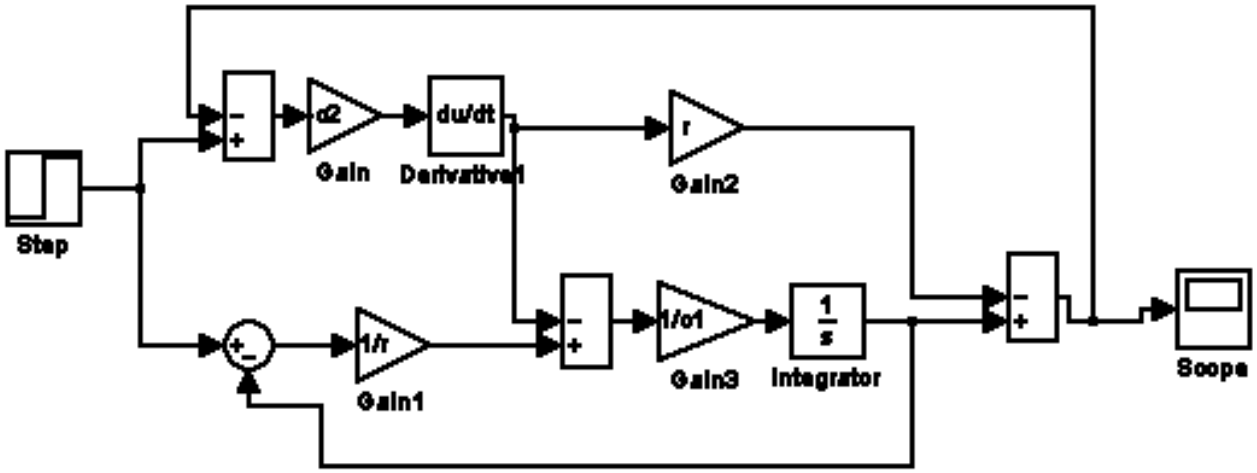
代数ループを含んだ バンドストップフィルタ

～数式変換によるSimulinkモデルの簡単化～



表記の説明

本ドキュメント中で、赤色の文字列は Maple のコマンドです。青字の文字列は Maple によって評価(計算)された出力結果です。

BlockImporter パッケージの読込み

restart

BlockImporter を Maple 上に読み込みます。

with(*BlockImporter*)

[*BuildDE, Import, PrintSummary, SimplifyModel*] (2.1)

パッケージを読み込むと、上記４つのコマンドが利用できるようになります。

コマンド名	説明
Import コマンド	与えられた Simulink モデル(.mdl ファイル)を解析し、数式化します。
PrintSummary コマンド	読み込まれた数式を表示します。通常、Import コマンドは Simulink モデルに相当する数式をモジュール(Maple のプログラム群)として読み込みます。
SimplifyModel コマンド	Importコマンドで読み込んだ数式に簡単化を適用します。
BuildDE コマンド	Importコマンドで読み込み、SimplifyModelコマンドで簡単化された結果から Maple でのシミュレーションに必要な Maple 表現へと変換します。

Simulinkモデルの読込み

代数ループを含んだバンドストップフィルタを Maple に読み込み、数式化します。

```
dataDir := BlockImporter:-DataDirectory( )
```

```
dataDir := "C:\Program Files\Maple 11\toolbox\BlockImporter\data"
```
(3.1)

Simulinkのモデルファイル filterb.mdl を読み込むには、Import コマンドを使用します。

```
model := Import("filterb", path = dataDir, init = "filterb_init", inplace = true)
```

```
model := module( )
    option record;
    export equations, initialeqs, inputvars, notes, outputvars, parameters, procs,
    sourceeqs, statevars;

end module
```
(3.2)

