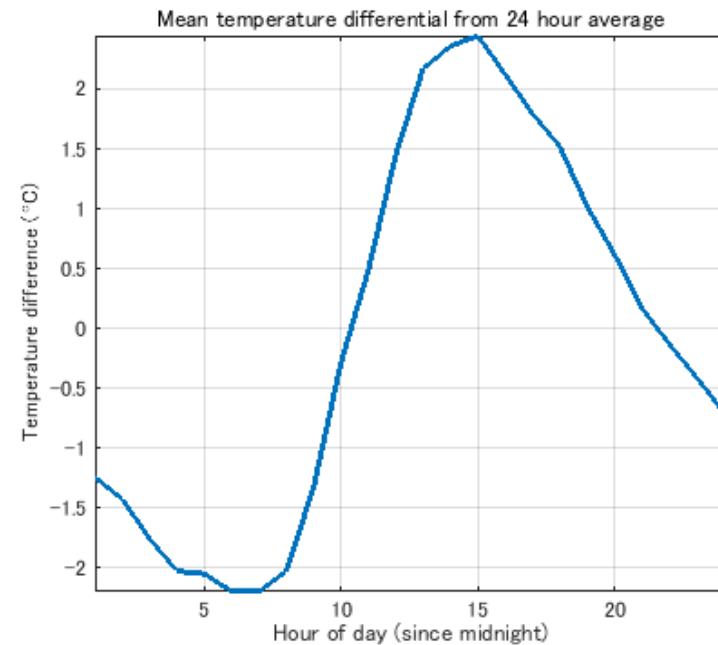
移動平均フィルタでスムージング

# Case1: 気温データ(アベレージング)

24時間データ重ね描き(31日分)



24時間データ(31日分平均)

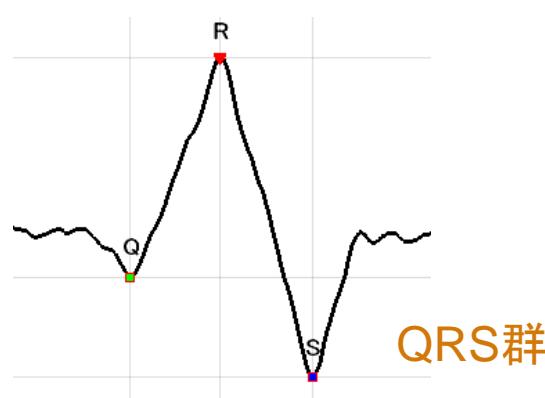
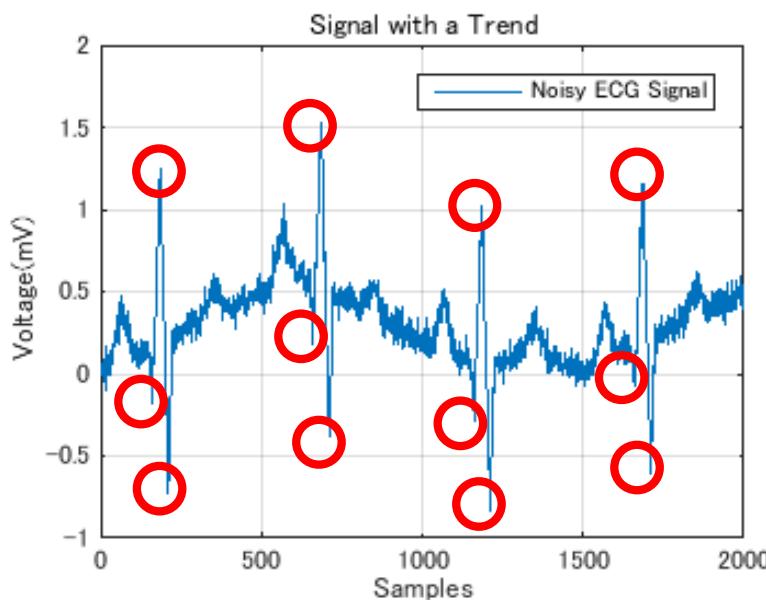


```
>>out = reshape(in, 24, 31).';    % 24行31列に並べ替え  
>>plot(1:24, mean(out))          % 全行の平均値を求める
```

行列処理で記述は簡潔に

# Case2: 心電図データ

## 心電図データ



### 目的:

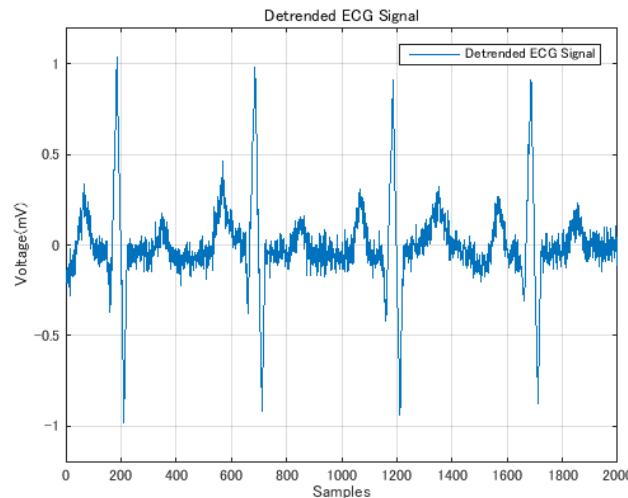
- Q波、R波、S波を得る
- 以下の情報を得る
  - 立ち上がり時間
  - 立ち下がり時間
  - 立ち上がり振幅
  - 立ち下がり振幅

### 課題:

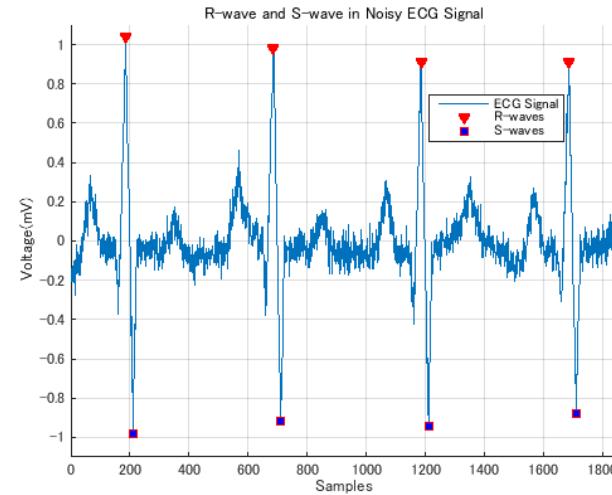
- 信号劣化要因の除去
- ピーク情報の取得方法

# Case2: 心電図データ(トレンド除去、ピーク値探索)

トレンド除去後の心電図データ



R波とS波  
(Q波は雑音との識別困難)



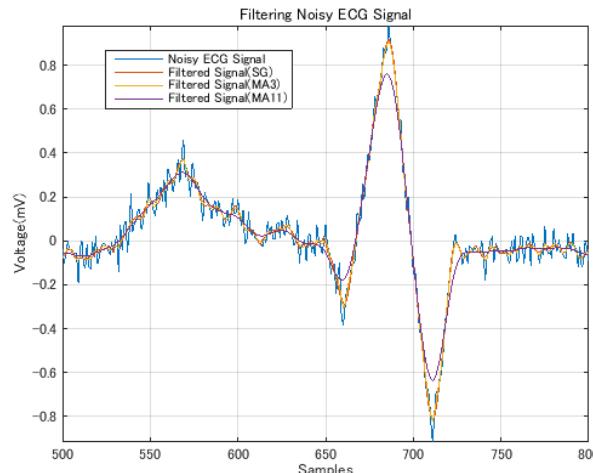
```
>> [p, s, mu] = polyfit((t, in, 6);
>> f_y = polyval(p, t, [], mu);
>> out = in - f_y;
```

```
>> [~, R] = findpeaks(in, ...
'MinPeakHeight', 0.5);
```

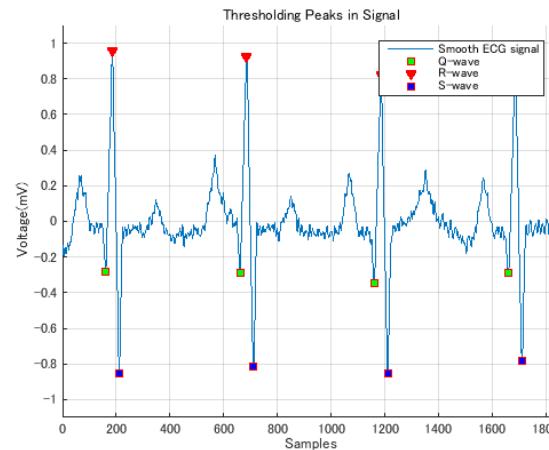
多項式近似でトレンド除去  
findpeaks関数でピーク値探索

# Case2: 心電図データ(平滑化とピーク値再探索)

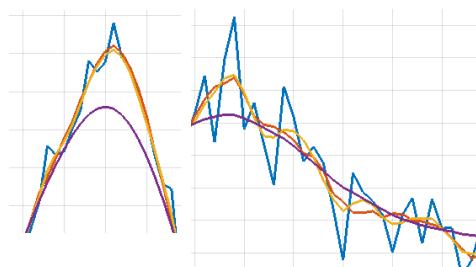
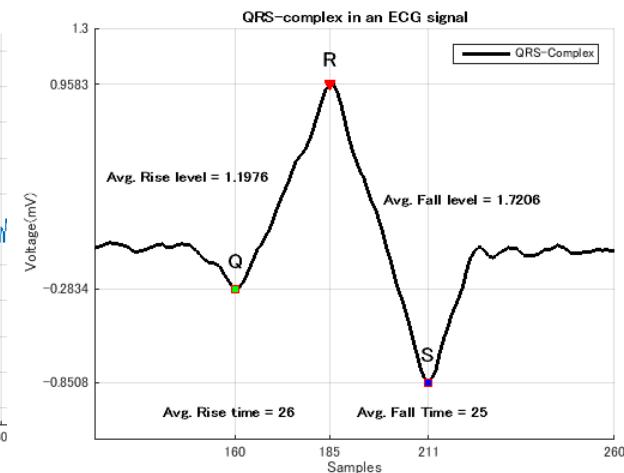
Savitzky-Golay法および  
MA法による平滑化



QRS群



QRS群各種統計値



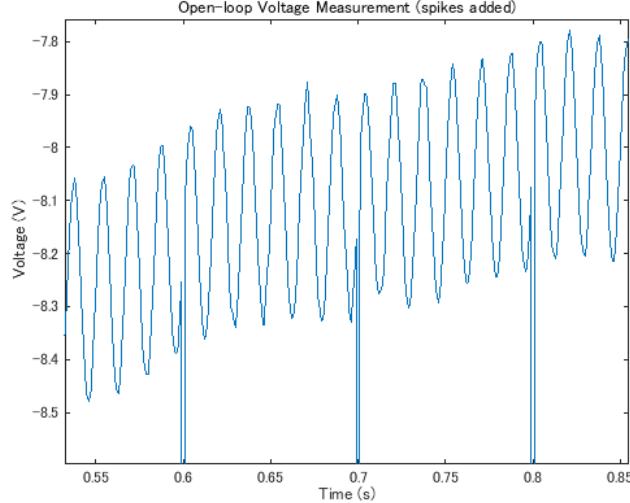
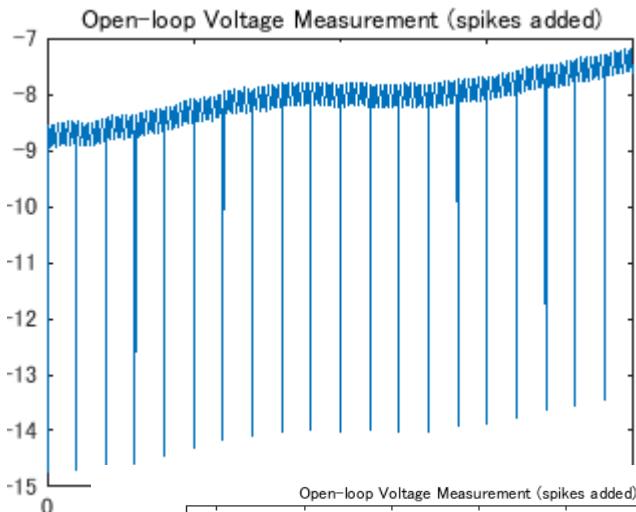
`>>out = sgolayfilt(in, 7, 21);`

- MA次数大: 振幅減衰
- MA次数小: 雜音残る

目的に適した手法選択

# Case3: センサーデータ

アナログセンサー電圧データ  
(ACノイズ、スパイクノイズ)



目的:

- 取得したセンサー信号から不要成分を除去

課題:

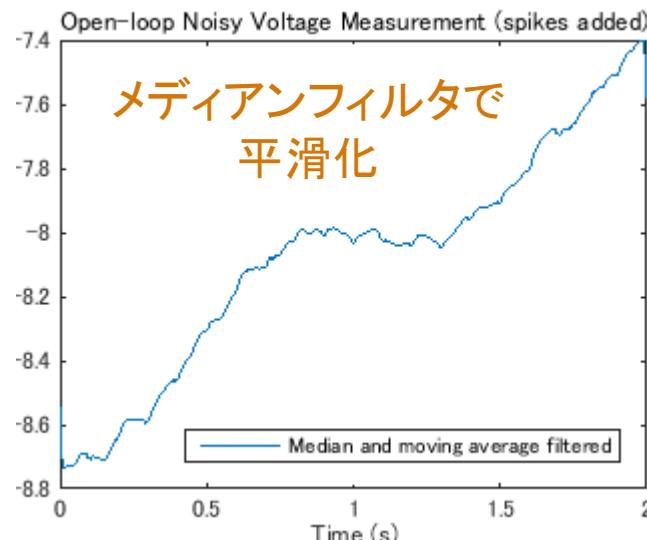
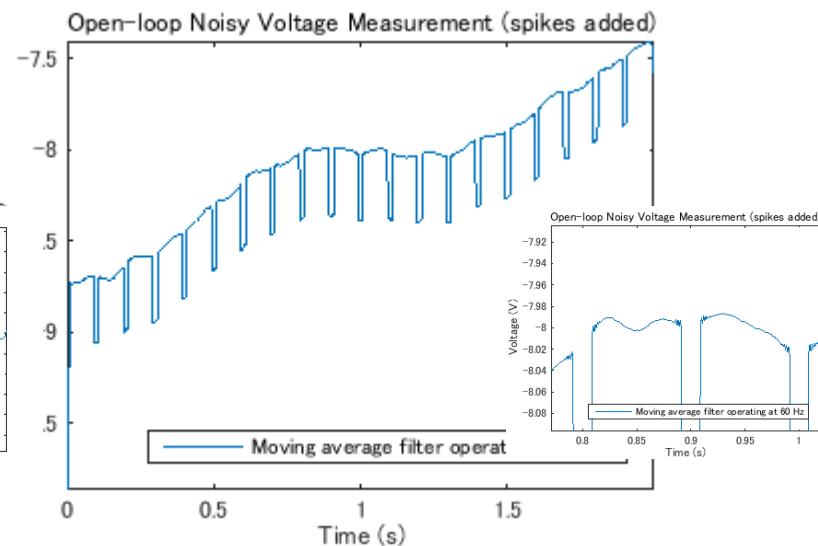
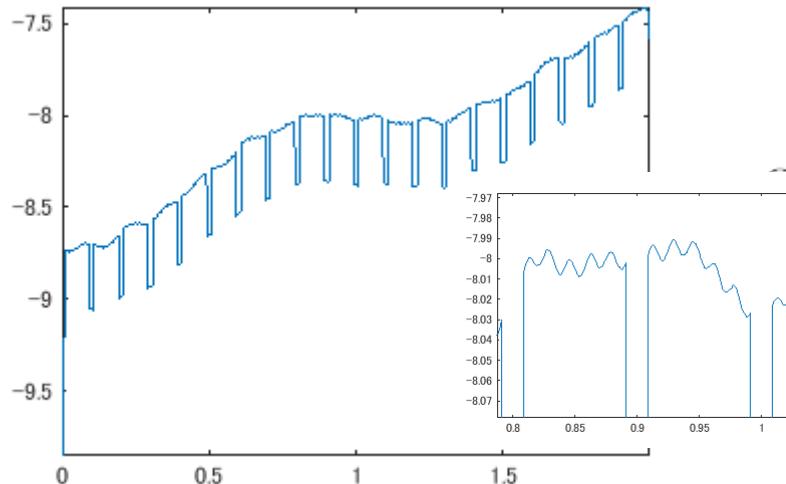
- ACラインからのノイズ除去
- スパイクノイズ除去

# Case3: センサーデータ(メディアンフィルタ)

リサンプリング後に

Savitzky-Golay法で平滑化

Savitzky-Golay法で平滑化

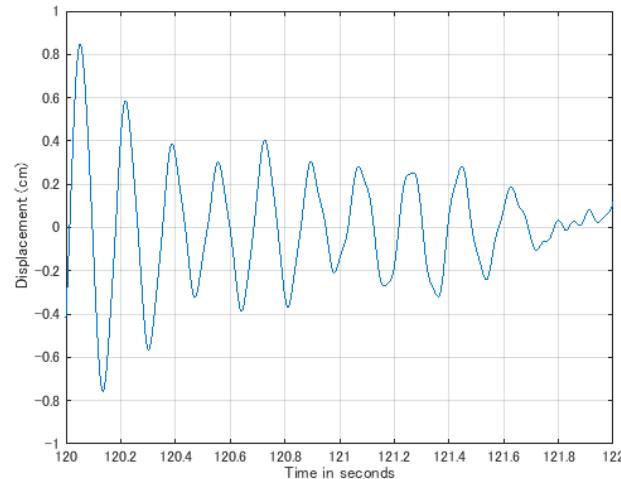


```
>>out1 = resample(in1, fs1, fs2);
>>out2 = medfilt1(in2, 3);
```