読み込んだモデルが実際にどのような形で数式化されているかを確認するには、PrintSummary コマンドに読み込んだモデルを引き渡します。

```
PrintSummary(model)
```

```
Equations [29]:
```

$$\left[u_{5,1,1}=y_{2,1,1}, u_{13,1,1}=y_{2,1,1}, u_{2,1,1}=y_{3,1,1}, u_{13,2,1}=y_{4,1,1}, u_{12,1,1}=y_{5,1,1}, u_{7,1,1}\right.$$
$$=y_{6,1,1}, u_{11,2,1}=y_{7,1,1}, u_{12,2,1}=y_{7,1,1}, u_{10,2,1}=y_{9,1,1}, u_{11,1,1}=y_{9,1,1}, u_{3,1,1}$$
$$=y_{10,1,1}, u_{4,1,1}=y_{11,1,1}, u_{8,1,1}=y_{12,1,1}, u_{10,1,1}=y_{12,1,1}, u_{6,1,1}=y_{13,1,1}, y_{2,1,1}$$
$$=\mathrm{D}\left(x_{2,1}\right), x_{2,1}=u_{2,1,1}, y_{3,1,1}=-K_{0,\text{"c2"}}\,u_{3,1,1}, y_{4,1,1}=\frac{u_{4,1,1}}{K_{0,\text{"r"}}}, y_{5,1,1}$$
$$=K_{0,\text{"r"}}\,u_{5,1,1}, y_{6,1,1}=\frac{u_{6,1,1}}{K_{0,\text{"c1"}}}, \mathrm{D}\left(x_{7,1}\right)=u_{7,1,1}, y_{7,1,1}=x_{7,1}, Sink_{Scope,8,1,1}$$
$$=u_{8,1,1}, y_{9,1,1}=Source_{Step,9,1}, y_{10,1,1}=-u_{10,1,1}+u_{10,2,1}, y_{11,1,1}=u_{11,1,1}$$
$$\left.-u_{11,2,1}, y_{12,1,1}=-u_{12,1,1}+u_{12,2,1}, y_{13,1,1}=-u_{13,1,1}+u_{13,2,1}\right]$$

```
State Variables [2]:
```

$$\left[x_{2,1}, x_{7,1}\right]$$

```
Initial Equations [2]:
```

$$\left[x_{2,1}(0)=0, x_{7,1}(0)=0\right]$$

```
Source Equations [1]:
```

$$\left[Source_{Step,9,1}=\begin{cases}0 & t<1\\ 1 & otherwise\end{cases}\right]$$

```
Input Variables [1]:
```

$$\left[Source_{Step,9,1}\right]$$

```
Output Variables [1]:
```

$$\left[Sink_{Scope,8,1,1}\right]$$

```
Parameters [3]:
```

$$\left[K_{0,\text{"r"}}=1000., K_{0,\text{"c1"}}=0.00100000000000000002, K_{0,\text{"c2"}}\right.$$
$$\left.=0.00100000000000000002\right]$$

(3.3)

Maple に読み込まれた結果は以下の通りです。

オブジェクト	説明
Equations [29]	29個の連立方程式が生成されたことを意味しています。
State Variables [2]	システムの状態変数は 2 個であったことを意味しています。
Initial Equations [2]	初期条件は 2 つの式からなっていることを意味しています。
Source Equations [1]	入力信号の方程式が 1 種類であったことを意味しています。
Input Variables [1]	入力信号の方程式で使われた変数が 1 種類であったことを意味しています。
Output Variables [1]	出力信号の方程式で使われた変数が 1 種類であったことを意味しています。
Parameters [3]	方程式系の係数パラメータが 3 種類あることを意味しています。

数式モデルの簡単化

Maple に読み込んだ数式モデル（連立方程式系）の中で、冗長な変数関係は SimplifyModel コマンドを適用することで簡単化されます。

```
simplifiedModel := SimplifyModel(model)
```

```
simplifiedModel := module( )                                    (4.1)
    option record;
    export equations, initialeqs, inputvars, notes, outputvars, parameters, procs,
    sourceeqs, statevars;

end module
```

```
PrintSummary(simplifiedModel)
```

Equations [3]:

$$\left[x_{2,1} = -K_{0,\text{"c2"}}\left(-Sink_{Scope,8,1,1} + Source_{Step,9,1}\right), \mathrm{D}\left(x_{7,1}\right) = -\frac{K_{0,\text{"r"}}\,\mathrm{D}\left(x_{2,1}\right) - Source_{Step,9,1} + x_{7,1}}{K_{0,\text{"r"}}\,K_{0,\text{"c1"}}}, Sink_{Scope,8,1,1} = -K_{0,\text{"r"}}\,\mathrm{D}\left(x_{2,1}\right) + x_{7,1}\right]$$

State Variables [2]:

$$\left[x_{2,1}, x_{7,1}\right]$$

Initial Equations [2]:

$$\left[x_{2,1}(0) = 0, x_{7,1}(0) = 0\right]$$

Source Equations [1]:

$$\left[Source_{Step,9,1} = \begin{cases} 0 & t < 1 \\ 1 & otherwise \end{cases}\right]$$

Input Variables [1]:

$$[Source_{Step,\,9,\,1}]$$

Output Variables [1]:

$$[Sink_{Scope,\,8,\,1,\,1}]$$

Parameters [3]:

$$[K_{0,\,"r"} = 1000.,\ K_{0,\,"c1"} = 0.00100000000000000002,\ K_{0,\,"c2"} = 0.00100000000000000002] \tag{4.2}$$

簡単化された結果、29 個の方程式からなっていたシステムモデルは 3 つの方程式からなるモデルへと変換されています。

システムの簡易シミュレーション

簡単化された数式モデルから微分方程式系を作るために、BuildDE コマンドを使用します。BuildDE コマンドは、読み込んだモデルのオブジェクトから Maple 上でのシミュレーションに必要となる、[微分方程式系, パラメータに値を代入した微分方程式系, 初期条件, 入力の関数名, 出力(変位)の関数名] のリストを作り出します。

$sys := BuildDE(simplifiedModel)$

$$sys := \left[x_{2,\,1}(t) = -K_{0,\,"c2"}\left(-Sink_{Scope,\,8,\,1,\,1}(t) + Source_{Step,\,9,\,1}(t)\right),\ \dot{x}_{7,\,1}(t) = -\frac{K_{0,\,"r"}\,\dot{x}_{2,\,1}(t) - Source_{Step,\,9,\,1}(t) + x_{7,\,1}(t)}{K_{0,\,"r"}\,K_{0,\,"c1"}},\ Sink_{Scope,\,8,\,1,\,1}(t) = -K_{0,\,"r"}\,\dot{x}_{2,\,1}(t) + x_{7,\,1}(t)\right],\ \left[x_{2,\,1}(t) = 0.00100000000000000002\,Sink_{Scope,\,8,\,1,\,1}(t) - 0.00100000000000000002\,Source_{Step,\,9,\,1}(t),\ \dot{x}_{7,\,1}(t) = -1000.000000\,\dot{x}_{2,\,1}(t) + Source_{Step,\,9,\,1}(t) - x_{7,\,1}(t),\ Sink_{Scope,\,8,\,1,\,1}(t) = -1000.\,\dot{x}_{2,\,1}(t) + x_{7,\,1}(t)\right],\ \left[x_{2,\,1}(0) = 0,\ x_{7,\,1}(0) = 0\right],\ \left[Source_{Step,\,9,\,1}(t) = \begin{cases} 0 & t < 1 \\ 1 & otherwise \end{cases}\right],\ \left[Sink_{Scope,\,8,\,1,\,1}(t)\right] \tag{5.1}$$

リストの各要素には以下の記述方法でアクセスできます。

記号的な微分方程式系：

$sys[1] =$

$$\left[x_{2,\,1}(t) = -K_{0,\,"c2"}\left(-Sink_{Scope,\,8,\,1,\,1}(t) + Source_{Step,\,9,\,1}(t)\right),\ \dot{x}_{7,\,1}(t) = -\frac{K_{0,\,"r"}\,\dot{x}_{2,\,1}(t) - Source_{Step,\,9,\,1}(t) + x_{7,\,1}(t)}{K_{0,\,"r"}\,K_{0,\,"c1"}},\ Sink_{Scope,\,8,\,1,\,1}(t) = -K_{0,\,"r"}\,\dot{x}_{2,\,1}(t) + x_{7,\,1}(t)\right]$$

パラメータに値を代入した微分方程式系：

$sys[2] =$

$$[x_{2,\,1}(t) = 0.00100000000000000002\,Sink_{Scope,\,8,\,1,\,1}(t) - 0.00100000000000000002\,Source_{Step,\,9,\,1}(t),\ \dot{x}_{7,\,1}(t) = -1000.000000\,\dot{x}_{2,\,1}(t) + Source_{Step,\,9,\,1}(t) - x_{7,\,1}(t),\ Sink_{Scope,\,8,\,1,\,1}(t) = -1000.\,\dot{x}_{2,\,1}(t) + x_{7,\,1}(t)]$$