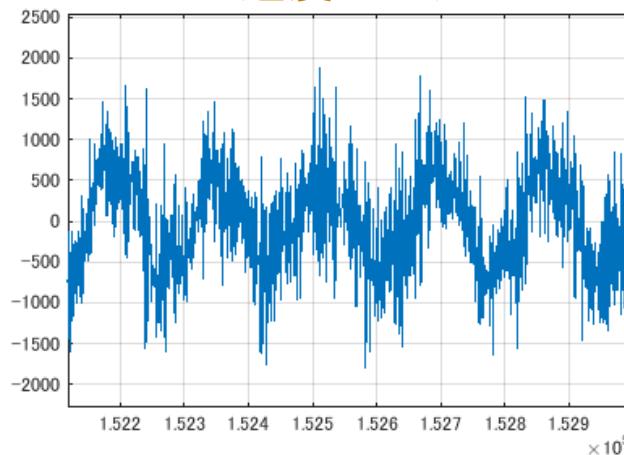
スパイクノイズには  
メディアンフィルタが効果的

# Case4: 変位データ、加速度データ

地震の条件下における  
3階建て構造物の1階変位データ



加速度データ



目的:

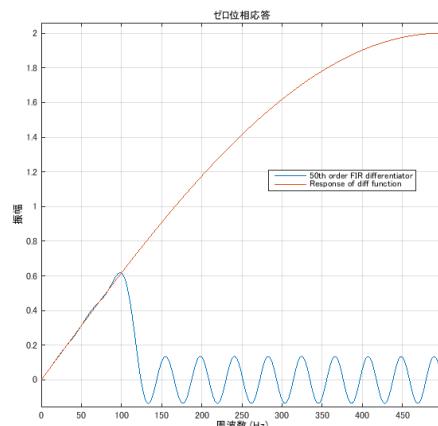
- 変位センサーから得られた床の位置データから、速度と加速度情報を取得
- 加速度データから、速度と変位情報を取得

課題:

- フィルタによる微分処理
- フィルタによる積分処理

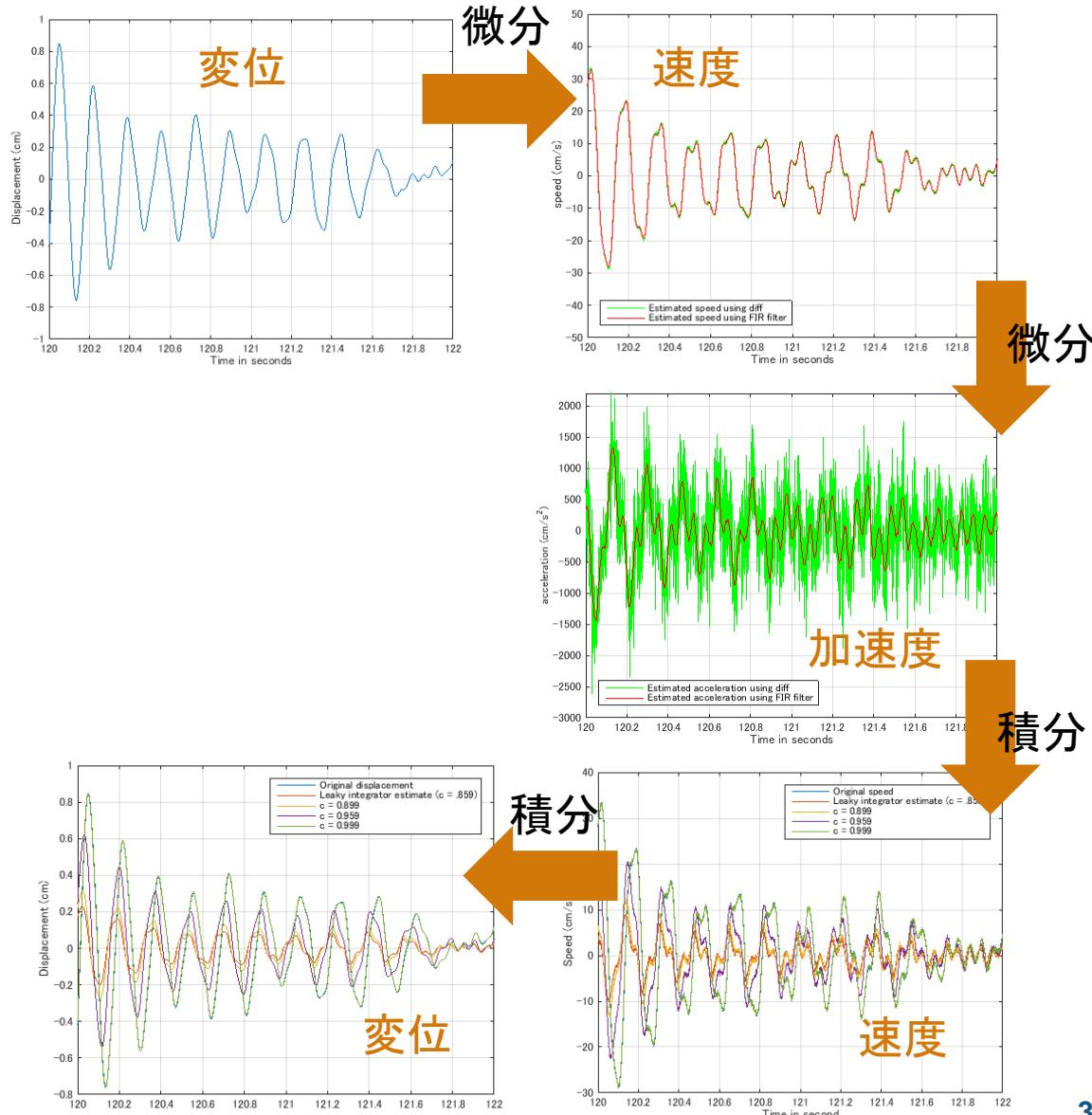
# Case4: 変位データ、加速度データ(微分・積分フィルタ)

振幅応答比較  
(diff関数vs微分フィルタ)



diff関数の伝達関数  
 $H(z) = 1 - z^{-1}$   
 ⇒高周波成分を強調  
 ⇒微分フィルタが効果的

フィルタにより  
微分・積分が可能

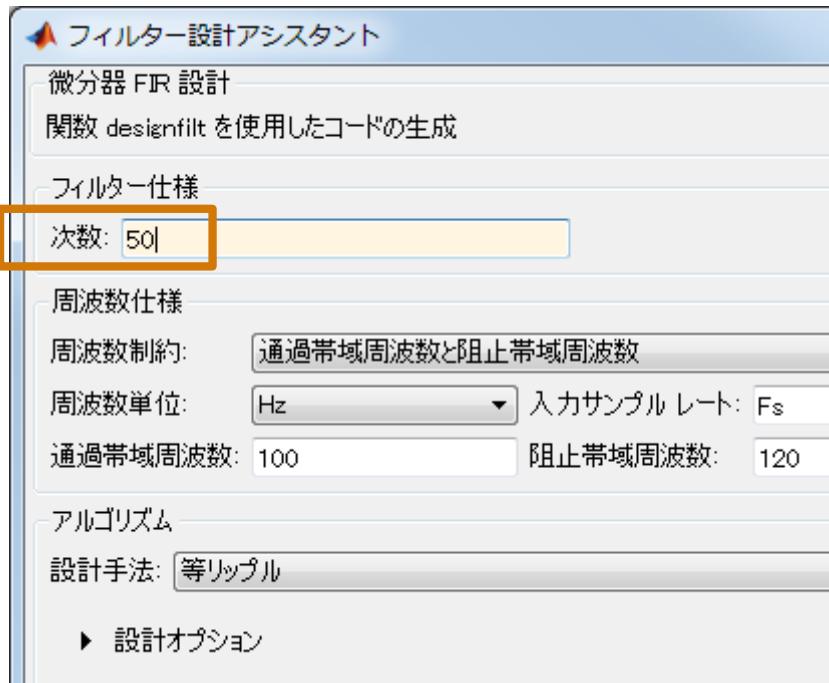


# Case4: 変位データ、加速度データ(cont'd) R2014a

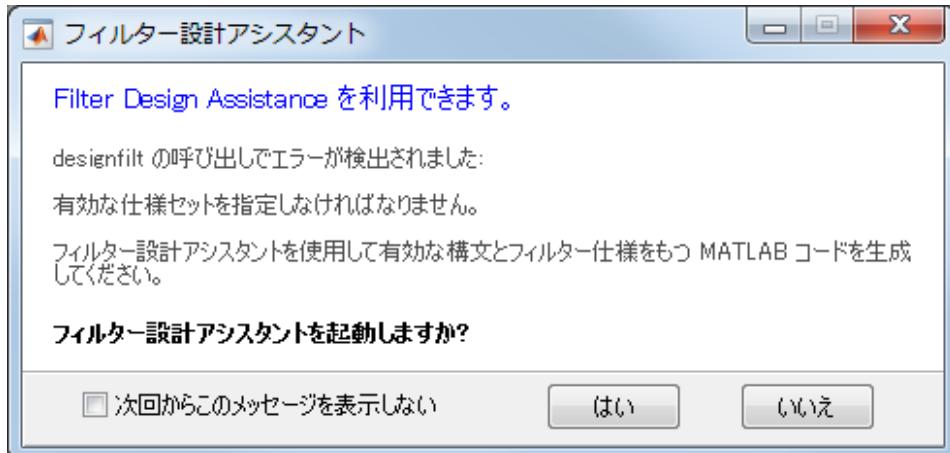
## ① フィルタオブジェクト定義

```
>> df = designfilt('differentiatorfir',...
    'PassbandFrequency',100,...
    'StopbandFrequency',120,...
    'SampleRate',Fs);
```

## ③ 不足情報の補完を促すGUI



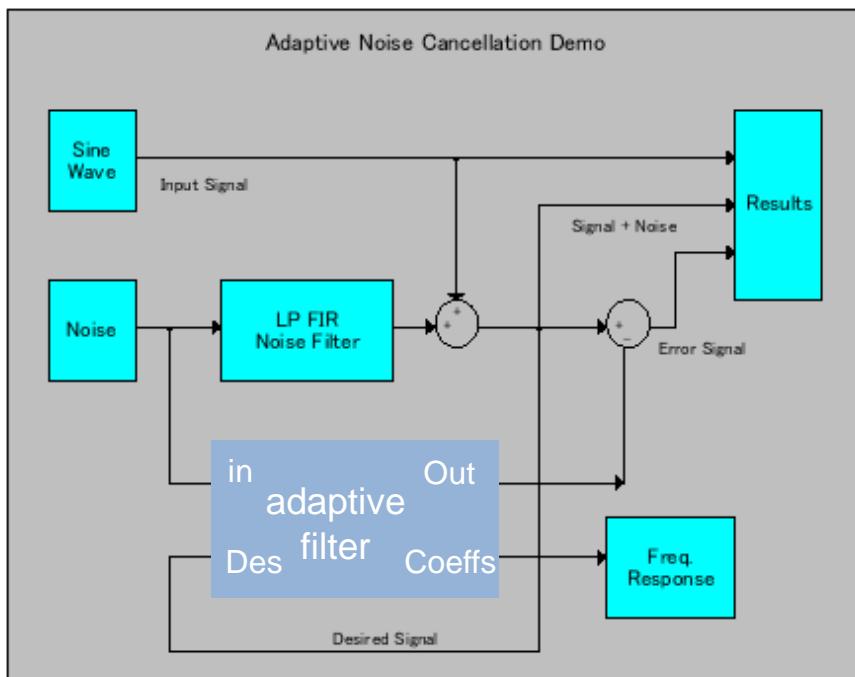
## ② 設計に必要な要求仕様が不足



## ④ 必要な仕様条件を補完し、自動実行

```
df = designfilt('differentiatorfir',...
    'FilterOrder', 50,...
    'PassbandFrequency', 100,...
    'StopbandFrequency', 120,...
    'SampleRate', Fs);
```

# Case5: ノイズキャンセリング



目的:

- 雑音が通過した経路を推定
- 推定した経路情報を用いて、雑音の重畠した信号から雑音のみ除去

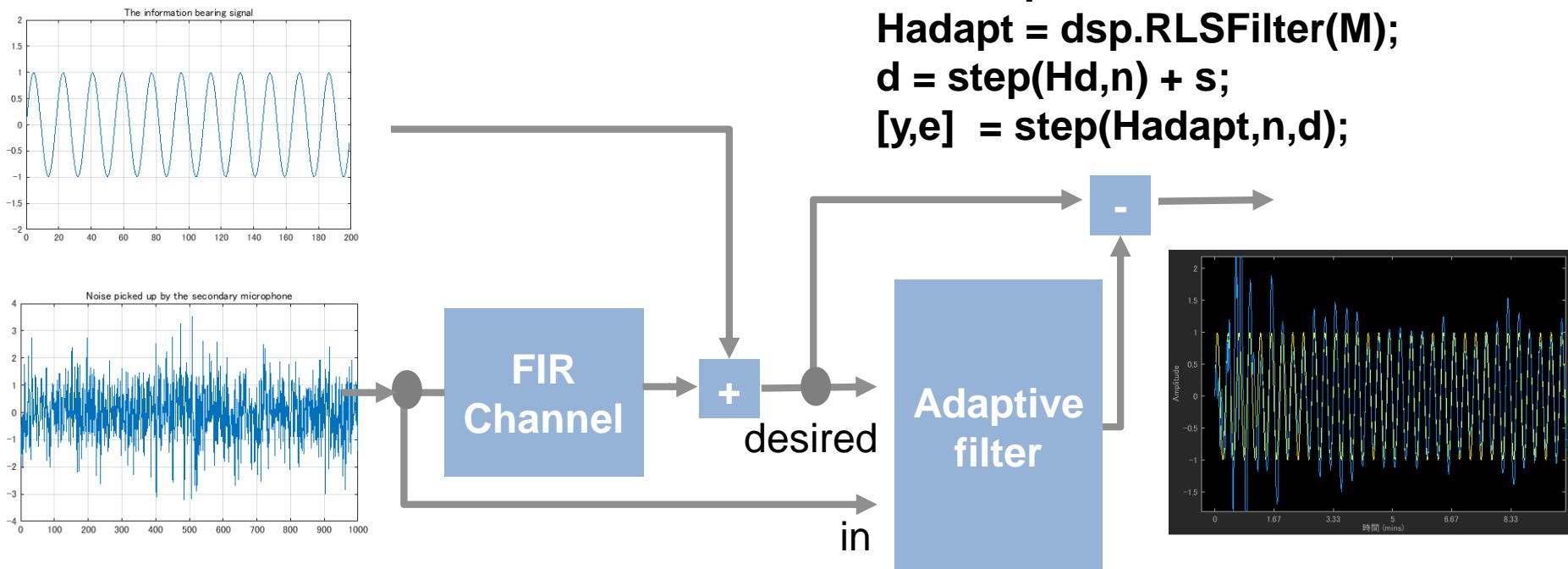
課題:

- MATLABによる適応アルゴリズムの実現
- Simulinkによる適応アルゴリズムの実現

# Case5: ノイズキャンセリング(MATLABによるRLS)

- n: noise
- s: signal

```
Hd = dsp.FIRFilter;
Hadapt = dsp.RLSFilter(M);
d = step(Hd,n) + s;
[y,e] = step(Hadapt,n,d);
```



Dsp.RLSFilter  
により実現

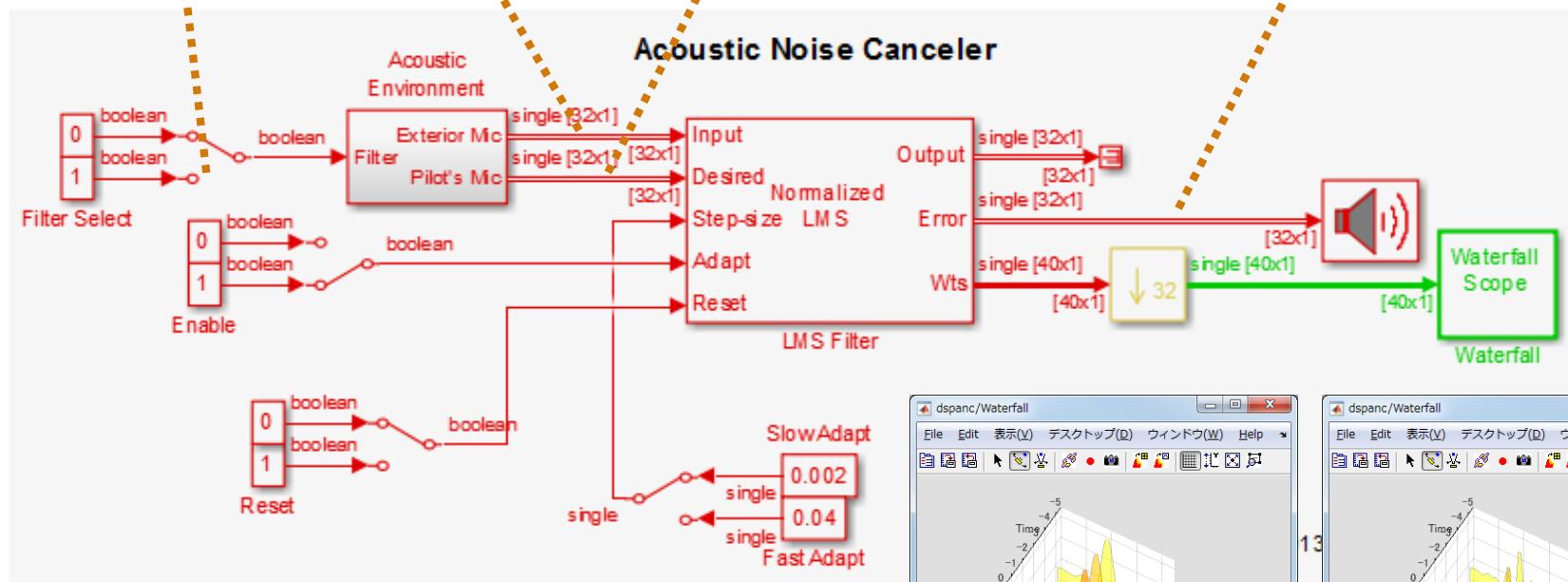
# Case5: ノイズキャンセリング(SimulinkによるLMS)