初期条件：

$sys[3] = [x_{2,\,1}(0) = 0,\ x_{7,\,1}(0) = 0]$

入力の関数名：

$$sys[4] = \left[Source_{Step,\,9,\,1}(t) = \begin{cases} 0 & t < 1 \\ 1 & otherwise \end{cases}\right]$$

出力(変位)の関数名：

$sys[5] = [Sink_{Scope,\,8,\,1,\,1}(t)]$

$sysDE := eval([op(sys[2]), op(sys[3])], sys[4])$

$$sysDE := \left[x_{2,\,1}(t) = 0.00100000000000000002\, Sink_{Scope,\,8,\,1,\,1}(t) \right. \tag{5.2}$$

$$- 0.00100000000000000002 \begin{cases} 0 & t < 1 \\ 1 & otherwise \end{cases}, \dot{x}_{7,\,1}(t) = -1000.000000\, \dot{x}_{2,\,1}(t)$$

$$+ \begin{cases} 0 & t < 1 \\ 1 & otherwise \end{cases} - x_{7,\,1}(t), Sink_{Scope,\,8,\,1,\,1}(t) = -1000.\, \dot{x}_{2,\,1}(t) + x_{7,\,1}(t),$$

$$\left. x_{2,\,1}(0) = 0, x_{7,\,1}(0) = 0 \right]$$

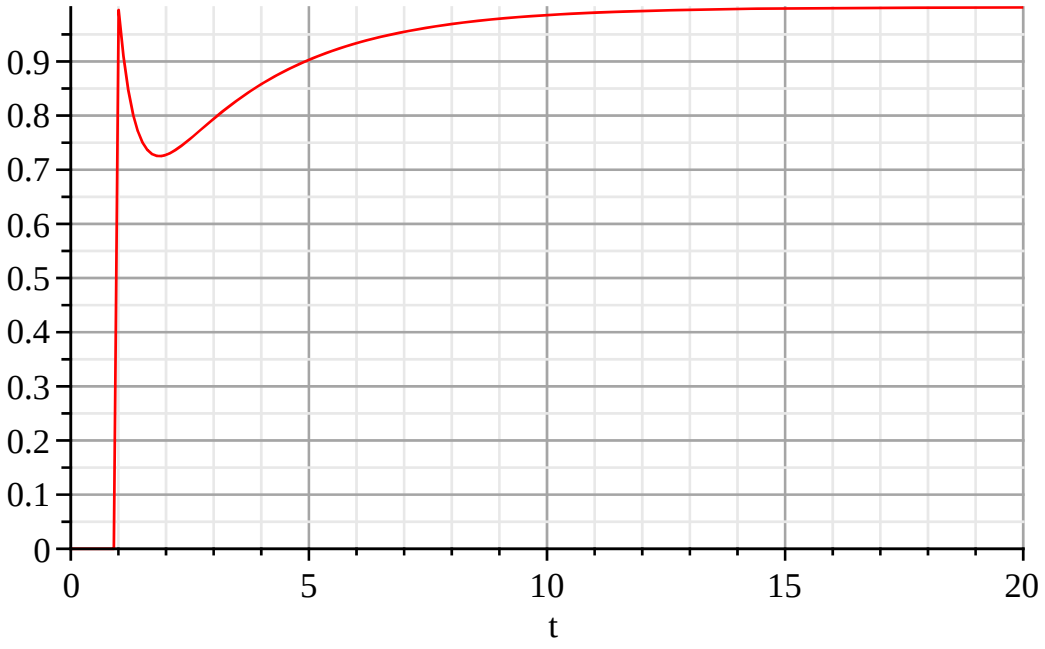
方程式系を用意できたので、Maple の dsolve コマンドで解を数値的に求めてみます。

$solution := dsolve(sysDE, numeric)$

$$solution := \mathbf{proc}(x\_rkf45\_dae) \ \ldots \ \mathbf{end\ proc} \tag{5.3}$$

dsolve を数値解の算出のために用いると、Maple は解関数を構成する中間式オブジェクト(補間式)を返してきます。得られた結果からシミュレーションの結果を描画してみます。

$plots[odeplot](solution, [t, op(sys[5])], 0..20, numpoints = 200, gridlines = true)$