雑音環境  
切り替え

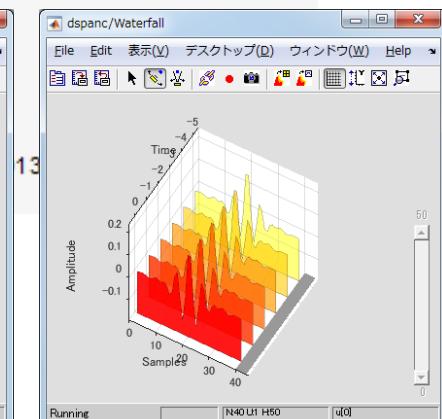
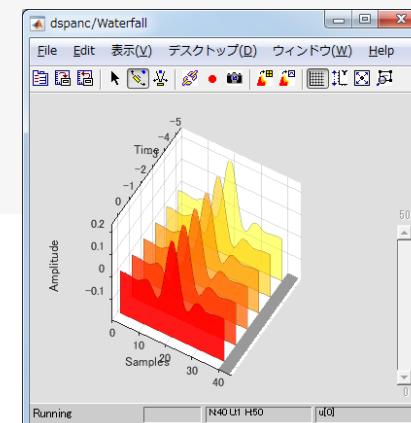
雑音取得用  
外部マイク

pilotマイク

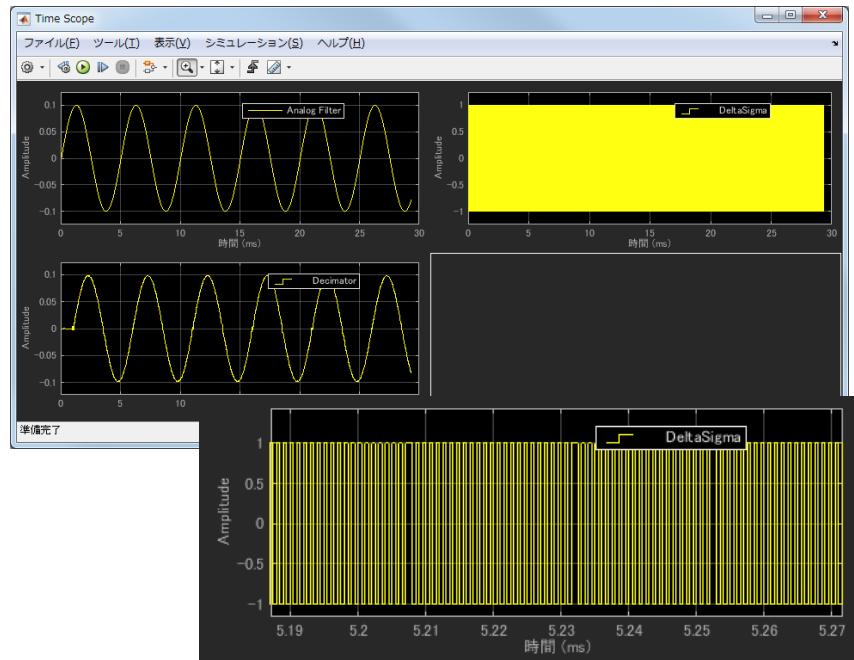
誤差信号  
(取得信号)



LMSブロック  
により実現



# Case6: $\Delta\Sigma$ 型ADコンバータ

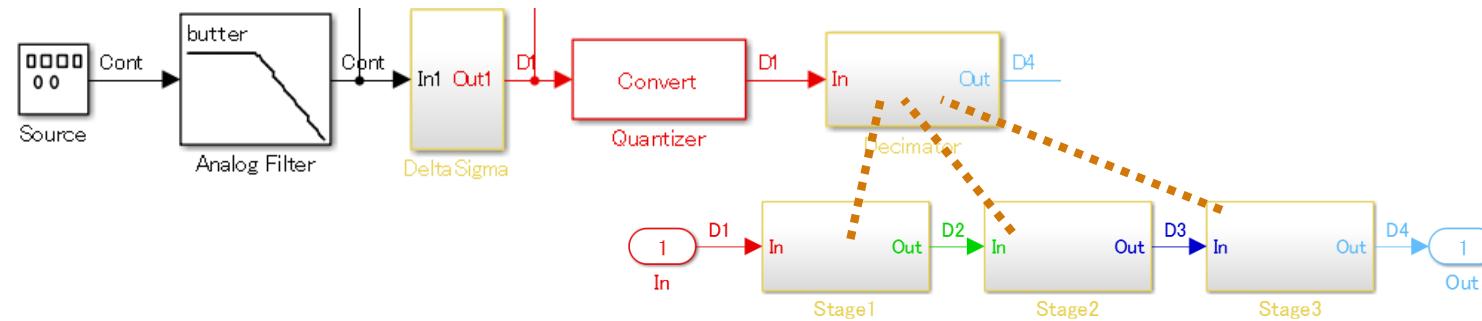


**目的:**

- アナログ部アーキテクチャ検討
- デジタル部レート変換、  
固定小数点解析

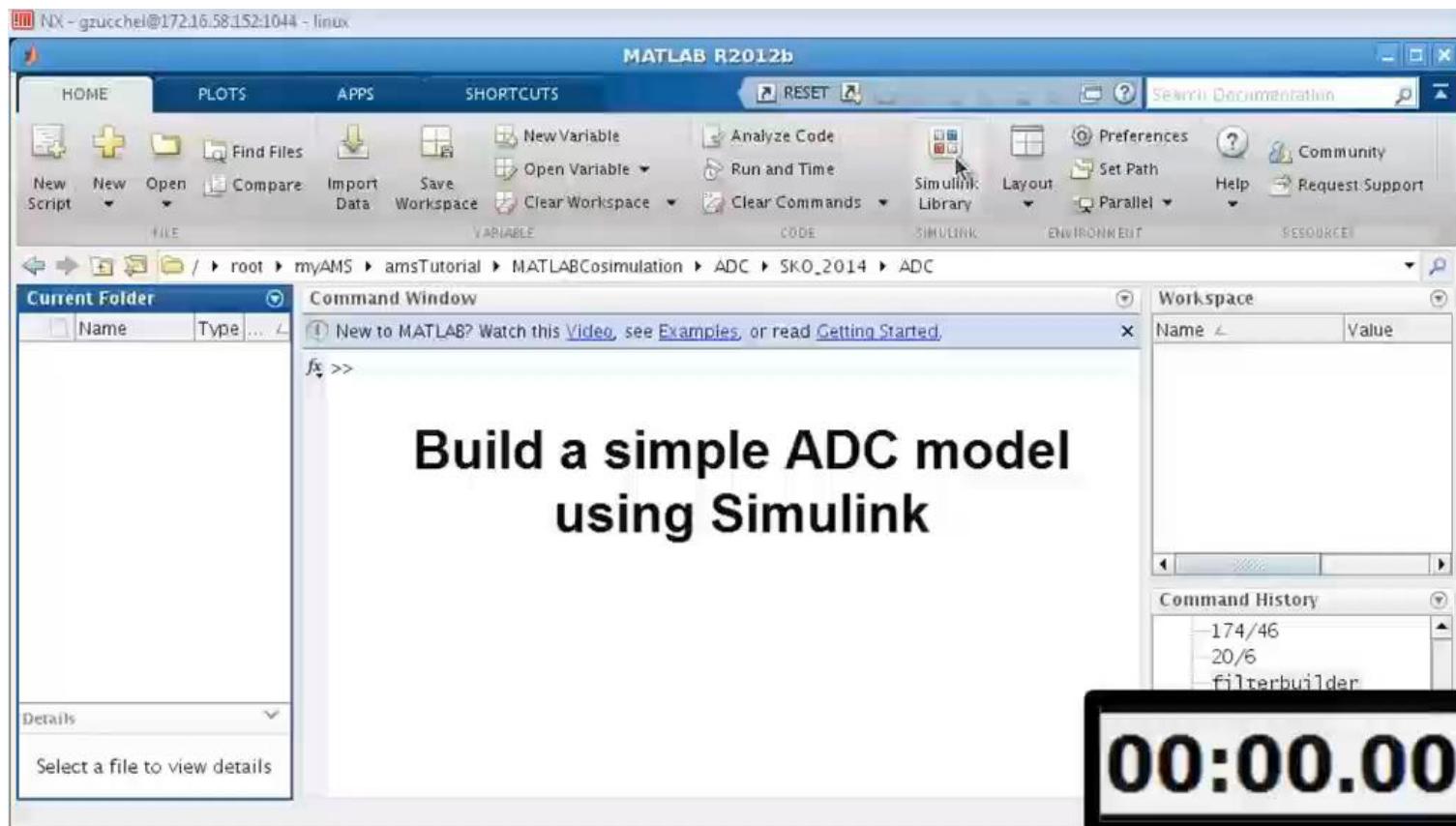
**課題:**

- アナログミックスドシグナルの  
ハンドリング
- フィルタ設計ワークフロー

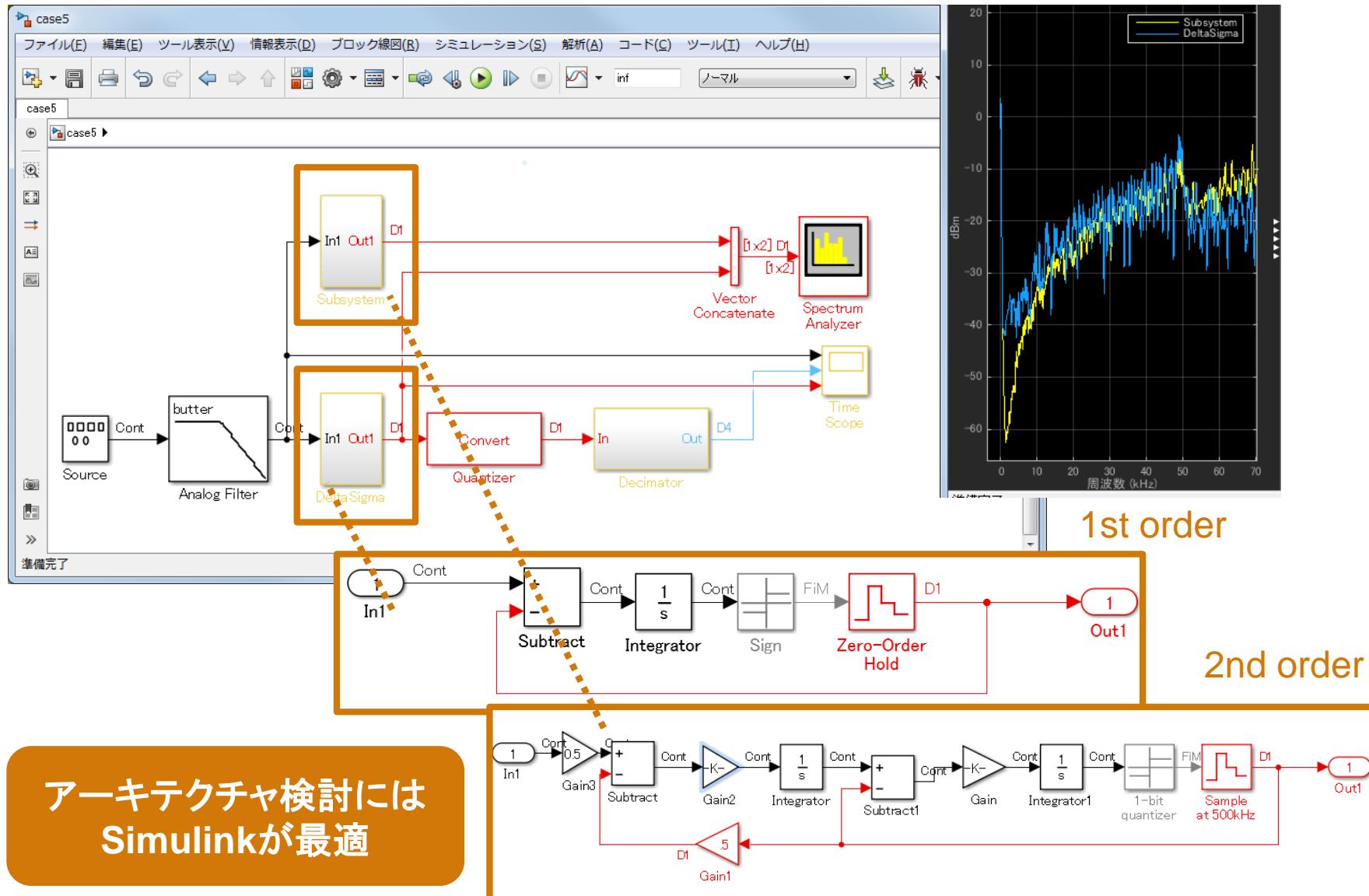


# Demo: Simulinkによる信号処理システムモデリング例

- $\Delta\Sigma$ ADコンバータ
  - アナログフィルタ(アンチエイリアシング)
  - デジタルフィルタ(デシメーション)



# Case6: ΔΣ型ADコンバータ(アナログ処理部)



アーキテクチャ検討には  
Simulinkが最適

# Case6: ΔΣ型ADコンバータ(デジタル処理部)

**①>>filterbuilder  
(応答選択)**

**③固定小数点設定**

**②間引き設定**

**④Simulink ブロック  
自動生成**

**デシメーションフィルタの  
周波数特性**

**filterbuilderによる  
フィルタ設計フロー**

# Agenda

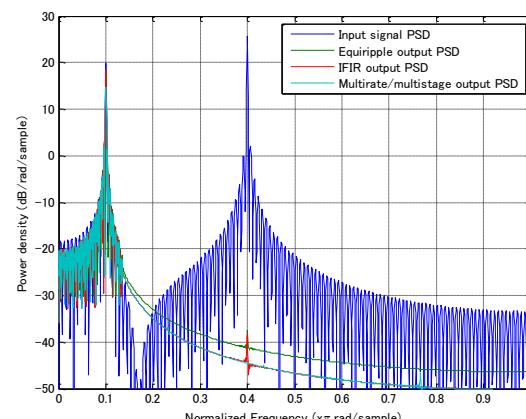
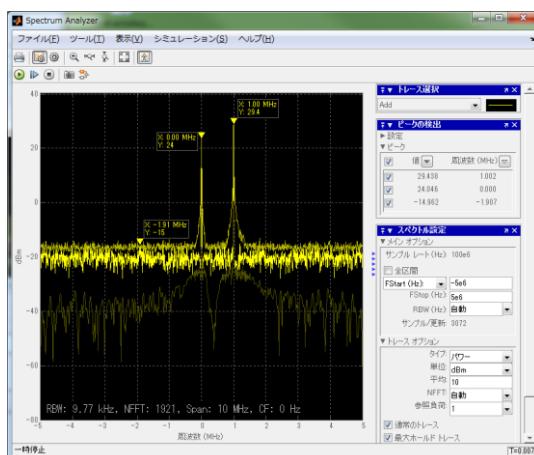
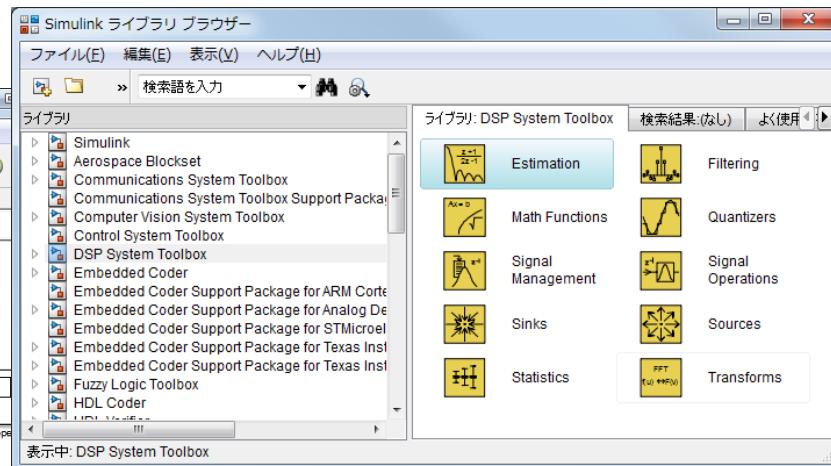
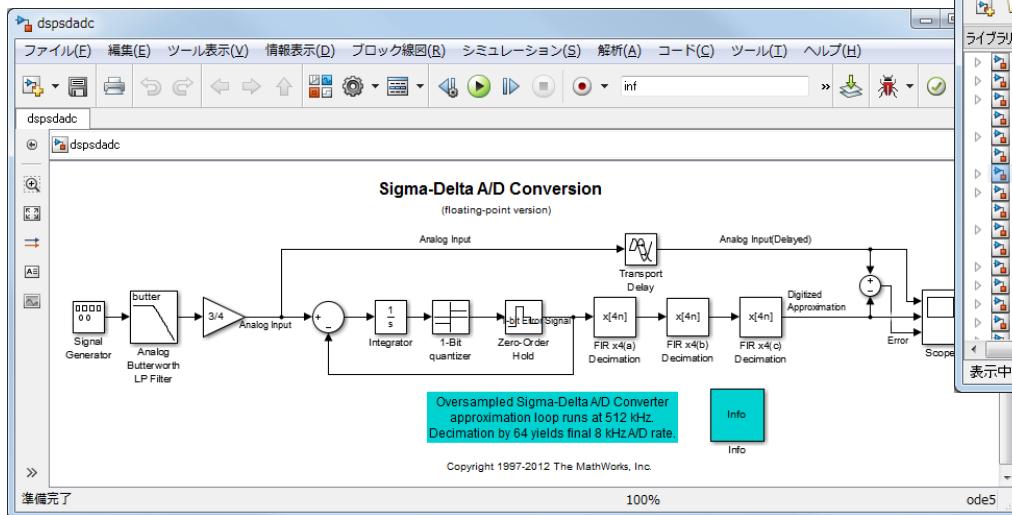
- Section1: フィルタとは？
- Section2: Case study
- **Section3: フィルタの実装**
- Section4: フィルタ設計FAQ
- まとめ

# Agenda

## ■ Section3: フィルタの実装

- 実装用コード生成環境
- CMSIS/Ne10ライブラリ対応状況
- デモ
- STMicro Discoveryボード

# 信号処理システム設計ライブラリ DSP System Toolbox™



- 高度なフィルタ設計
  - マルチレート, 適応フィルタ, 固定小数点  
(*Fixed-Point Designer™*)
- スペクトル解析・スペアナ表示
- 行列・統計処理
- FFT/DCT/DWT
- ARM® Cortex®-M CMSIS, Cortex-A Ne10対応  
(*Embedded Coder™*)

# CMSIS/Ne10ライブラリサポート状況

**R2013b**

**R2014b**

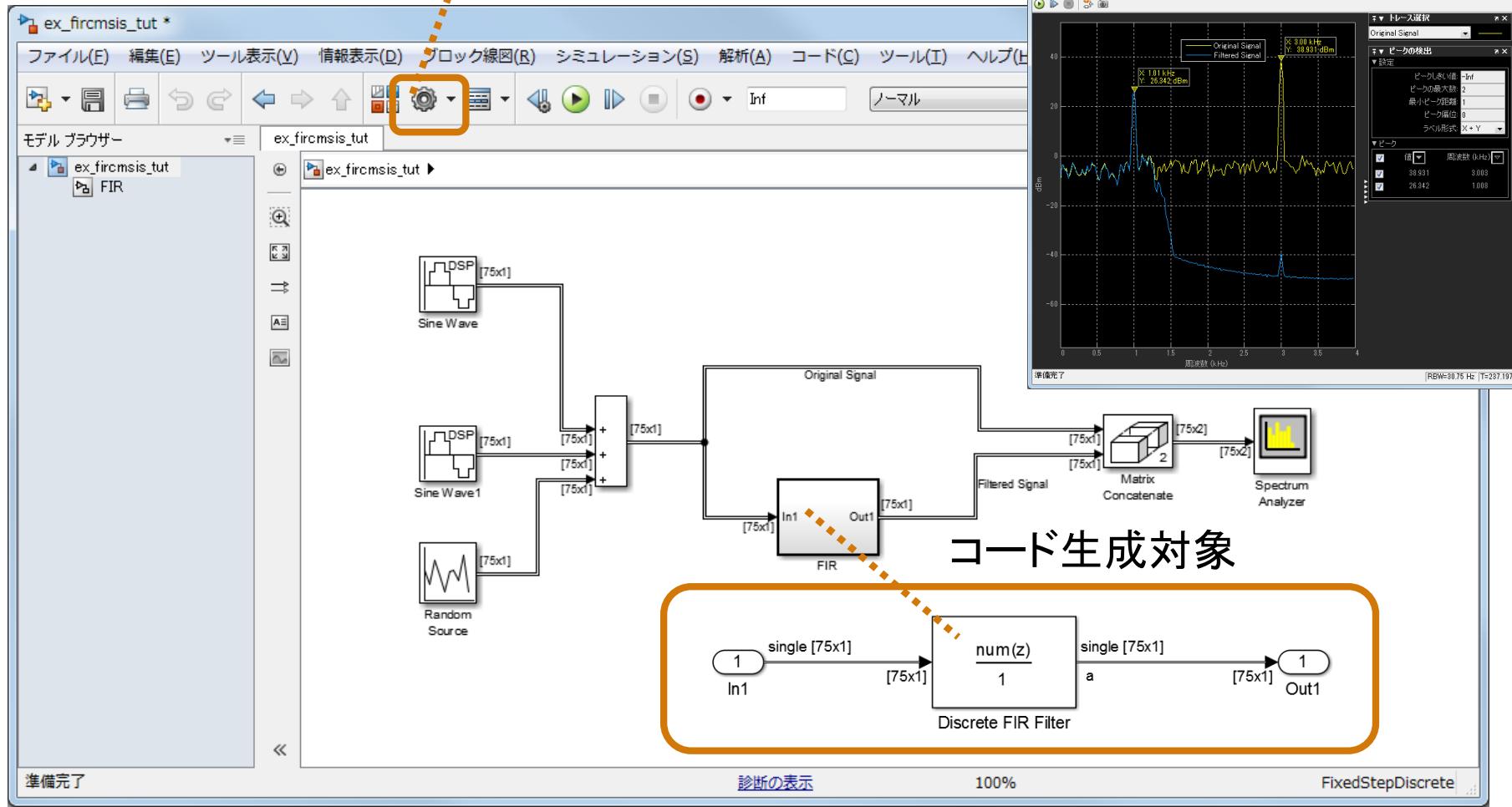
機能	概要	CMSIS	Ne10
<a href="#">Discrete FIR Filter</a>	Model FIR filters	○	○
<a href="#">FIR Decimation</a>	Filter and downsample input signals	○	○
<a href="#">FIR Interpolation</a>	Upsample and filter input signals	○	○
<a href="#">LMS Filter</a>	Compute output, error, and weights using LMS adaptive algorithm	○	
<a href="#">Biquad Filter</a>	Model biquadratic IIR (SOS) filters	○	
<a href="#">FFT</a>	Fast Fourier transform (FFT) of input	○	○
<a href="#">IFFT</a>	Inverse fast Fourier transform (IFFT) of input	○	○
<a href="#">Correlation</a>	Cross-correlation of two inputs	○	
<a href="#">Convolution</a>	Convolution of two inputs	○	
<a href="#">Mean</a>	Find mean value of input or sequence of inputs	○	
<a href="#">RMS</a>	Compute root-mean-square value of input or sequence of inputs	○	
<a href="#">Variance</a>	Compute variance of input or sequence of inputs	○	
<a href="#">Standard Deviation</a>	Find standard deviation of input or sequence of inputs	○	

ブロック・システムオブジェクトに対応

# Case1:浮動小数点FIRフィルタ

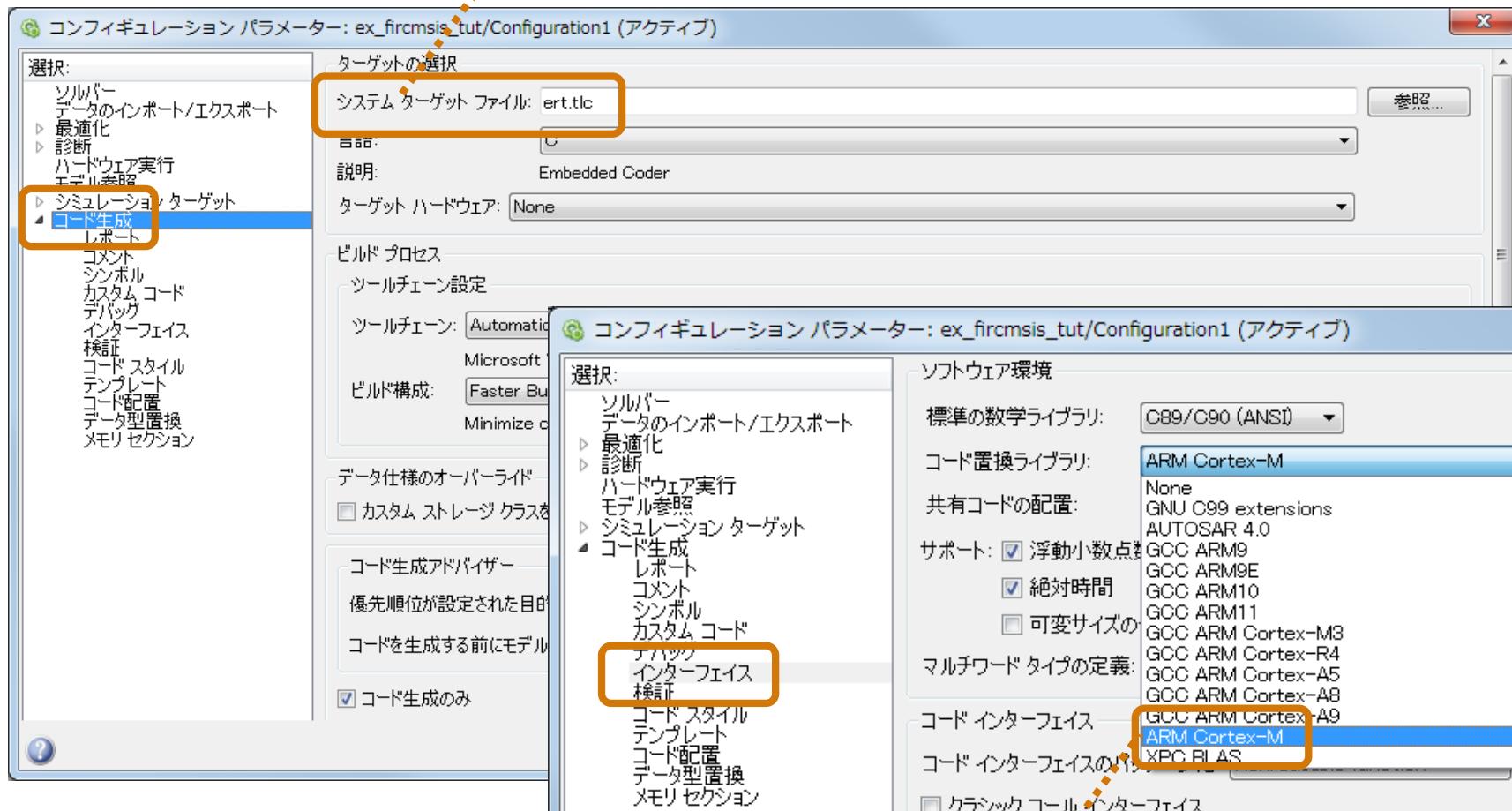
>>ex\_fircmsis\_tut

コンフィギュレーションパラメータ



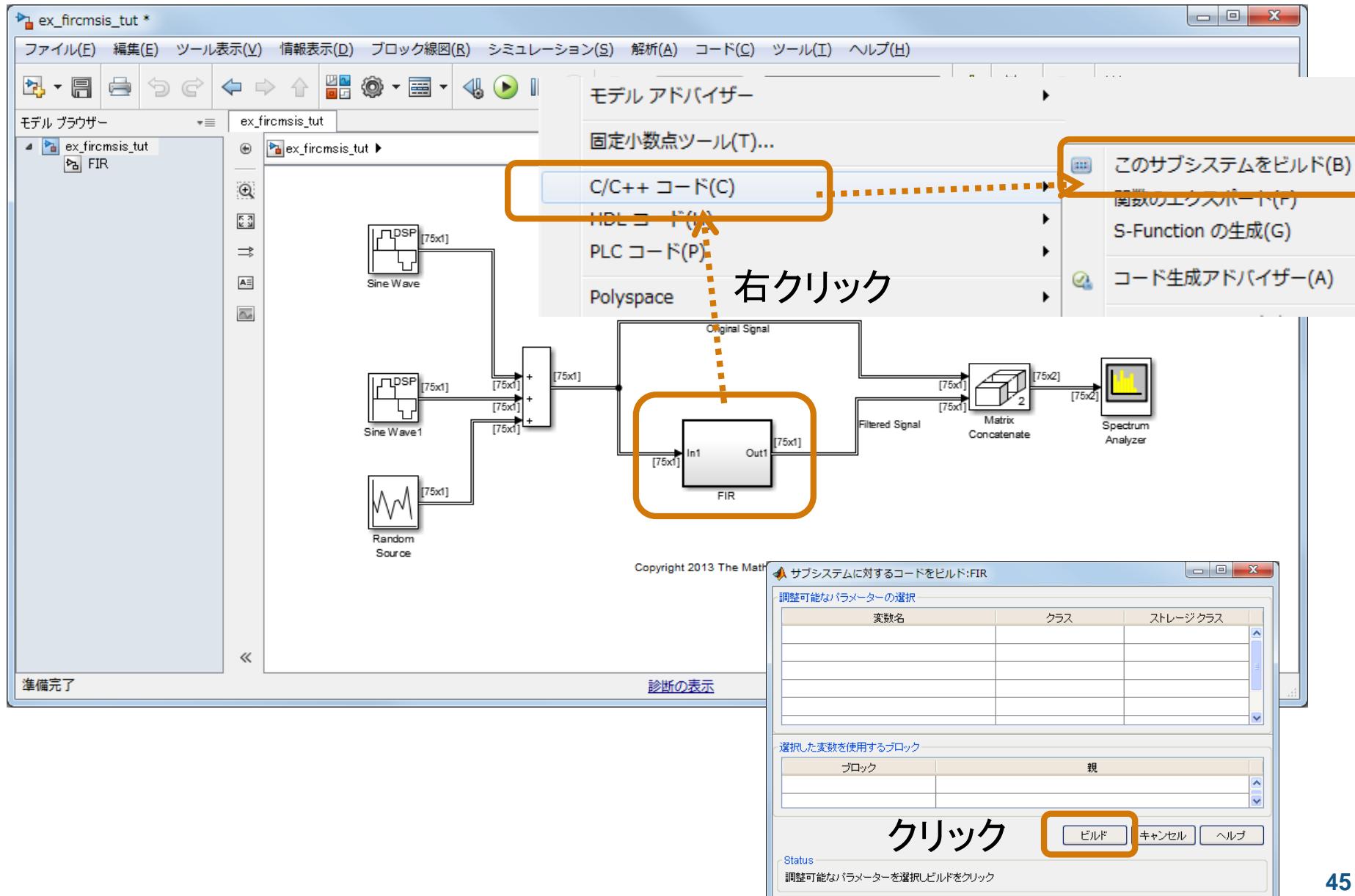
# Case1: 浮動小数点FIRフィルタ(Cortex-M)

システムターゲットファイルに、“ert.tlc”を選択



インターフェイス>コード置換ライブラリに、"ARM Cortex-M"を選択

# Case1:浮動小数点FIRフィルタ(Cortex-M)



# Case1:浮動小数点FIRフィルタ(Cortex-M)

Code Generation Report

検索: 大文字小文字を区別

**Contents**

- [Summary](#)
- [Subsystem Report](#)
- [Code Interface Report](#)
- [Traceability Report](#)
- [Static Code Metrics Report](#)
- [Code Replacements Report](#)

---

**Generated Code**

- [\[-\] Main file](#)
  - [ert\\_main.c](#)
- [\[-\] Model files](#)
  - [FIR.c](#) (highlighted)
  - [FIR.h](#)
  - [FIR\\_private.h](#)
  - [FIR\\_types.h](#)