BlockBuilder を用いたブロック生成

Maple 上に読み込み数式化されたシステムを、再度 Simulink ブロックへと変換します。
まず、BlockBuilder for Simulink のパッケージを読み込みます。

```
with(BlockBuilder)
```

```
[BodePlot, CharacteristicPolynomial, Chirp, Coefficients, ControllabilityMatrix,
    Controllable, DiffEquation, DiscretePlot, ExecuteMCode, FrequencyResponse,
    GainMargin, GenerateSFunction, GenerateSimulink, Grammians,
    ImpulseResponse, ImpulseResponsePlot, Interactive, IsSystem, MagnitudePlot,
    NewSystem, NyquistPlot, ObservabilityMatrix, Observable, PhaseMargin,
    PhasePlot, PreviewCode, PrintSystem, Ramp, ResponsePlot, RootContourPlot,
    RootLocusPlot, RouthTable, SSModelReduction, SSTransformation, SaveCode,
    Simulate, Sinc, Sine, Square, StateSpace, Step, System, SystemOptions, ToDiscrete,
    TransferFunction, Triangle, Verify, ZeroPoleGain, ZeroPolePlot]
```
(6.1)

BlockBuilder を用いて、システムモデルを Block にするために、まずシステムオブジェクトを定義します。

```
deq := sys[1]
```

$$deq := \left[x_{2,1}(t) = -K_{0,"c2"} \left(-Sink_{Scope,8,1,1}(t) + Source_{Step,9,1}(t) \right), \dot{x}_{7,1}(t) = -\frac{K_{0,"r"}\, \dot{x}_{2,1}(t) - Source_{Step,9,1}(t) + x_{7,1}(t)}{K_{0,"r"}\, K_{0,"c1"}}, Sink_{Scope,8,1,1}(t) = -K_{0,"r"}\, \dot{x}_{2,1}(t) + x_{7,1}(t) \right] \quad (6.2)$$

```
infunc := map(lhs, sys[4])
```

$$infunc := \left[Source_{Step,9,1}(t) \right] \quad (6.3)$$

```
outfunc := sys[5]
```

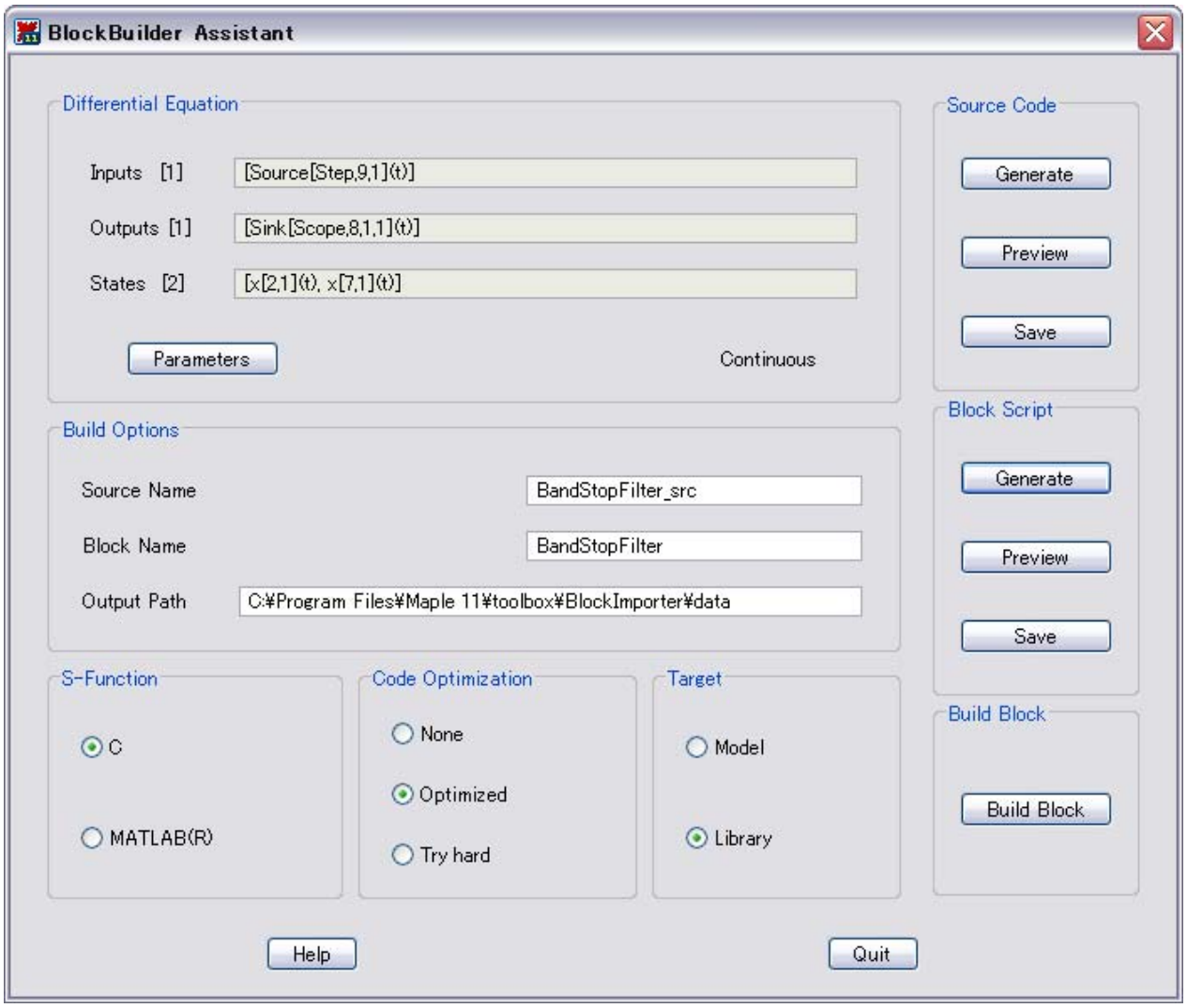
$$outfunc := \left[Sink_{Scope,8,1,1}(t) \right] \quad (6.4)$$

```
sysobj := DiffEquation(deq, infunc, outfunc)
```

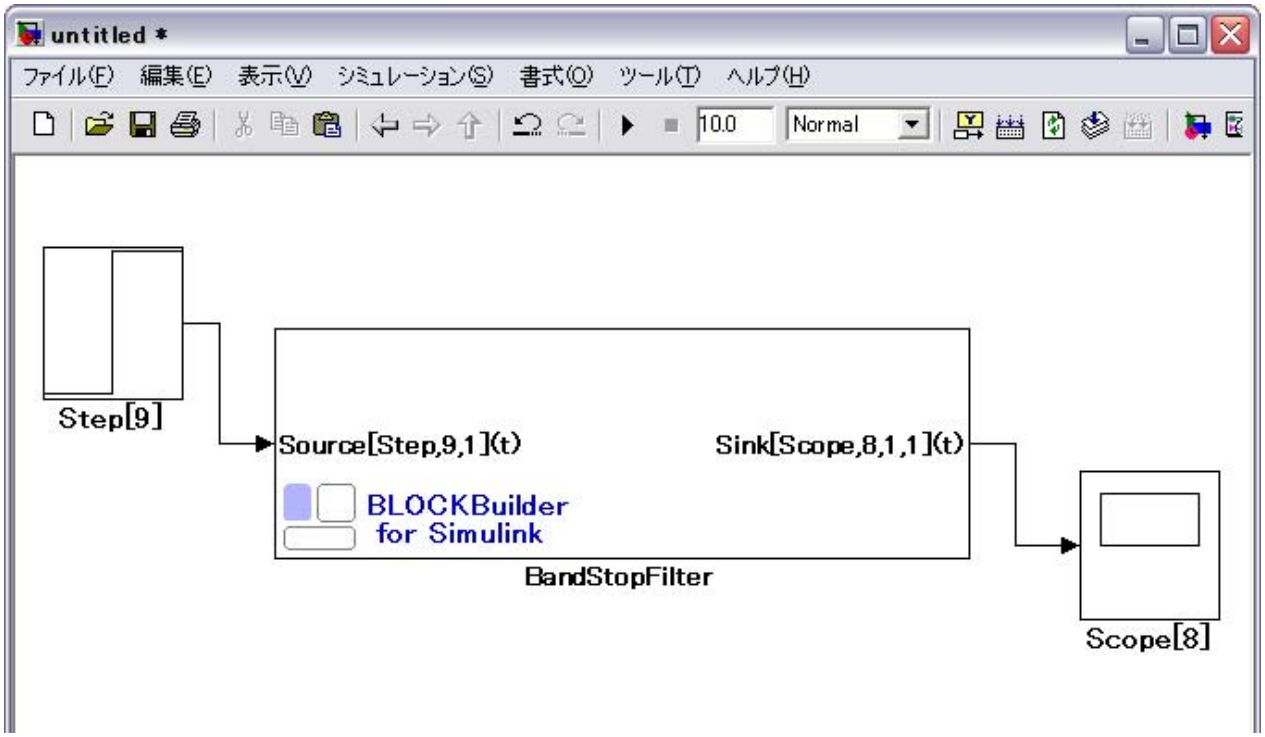
$$sysobj := \left[\begin{array}{c} \textbf{Diff. Equation} \\ \text{continuous} \\ \text{1 output(s); 1 input(s)} \\ \text{inputvariable} = \left[Source_{Step,9,1}(t) \right] \\ \text{outputvariable} = \left[Sink_{Scope,8,1,1}(t) \right] \end{array} \right] \quad (6.5)$$