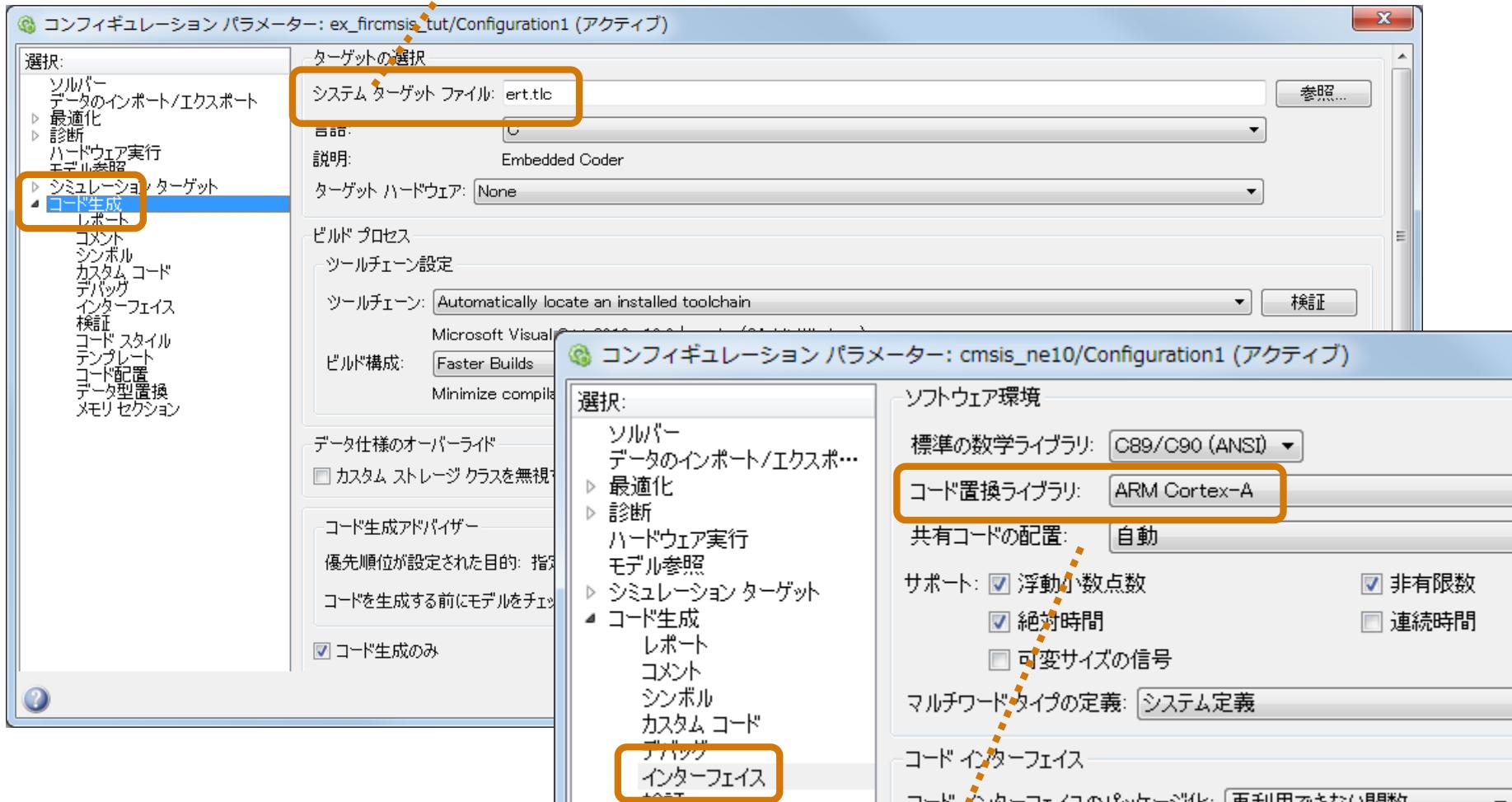
```

20 DW FIR T FIR_DW;
21
22 /* External inputs (root import signals with auto storage) */
23 ExtU FIR T FIR_U;
24
25 /* External outputs (root outports fed by signals with auto storage) */
26 ExtY FIR T FIR_Y;
27
28 /* Real-time model */
29 RT_MODEL FIR T FIR_M;
30 RT_MODEL FIR T *const FIR_M = &FIR_M;
31
32 /* Model step function */
33 void FIR_step(void)
34 {
35     real32 T rtb_DiscreteFIRFilter[75];
36
37     /* Outputs for Atomic SubSystem: '<Root>/FIR' */
38     /* DiscreteFir: '<SI>/Discrete FIR Filter' incorporates:
39      * Inport: '<Root>/InT'
40      */
41     arm_fir_f32(&FIR_DW.S, &FIR_U.OriginalSignal[0], &rtb_DiscreteFIRFilter[0]
42                                         75U);
43
44     /* End of Outputs for SubSystem: '<Root>/FIR' */
45
46     /* Outport: '<Root>/Out1' */
47     memcpy(&FIR_Y.Out1[0], &rtb_DiscreteFIRFilter[0], 75U * sizeof(real32 T));
48 }
```

FIRフィルタ部について、  
CMSIS関数"arm\_fir\_f32"を生成

# Case1:浮動小数点FIRフィルタ(Cortex-A)

システムターゲットファイルに、“ert.tlc”を選択



インターフェイス>コード置換ライブラリに、"ARM Cortex-A"を選択

# Case1:浮動小数点FIRフィルタ(Cortex-A)

Code Generation Report

検索: 大文字小文字を区別

**Contents**

- [Summary](#)
- [Subsystem Report](#)
- [Code Interface Report](#)
- [Traceability Report](#)
- [Static Code Metrics Report](#)
- [Code Replacements Report](#)

**Generated Code**

- [–] **Main file**
  - [ert\\_main.c](#)
- [–] **Model files**
  - [FIR.c](#) (highlighted)
  - [FIR.h](#)
  - [FIR\\_private.h](#)
  - [FIR\\_types.h](#)
- [+] Utility files (1)

```

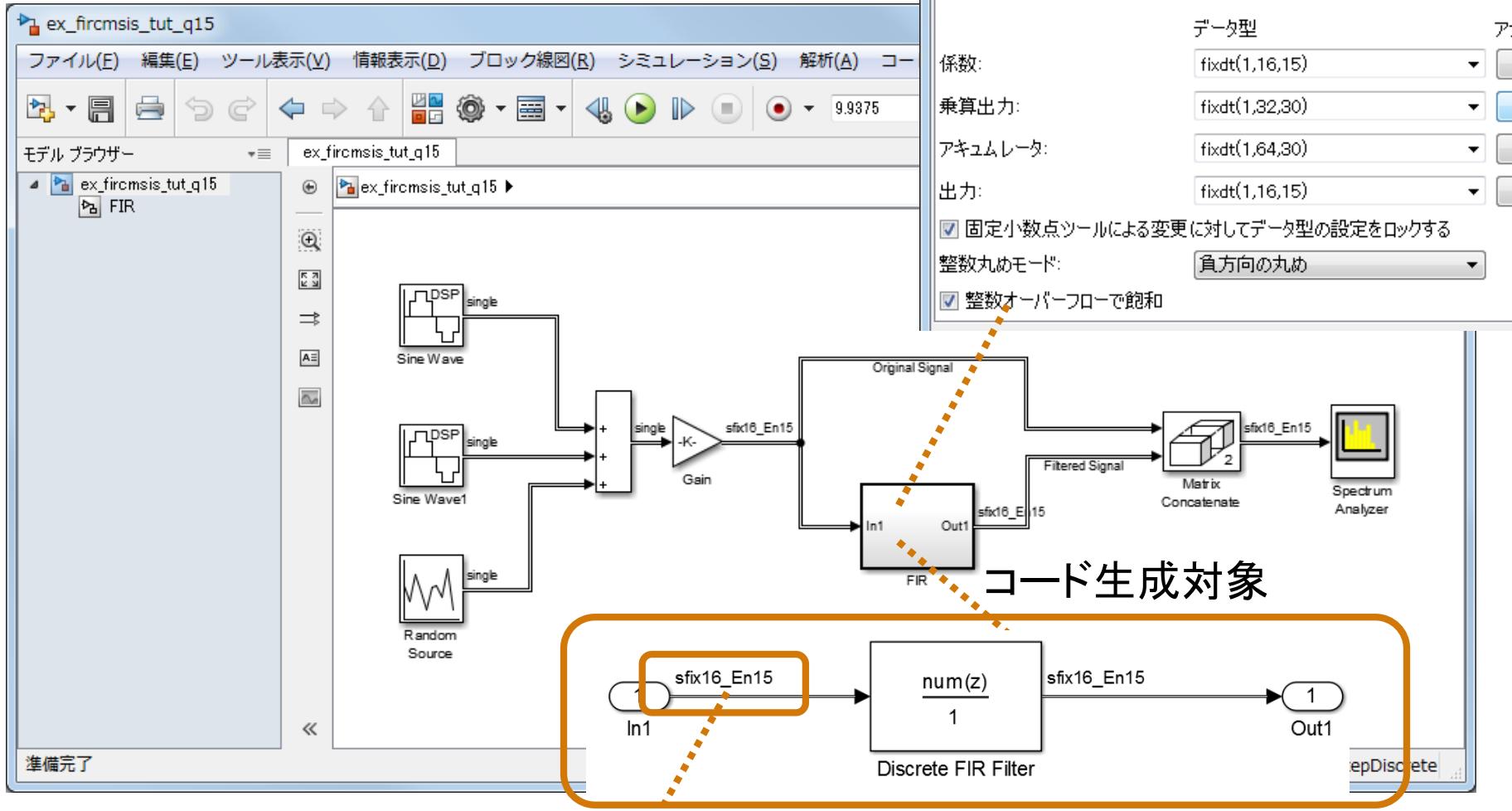
19 /* Block states (auto storage) */
20 DW_FIR_T FIR_DW;
21
22 /* External inputs (root import signals with auto storage) */
23 ExtU_FIR_T FIR_U;
24
25 /* External outputs (root outports fed by signals with auto storage) */
26 ExtY_FIR_T FIR_Y;
27
28 /* Real-time model */
29 RT_MODEL_FIR_T FIR_M;
30 RT_MODEL_FIR_T *const FIR_M = &FIR_M;
31
32 /* Model step function */
33 void FIR_step(void)
34 {
35     real32_T rtb_DiscreteFIRFilter[75];
36
37     /* Outputs for Atomic SubSystem: '<Root>/FIR' */
38     /* DiscreteFir: '<S1>/Discrete FIR Filter' incorporates:
39      * Import: '<Root>/In1'
40      */
41     ne10_fir_float_neon(&FIR_DW.S, &FIR_U.OriginalSignal[0],
42                         &rtb_DiscreteFIRFilter[0], 75U);
43
44     /* End of Outputs for SubSystem: '<Root>/FIR' */
45

```

FIRフィルタ部について、  
"ne10\_fir\_float\_neon"を生成

# Case2: 固定小数点FIRフィルタ

>>ex\_fircmsis\_tut\_q15



sfix16\_En15: 符号付、語長16bit、小数部15bit

## Case2: 固定小数点FIRフィルタ

Code Generation Report

検索: 大文字小文字を区別

**Contents**

[Summary](#)

[Subsystem Report](#)

[Code Interface Report](#)

[Traceability Report](#)

[Static Code Metrics Report](#)

[Code Replacements Report](#)

---

**Generated Code**

[ -] Main file  
  [ert\\_main.c](#)

[ -] Model files

[FIR.c](#) (highlighted)

[FIR.h](#)

[FIR\\_private.h](#)

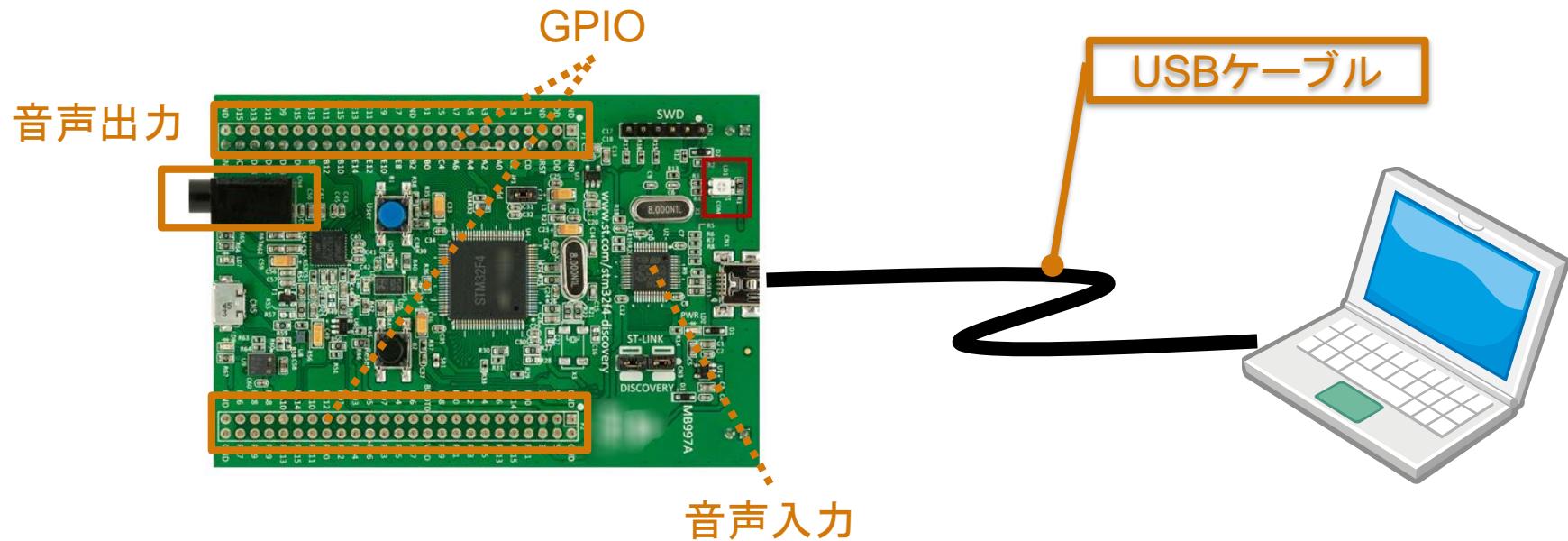
[FIR\\_types.h](#)

```
24 /* External outputs (root outports fed by signals with auto storage) */
25 ExtY FIR_T FIR_Y;
26
27 /* Real-time model */
28 RT_MODEL FIR_T FIR_M;
29 RT_MODEL FIR_T *const FIR_M = &FIR_M;
30
31 /* Model step function */
32 void FIR_step(void)
33 {
34   int16_T rtb_DiscreteFIRFilter[75];
35
36   /* Outputs for Atomic SubSystem: '<Root>/FIR' */
37   /* DiscreteFir: '<S1>/Discrete FIR Filter' incorporates:
38    * Inport: '<Root>/In1'
39    */
40
41   arm_fir_q15(&FIR_DW.S, &FIR_U.OriginalSignal[0], &rtb_DiscreteFIRFilter[0],
42               75U);
43
44   /* End of Outputs for SubSystem: '<Root>/FIR' */
45
46   /* Outport: '<Root>/Out1' */
47   memcpy(&FIR_Y.Out1[0], &rtb_DiscreteFIRFilter[0], 75U * sizeof(int16_T));
48 }
49
50 /* Model initialize function */
51 void FIR_initialize(void)
52 {
53   /* Pre-initialization code
54 }
```

FIRフィルタ部について、  
CMSIS関数"arm\_fir\_q15"を生成

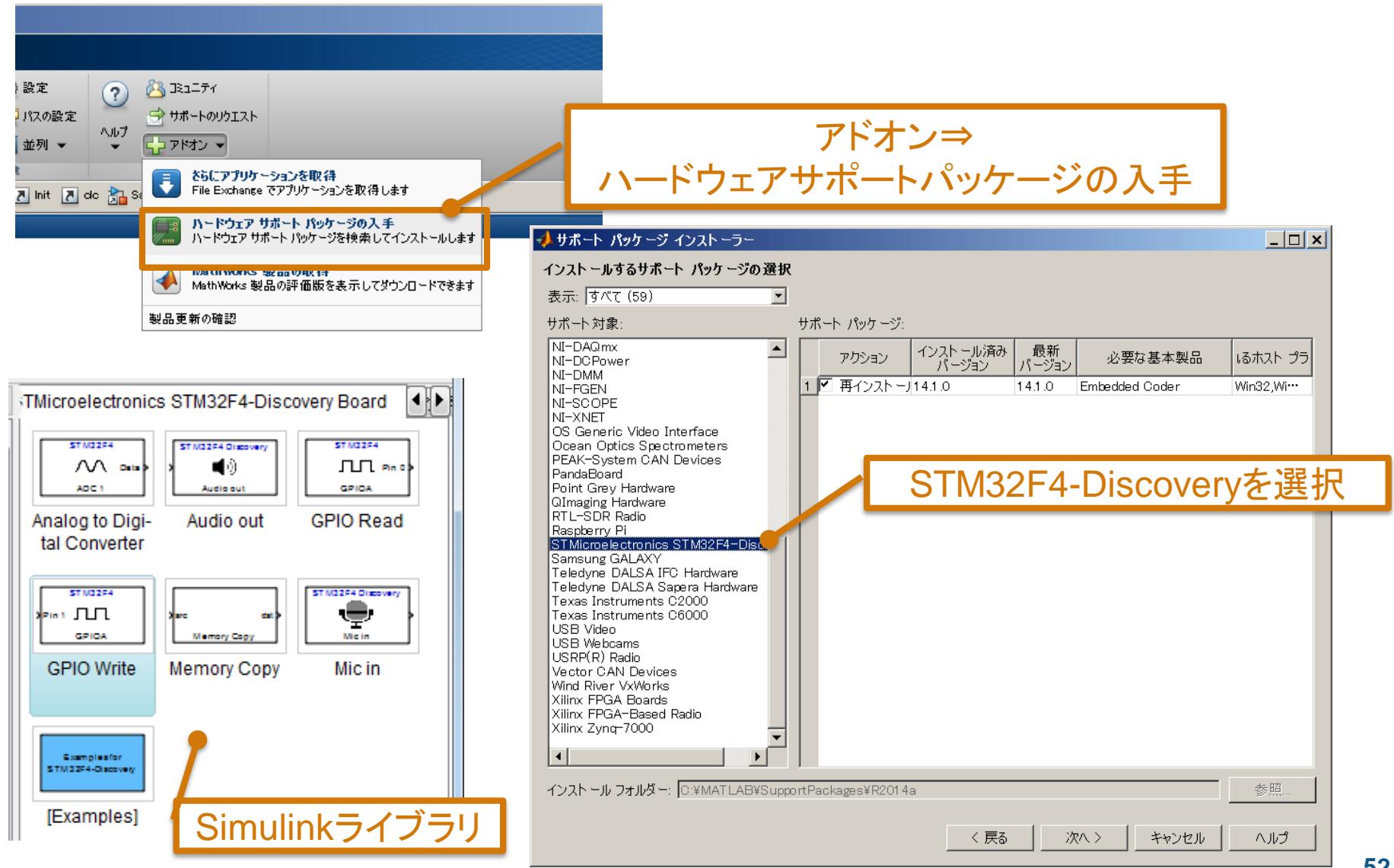
# STM32F4-Discovery®

- Simulinkで作成したモデルを直接プロセッサーにダウンロードして実行
- ダウンロードにはST-LINKを使用
- 接続はUSBのみ\*(電源供給、ダウンロード、PIL)



\*高速PILシミュレーションおよびエクステンダルモードでの実行には、シリアル通信用アダプターを使用

# ライブラリインストール

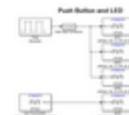


# チュートリアル、例題

## インストールされたサポート パッケージ

DSP System Toolbox Support Package for ARM Cortex-M Processors  
MATLAB Support Package for USB Webcams  
Simulink Support Package for Arduino Hardware  
Communications System Toolbox Support Package for RTL-SDR Radio  
Embedded Coder Support Package for ARM Cortex-M Processors  
Simulink Support Package for Raspberry Pi Hardware  
MATLAB Support Package for Arduino Hardware  
Simulink Support Package for Arduino Due Hardware  
**Embedded Coder Support Package for STMicroelectronics STM32F4-Discovery Board**  
DSP System Toolbox Support Package for ARM Cortex-A Processors  
Embedded Coder Support Package for ARM Cortex-A Processors  
MATLAB Support Pack

## Examples



**Push Button and LED**  
Uses: Embedded Coder



**Parametric Audio Equalizer for STM32F4-Discovery Board**  
Uses: Embedded Coder, DSP System Toolbox



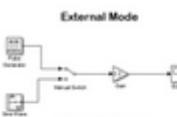
**Asynchronous Scheduling**  
Uses: Embedded Coder

## Tutorials

### Tutorials



**Getting Started with Embedded Coder Support Package for STMicroelectronics STM32F4-Discovery**  
Uses: Embedded Coder



**Code Verification and Validation with PIL and External Mode**  
Uses: Embedded Coder



**Code Optimization using CMSIS DSP Library**  
Uses: Embedded Coder, DSP System Toolbox

## Examples

# コード生成～マニュアル・ダウンロード手順

1. 例「Push Button and LED」をオープン
2. [Build action]を[Build]に設定



3. コード生成ボタン(モデルのビルド)を押下すると  
ビルドが行われ、カレントディレクトリに\*\*.hexが生成される。
4. ボードにUSB接続したら  
STM32 ST-Link Utilityをオープン
5. [Target]メニュー⇒[Connect]
6. [Target]メニュー⇒[Program & Verify]  
を選択し、生成された\*.hexファイルを指定しダウンロード。



# コード生成～オート・ダウンロード手順

1. 例「Push Button and LED」をオープン
2. コンフィギュレーションパラメータの[Build action]を[Build, load and run]に設定



3. コード生成ボタン(モデルのビルド)を押下すると  
ビルドが行われ、カレントディレクトリに\*\*.hexが生成される。
4. プログラムのダウンロードまで自動的に行われる。



# Demo (GPIOテスト)

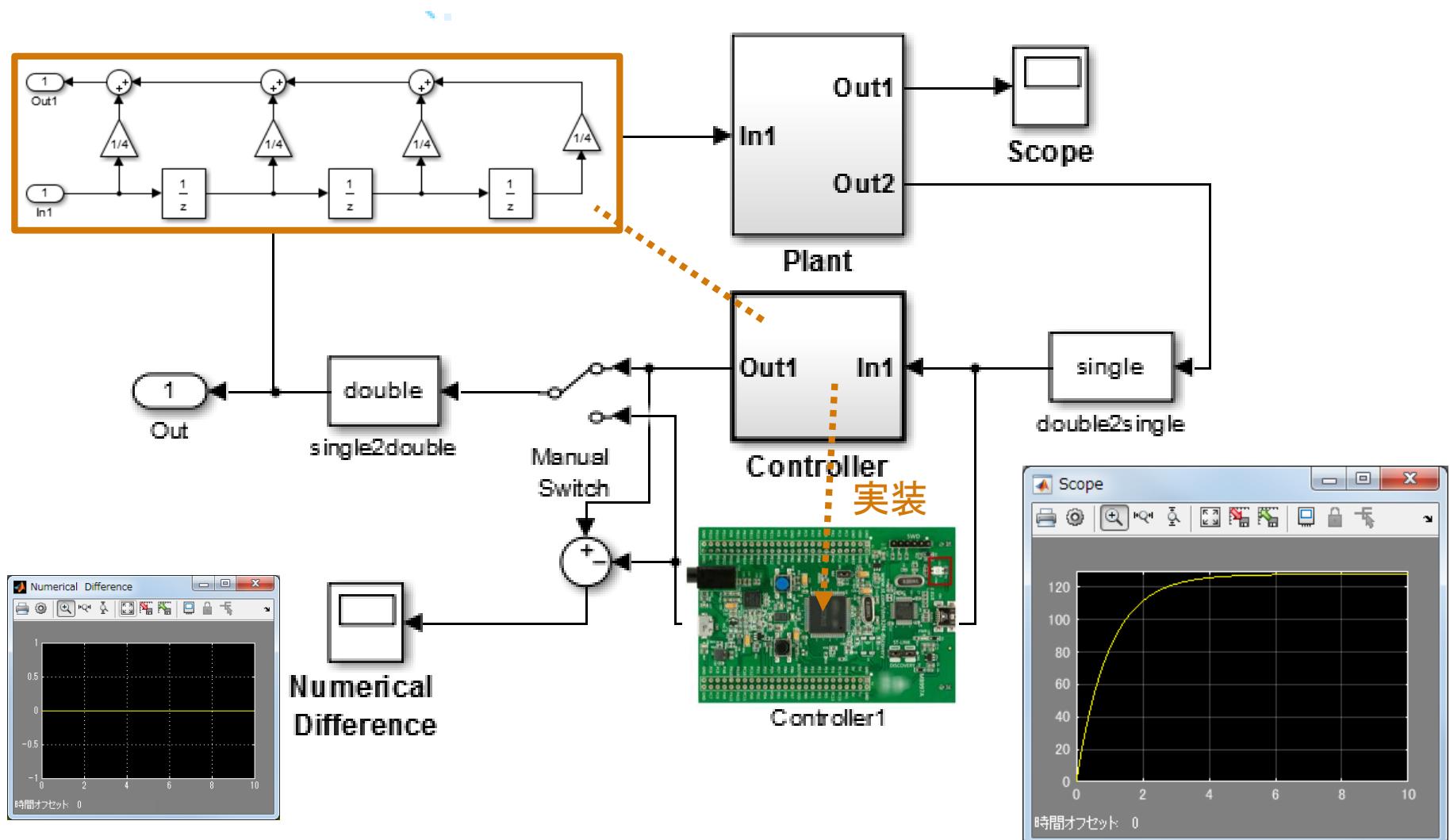
The screenshot shows the MATLAB R2014a desktop environment. The top menu bar includes Home, Plot, Application, and Shortcut tabs. The toolbar contains icons for New Script, New File, Open, Import Data, Save, New Variable, Code Analysis, Run and Time Profiler, Clear Command Window, Simulink, Layout, Help, and Add-ons. The current workspace path is displayed as c:\mywork\Event\ARMSTM2014. The left pane shows a file browser with the following contents:

名前	サイズ	更新日	タイプ
Controller_ert_rtw		2014/08/18 13:47	フォルダー
presentation		2014/08/18 13:56	フォルダー
slprj		2014/08/18 11:12	フォルダー
stm32f4discovery_pb_led_ert_rtw		2014/08/18 11:19	フォルダー
stm32f4discovery_top_model_pil_ert_rtw		2014/08/18 14:26	フォルダー
videos		2014/08/26 11:20	フォルダー
書類		2014/07/02 9:36	フォルダー
ARMSTMMW2014.pptx	10 MB	2014/07/09 10:19	Microsoft Powerpoint 演示文
ARMSTMMW2014v2.pptx	12 MB	2014/08/26 11:21	Microsoft Powerpoint 演示文
Controller.bin	11 KB	2014/08/18 13:48	VLC media file
Controller.hex	33 KB	2014/08/18 13:48	HEX ファイル
Controller_pbs.mexw64	30 KB	2014/08/18 13:47	MEX ファイル
memo.txt	1 KB	2014/07/02 9:36	テキスト ファイル

The right pane is the Command Window, which currently displays "fx >> |". In the background, there is a photograph of a person's hands connecting a breadboard to a computer via a USB cable.

STM32F4 Discovery GPIO test

# PIL (Processor in the loop)



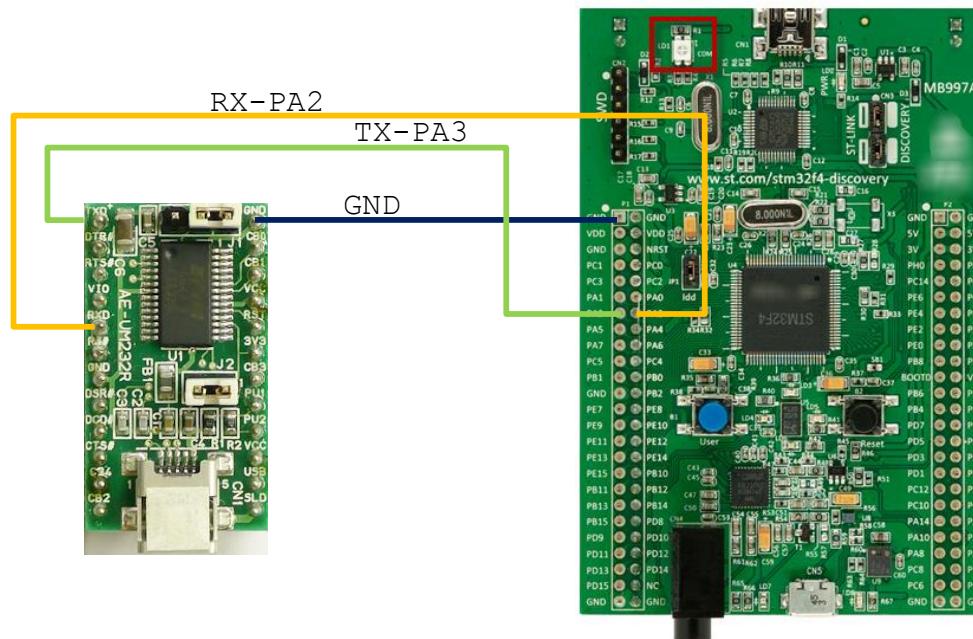
## PIL (cont'd)

### PILは2種類のインターフェースを選択可能

- ST-LinkによるPIL
  - Discovery-PC間のケーブル1本で実施できるので手軽
  - 通信速度が遅い
  - ベンチマーク結果(Tutorial: Code Verification and ...)  
`>> tic, sim(gcs), toc`  
経過時間は 207.729494 秒です。
- USBシリアルインターフェースによるPIL
  - PC-USB Serial-Discovery-PCで接続の必要がある
  - 通信速度が速い
  - ベンチマーク結果(同上) : ST-Linkより8倍高速  
`>> tic, sim(gcs), toc`  
経過時間は 25.948735 秒です。

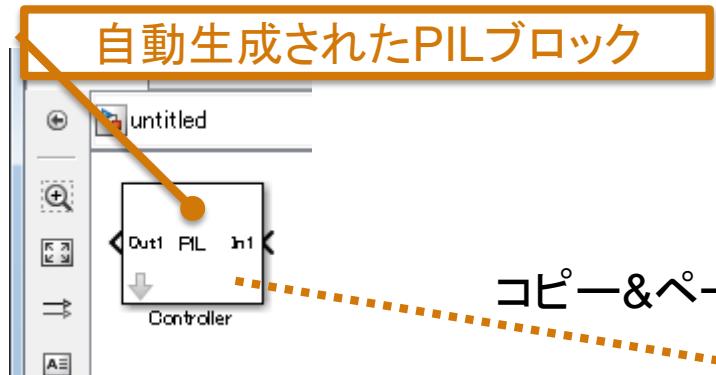
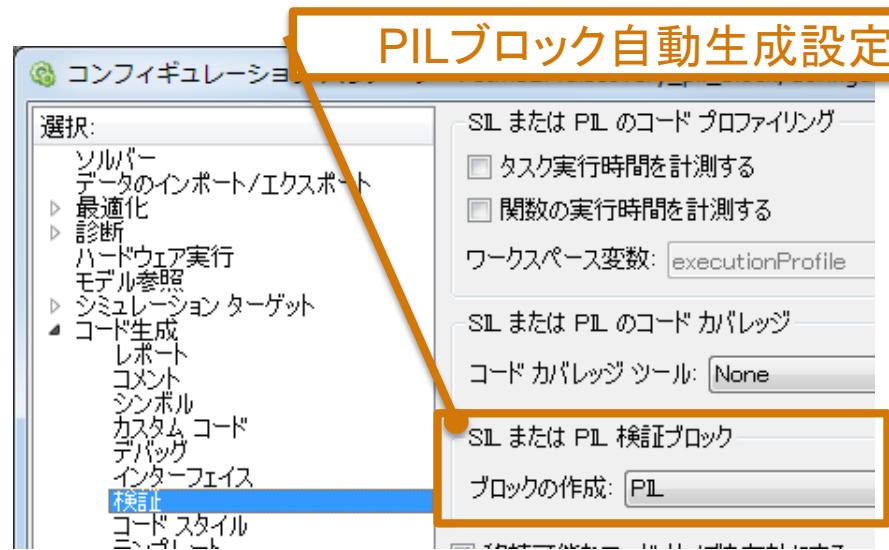
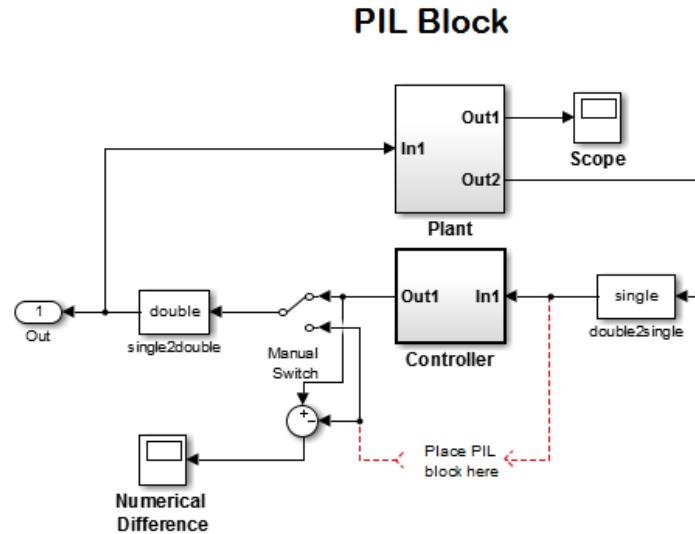
# PIL/エクステナルモード基板配線図

FT-232側ピン	STM32F4 Discovery側ピン
GND	GND
RX(RXD)	PA2
TX(TXD)	PA3

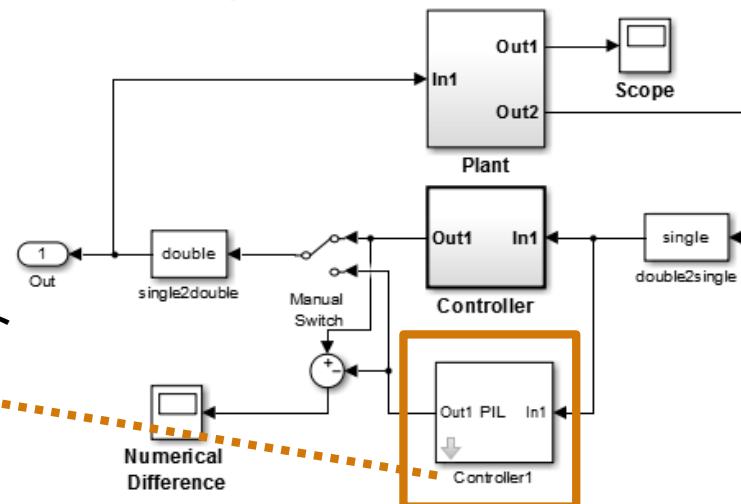


# PIL (Processor in the loop)

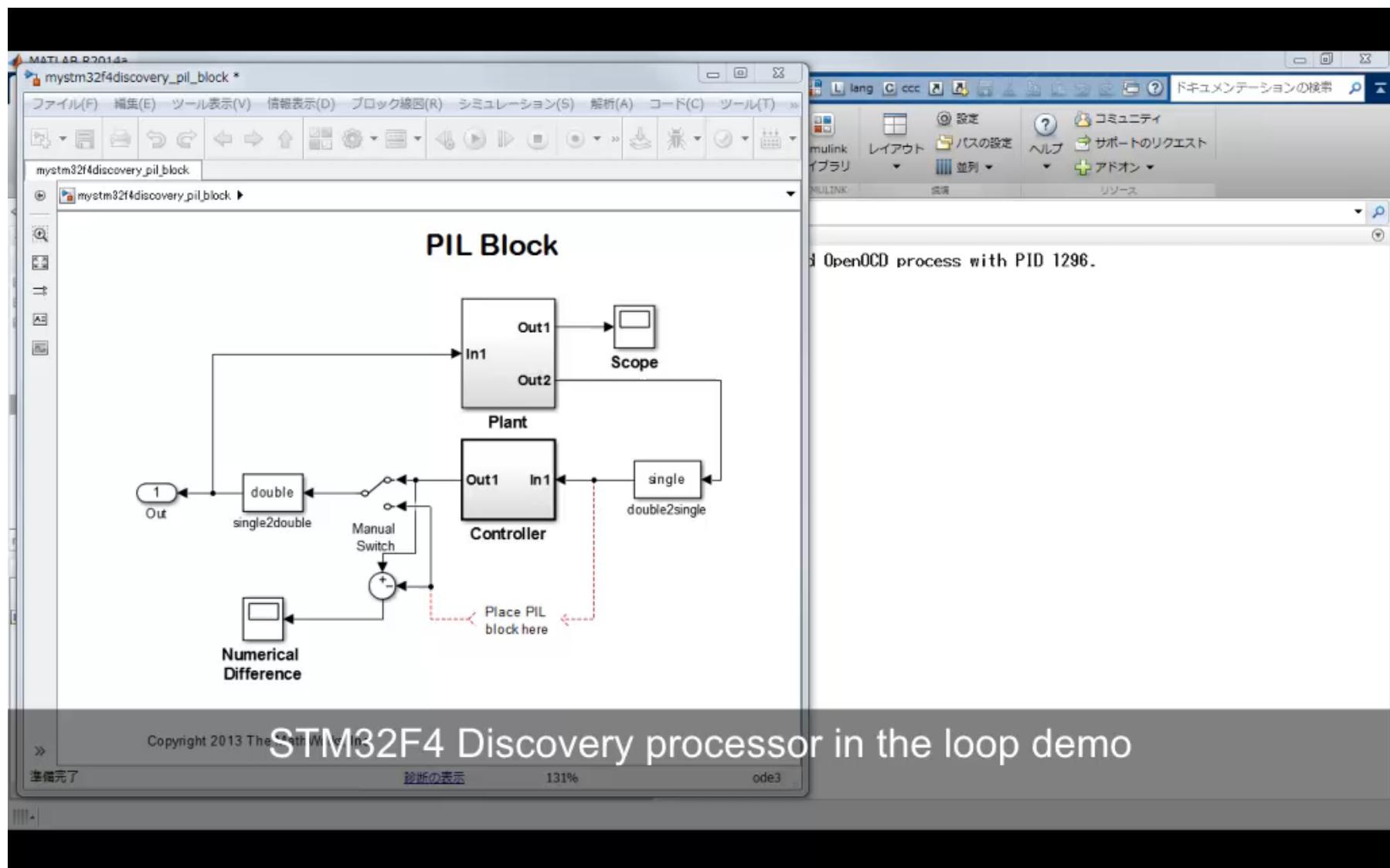
>>stm32f4discovery\_pil\_block



コピー&ペースト



# Demo (PIL)



# Agenda

- Section1: フィルタとは？
- Section2: Case study
- Section3: フィルタの実装
- **Section4: フィルタ設計FAQ**
- まとめ