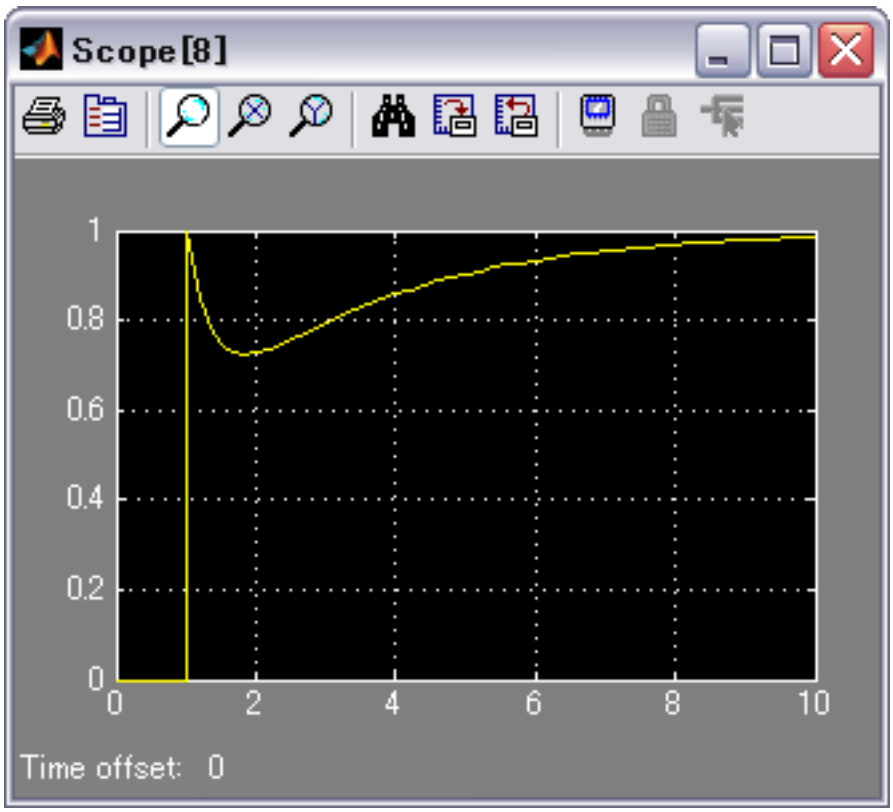
システムオブジェクトが定義できたら、BlockBuilder for Simulink の Interactive コマンドを使って対話的に Simulink Block を作ります。

```
Interactive(sysobj, parameters = simplifiedModel:-parameters, blockname = "BandStopFilter",
    sourcename = "BandStopFilter_src")
```



ソースの生成、ブロックのビルドを実施すると、Simulink ブロックが完成します。
以下は、生成されたブロックを用いたダイアグラムとシミュレーションを行った結果です。





まとめ

BlockImporter for Simulink と BlockBuilder for Simulink、そして数式処理ソフトウェア Maple を用いることで、複雑化する Simulink モデルを一旦数式に落とし込むことで簡単化し、元々のモデルと等価な Simulink ブロックを生成しました。
この手法を用いることで、以下のようなシミュレーションや設計を実現することが可能となります；

1 数式ベースのモデル表現により、数学的な意味を考察する。(例：変数の依存関係の考察、等)

2 代数ループといった Simulink の問題点を数式変換により克服する。

3 数式を C 言語形式の Simulink ブロック化することで高速なシミュレーションモデルを生成できる。