# Agenda

## ■ Section4: フィルタ設計FAQ

- フィルタの遅延補正の方法は？
- 各種設計環境の違いは？
- アナログフィルタは設計できますか？
- 紙のデータにフィルタをかけるには？
- 信号に欠損がある場合や、  
サンプリングが不等間隔のデータを扱うには？
- テストベンチの効果的な作成方法は？
- テキストやバイナリファイルの  
ストリーミング処理を実現するには？

# Q: フィルタの遅延補正の方法は？

```
d = designfilt( 'lowpassiir' );
```

%%

```
grpdelay(d, N, Fs)
```

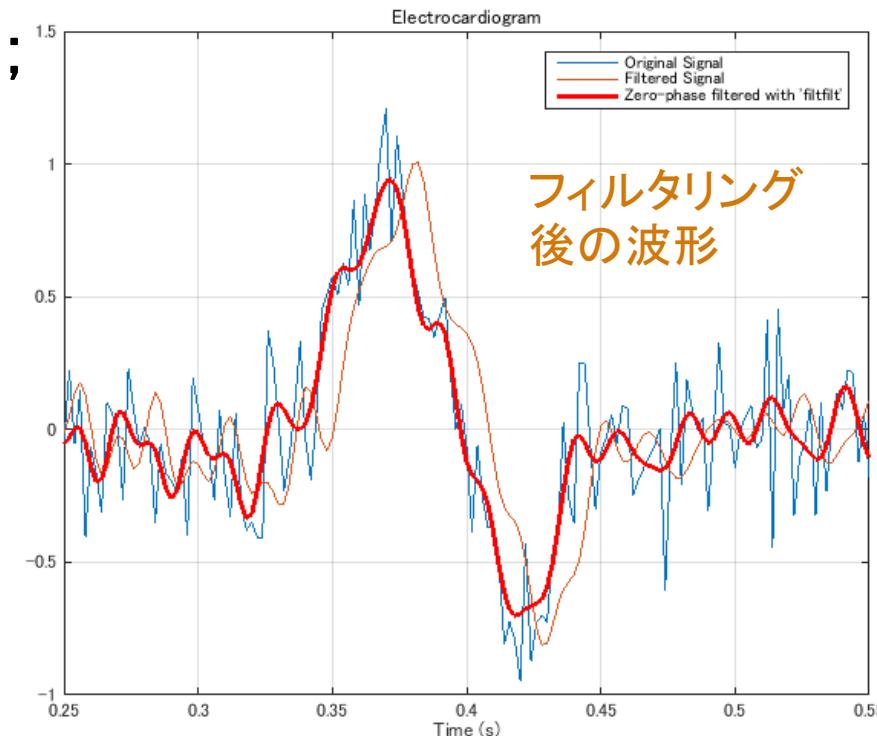
%%

```
out1 = filter(d, xn);
```

通常のフィルタ処理

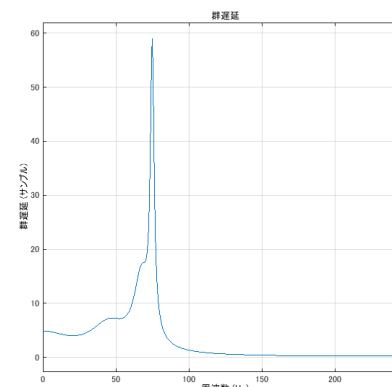
```
out2 = filtfilt(d, xn);
```

遅延補正フィルタ



順方向と逆方向でフィルタリングすることで、  
遅延補正および波形歪低減の効果

**filtfilt**関数で実現

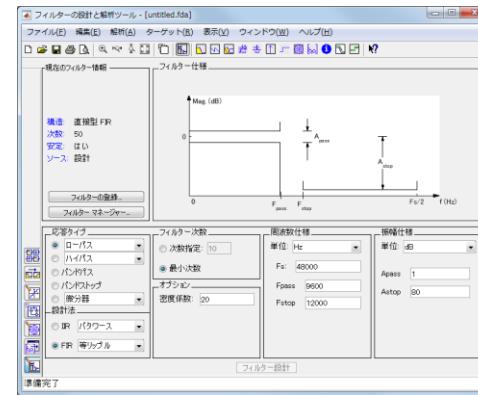


群遅延特性

# Q:各種フィルタ設計環境の違いは?

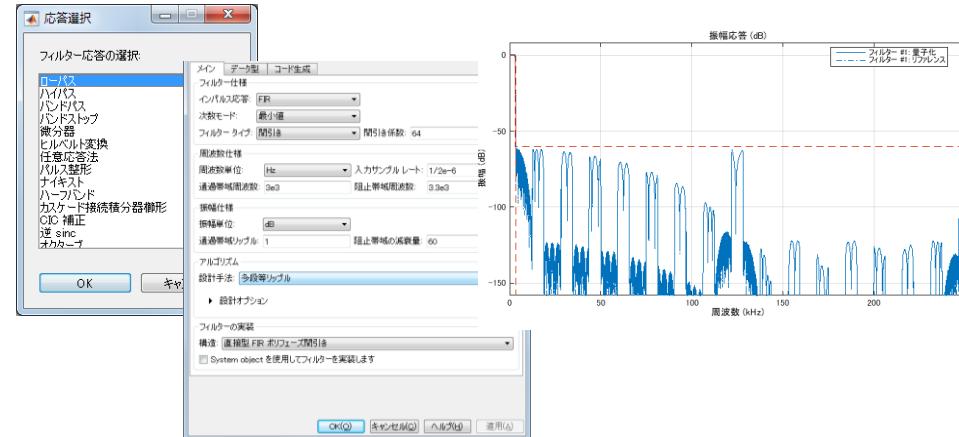
## ■ FDATool

- 従来手法
- フィルタの登録・カスケード
- フィルタの変換( $\text{HPF} \Rightarrow \text{LPF}$ 等)
- 固定小数点化
- Cヘッダ、HDLコード生成



## ■ filterbuilder

- 仕様ベース
- メソッド選択の試行錯誤削減
- 固定小数点化、コード生成等、FDAToolの主機能を踏襲



## ■ Designfilt GUI

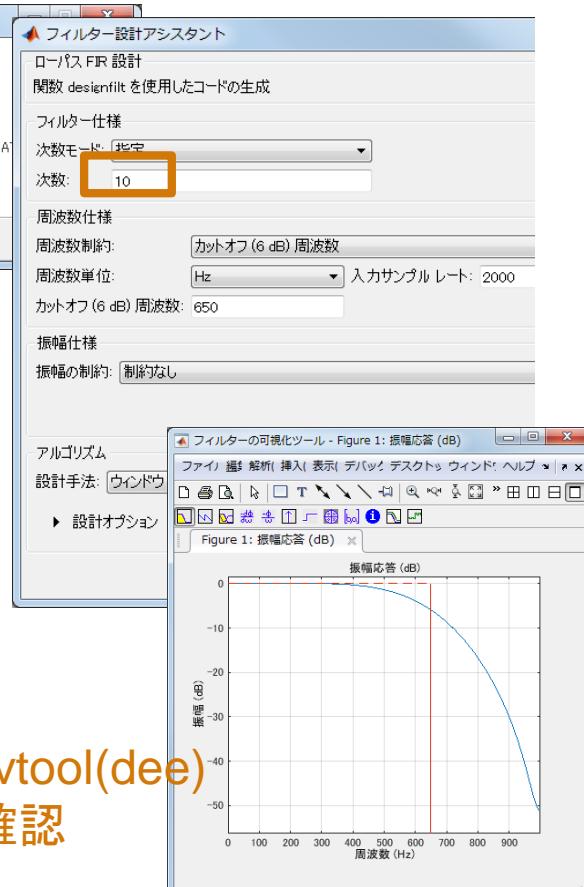
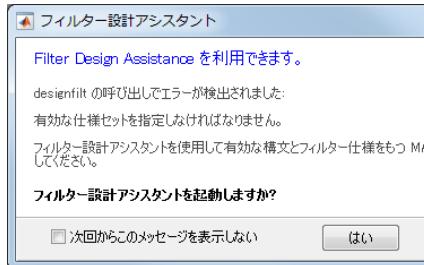
- R2014a新機能(filterbuilderがベース)
- 仕様ベースの設計手法の手順を簡略化  
(従来は2ステップのオブジェクト生成作業が必要)
- フィルタ設計フローのアシスト(足りない引数の候補推定等)

# Q:各種フィルタ設計環境の違いは？(cont'd) R2014a

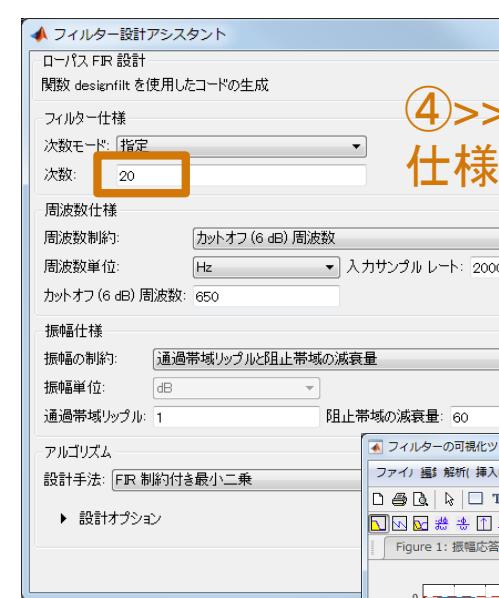
①サンプリング2000[Hz]、カットオフ650[Hz]のローパスFIRフィルタを作成

```
>> dee = designfilt('lowpassfir', 'CutoffFrequency', 650, 'SampleRate', 2000)
```

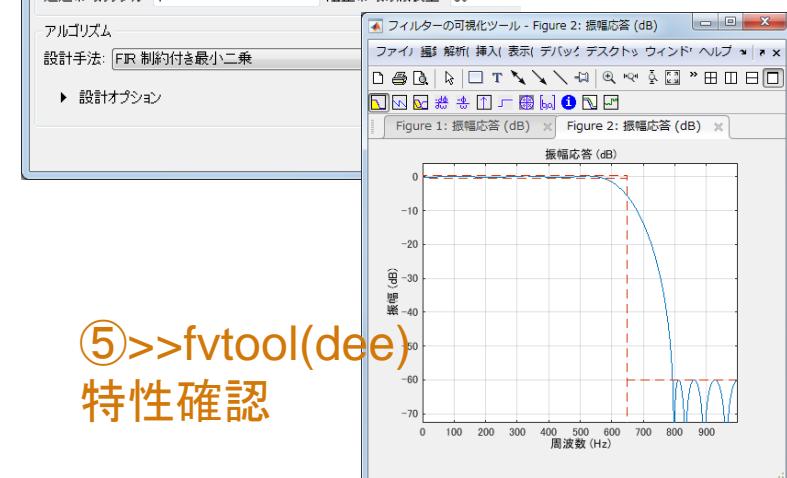
②足りない引数を補完するGUI



③>>fvtool(dee)  
特性確認



④>>designfilt(dee)  
仕様変更



⑤>>fvtool(dee)  
特性確認

# Q: アナログフィルタは設計できますか？

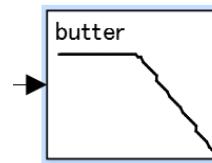
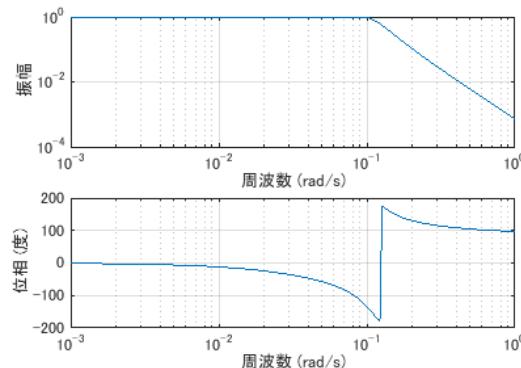
Case1. 仕様が決まっている場合

## Signal Processing Toolbox™

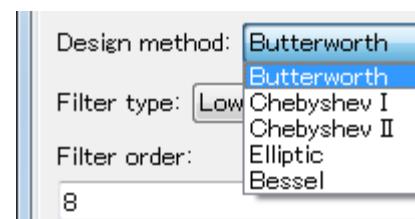
bilinear	アナログ-デジタル フィルター変換用の双一次変換
besselap	ベッセル アナログ ローパス フィルターのプロトタイプ
besself	ベッセル アナログ フィルターの設計
buttap	バタワース フィルターのプロトタイプ
butter	バタワース フィルターの設計
chebiap	チェビシェフ I 型アナログ ローパス フィルターのプロトタイプ
cheb2ap	チェビシェフ II 型アナログ ローパス フィルターのプロトタイプ
cheby1	チェビシェフ I 型フィルターの設計 (通過帯域リップル)
cheby2	チェビシェフ II 型フィルターの設計 (阻止帯域リップル)
ellip	楕円フィルターの設計
ellipap	楕円アナログ ローパス フィルターのプロトタイプ

e.g.)

```
>> [b,a]=cheby1(3,0.5,0.1,'s');
>> freqs(b,a)
```



Analog Filter

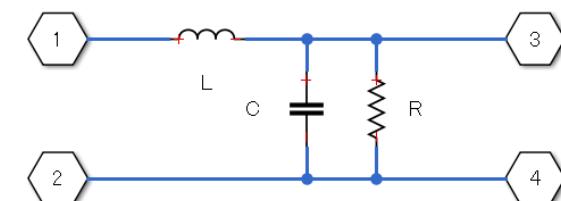


## DSP System Toolbox

Case2. 回路が決まっている場合

## 伝達関数表現 (Simlink)

$$G(s) = \frac{1}{s^2 + \frac{1}{RC}s + \frac{1}{CL}}$$



## 回路表現 (Simscape™)

# Q: アナログフィルタは設計できますか (cont'd) ?

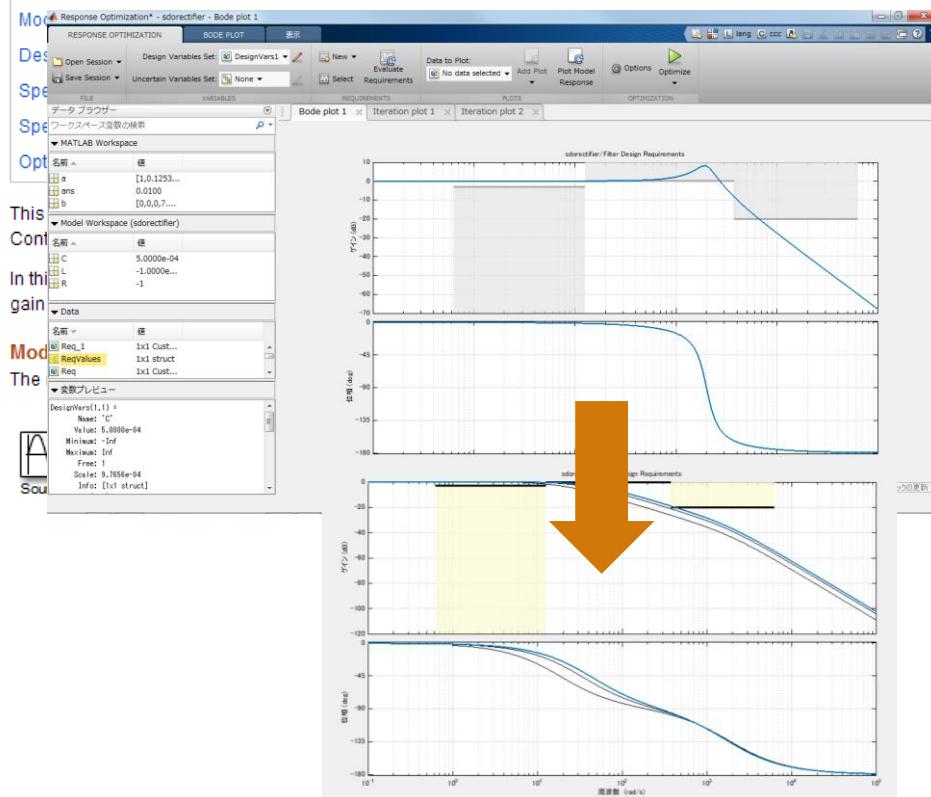
## Simulink Design Optimization™ によるRLCフィルタ定数最適化

Simulink Design Optimization

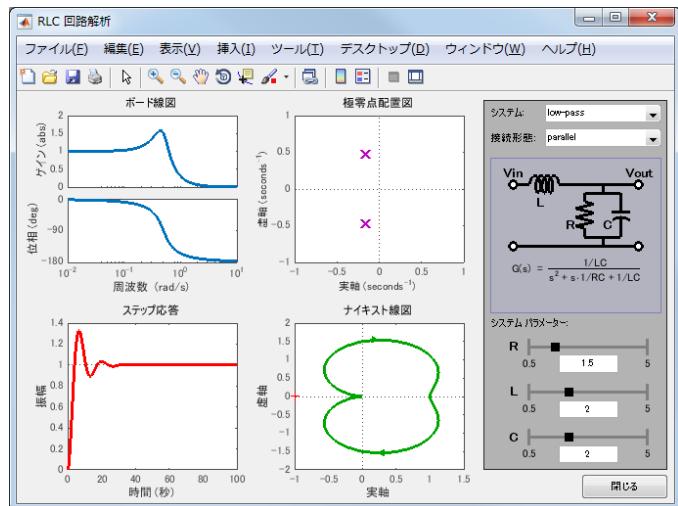
Getting Started with Simulink Design Optimization

### Design Optimization Using Frequency-Domain Check Blocks (GUI)

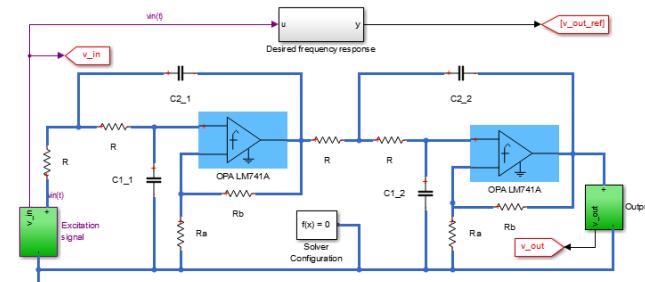
On this page...



## Control System Toolbox™ の RLCフィルタ設計GUI

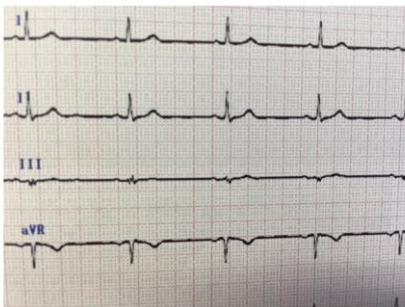


## SimElectronics™による アクティブラインフィルタ

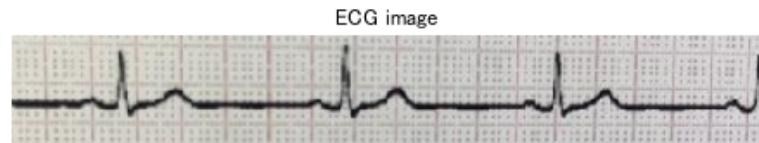


各種手法で実現

# Q:紙のデータにフィルタをかけるには？



JPEG画像データ



...

```
HR2 = imread('HR_s2.jpg');
```

...

```
HR2b = im2bw(HR2, 0.55);
```

バイナリイメージ  
変換

```
for k = 1:515;
```

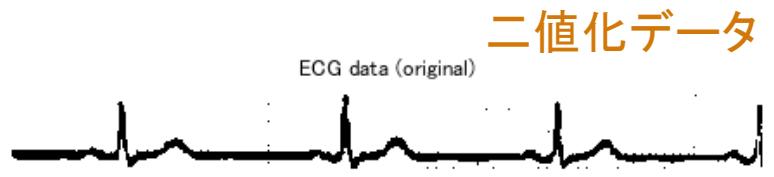
```
    HR2f = find(HR2b(:, k));
```

```
    HR2mean(k) = mean(HR2f);
```

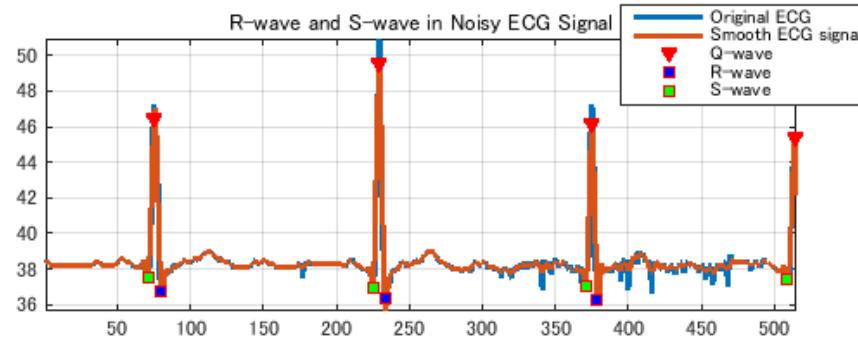
```
end
```

列の重心(データ部の  
インデックス平均値)

Image Processing Toolbox™ の  
関数を応用



二値化データ



画像処理オプションをお試し下さい

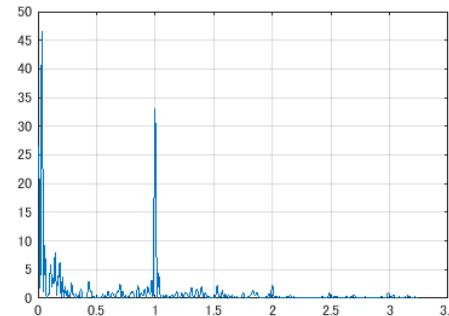
# Q:信号に欠損がある場合や、サンプリングが不等間隔のデータを扱うには？

R2014b

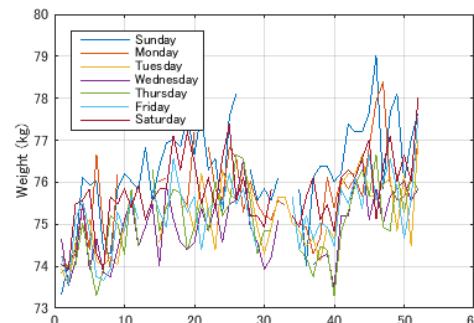
```

9    %% Lomb-Scargle法によるピリオドグラム推定
10   [p,f] = plomb(wgt,7,'normalized');
11   plot(f,p)
12   xlabel('Frequency (week^-1)')
13   grid
14   %% 一周期(7日分)のデータで束ねる
15   wgd = reshape(wgt(1:7*52),[7 52])';
16   plot(wgd)
17   xlabel('Week')
18   ylabel('Weight (kg)')
19   grid
20   q = legend(datestr(datenum(2012,1,1:7),'ddd'));
21   q.Location = 'NorthWest';
22
23   %% スムージングフィルタ処理
24   wgs = sgolayfilt(wgd,3,7);
25   plot(wgs)
26   xlabel('Week')
27   ylabel('Smoothed weight (kg)')
28   grid
29   q = legend(datestr(datenum(2012,1,1:7),'ddd'));

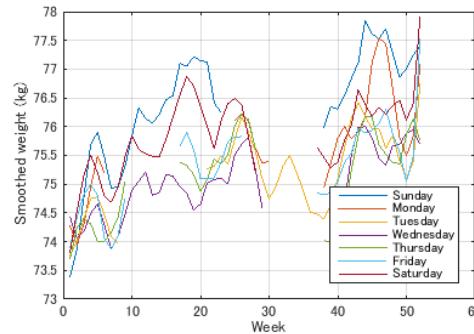
```



plomb関数  
実行結果



時系列データ  
(オリジナル)



時系列データ  
(フィルタ後)

plomb関数をお試し下さい

# Q: テストベンチの効率的な作成方法は？

R2014b

>>testbenchGeneratorExampleApp

The screenshot shows the DSP Testbench Generator app interface. The main window is divided into several sections:

- Inputs:** Shows two selected sources: "In1: Sine Wave" and "In2: White Noise (randn)".
- User Algorithm:** A text input field where the function name "hTestbenchLowpass" is entered.
- Outputs:** Shows one output sink: "Out1: Spectrum Analyzer".
- Code Generation:** A section at the bottom left displays generated MATLAB code, which includes the function definition and a script for setting up the algorithm and running the testbench.
- Plot:** A spectrum analyzer plot showing a low-pass filtered signal. The plot includes frequency markers at 1.00 kHz, 12.69 kHz, and 26.28 kHz, and a noise floor at -12.76 dBm.

Annotations with orange numbers and arrows explain the steps:

- ① 信号源の選択 (Selecting Signal Sources): Points to the list of available sources on the left.
- ② アルゴリズムの選択 (Selecting Algorithm): Points to the User Algorithm input field.
- ③ 解析手法の選択 (Selecting Analysis Methods): Points to the list of available sinks on the right.
- ④ テストベンチ自動生成 (Automatic Testbench Generation): Points to the "Generate MATLAB Code" button in the top menu bar.
- ⑤ 自動生成されたテストベンチと実行結果 (Generated Testbench and Execution Results): Points to the generated MATLAB code and the spectrum analysis plot.

**Appsで自動生成可能**

⑤自動生成されたテストベンチと  
実行結果

# Q: ストリーミング処理を実現するには？

R2014b

- dspdemo.BinaryFileReader
- dspdemo.BinaryFileWriter
- dspdemo.TextFileReader
- dspdemo.TextFileWriter

```

classdef TextFileReader < matlab.System &
    matlab.system.mixin.FiniteSource
properties (Nontunable)
    Filename      = 'tempfile.txt'
    HeaderLines   = 4
end

properties
    DataFormat = '%g'
    Delimiter  = ','
    SamplePerFrame = 1024
end

methods(Access = protected)
    function setupImpl(obj)
        ...
    end
    ...
end

```

e.g.)

```
wobj = dspdemo.TextFileWriter
data = rand(8, 3);
step(wobj, data)
%
step(wobj, 2*data)
```

simplesample.m    tempfile.txt    +

```

1 Data:
2
3
4
5 0.56782164072522112, 0.46939064105820583, 0.60198194140163652
6 0.075854289563063615, 0.011902069501241397, 0.26297128454014429
7 0.053950118666607
8 0.530797553008972
9 0.779167230102011
10 0.934010684229182
11 0.12990620847373015,
12 0.56882366087219272, 0.16564872949978093, 0.22897696871681883
13 1.1356432814504422, 0.93878128211641165, 1.203963882803273
14 0.15170857912612723, 0.023804139002482794, 0.52594256908028858
15 0.1079002373332143
16 1.0615951060179454
17 1.5583344602040223
18 1.868021368458366,
19 0.25981241694746027, 1.0570662710124255, 0.16764275599386513
20 1.1376473217443854, 0.33129745899956187, 0.45795393743363766

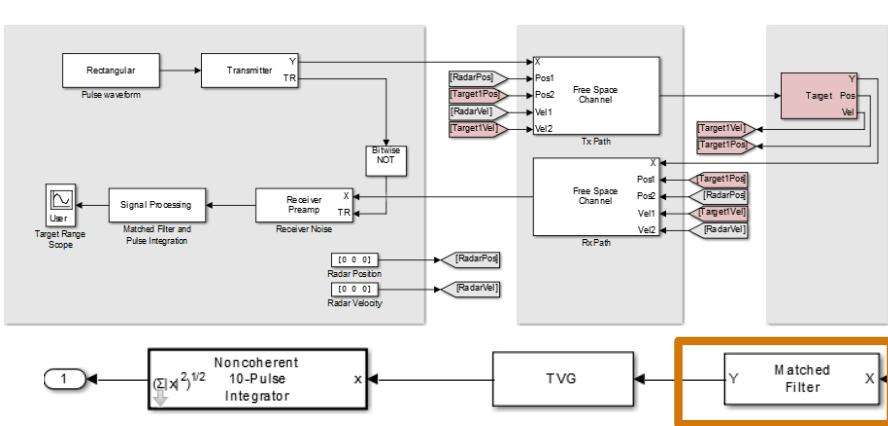
```

生成されたtextファイル  
(stepコマンド1回目)

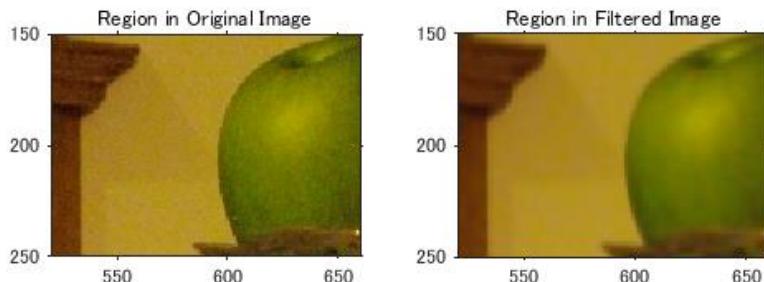
生成されたtextファイル  
(stepコマンド2回目)

カスタマイズ可能

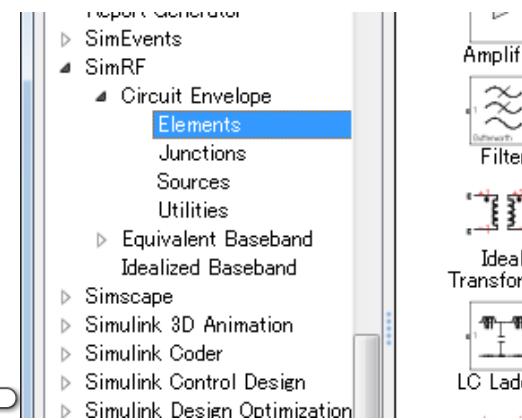
# Q:各種アプリケーションへの応用例は？



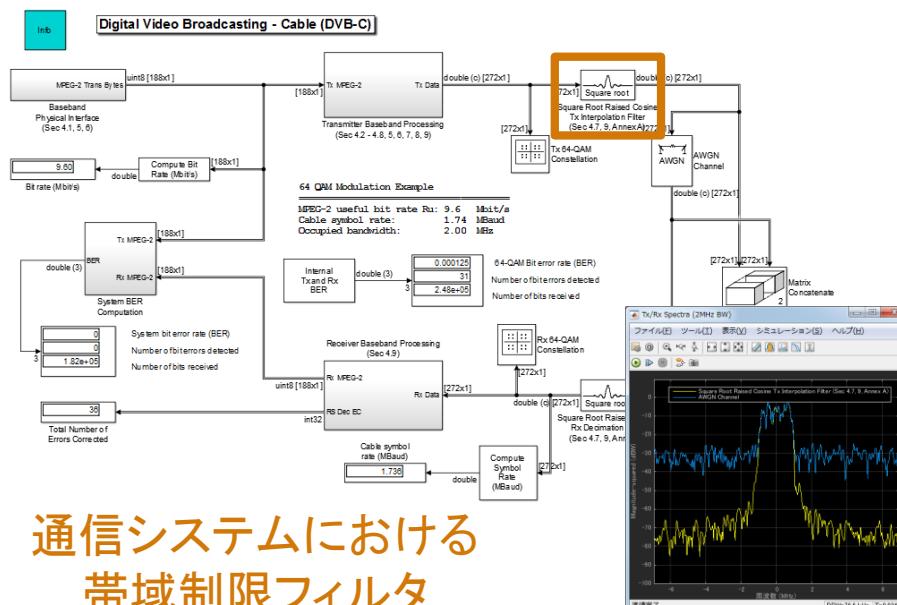
レーダーシステムにおける  
マッチドフィルタ



ガイド付きフィルタによる  
エッジを保持したノイズ除去



高周波用  
フィルタブロック



通信システムにおける  
帯域制限フィルタ

各種アプリケーションで応用

# Agenda

- Section1: フィルタとは？
- Section2: Case study
- Section3: フィルタの実装
- Section4: フィルタ設計FAQ
- まとめ

# まとめ

## ■ Case study

- 種々のケースにおけるフィルタの適用例
- ベーシックな信号処理には**Signal Processing Toolbox**
- 応用的な信号処理、Simulink用には**DSP System Toolbox**

## ■ フィルタの実装

- ARM Cortex-M, Cortex-Aの信号処理系ライブラリ対応
- STMicro社Discoveryへの簡易実装

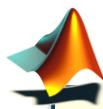
## ■ フィルタ設計FAQ

### ■ 各種新機能

- 不等間隔データ対応
- テストベンチ自動生成等の各種新機能
- ストリーミング処理

帰ったら  
是非お試しください!

# デモブースのご案内



信号処理 / 画像処理・  
コンピュータビジョン ソリューション

