

初心者向けテキスト：制御

## MATLAB を利用した制御系設計

### --- 基礎編 ---

発行：2005年5月第三版作成  
2003年7月第二版作成，1997年初版作成  
著者：加納 学  
京都大学大学院工学研究科化学工学専攻  
連絡先：  
<http://www-pse.cheme.kyoto-u.ac.jp/~kano/>

Copyright (C) 1997-2005 by Manabu KANO. All rights reserved.

本資料の内容の一部あるいは全部を無断で転載，複製，複写することを禁じます。  
また，本資料の間違いなどによって生じた不利益などに対して，著者は一切責任を負いません。  
MATLAB と SIMULINK は MATH WORKS 社の登録商標です。

## 第 1 章 MATLAB を使おう

### 1.1. ”MATLAB” とは？

MATLAB は米国 The Math Works 社のソフトウェアであり、

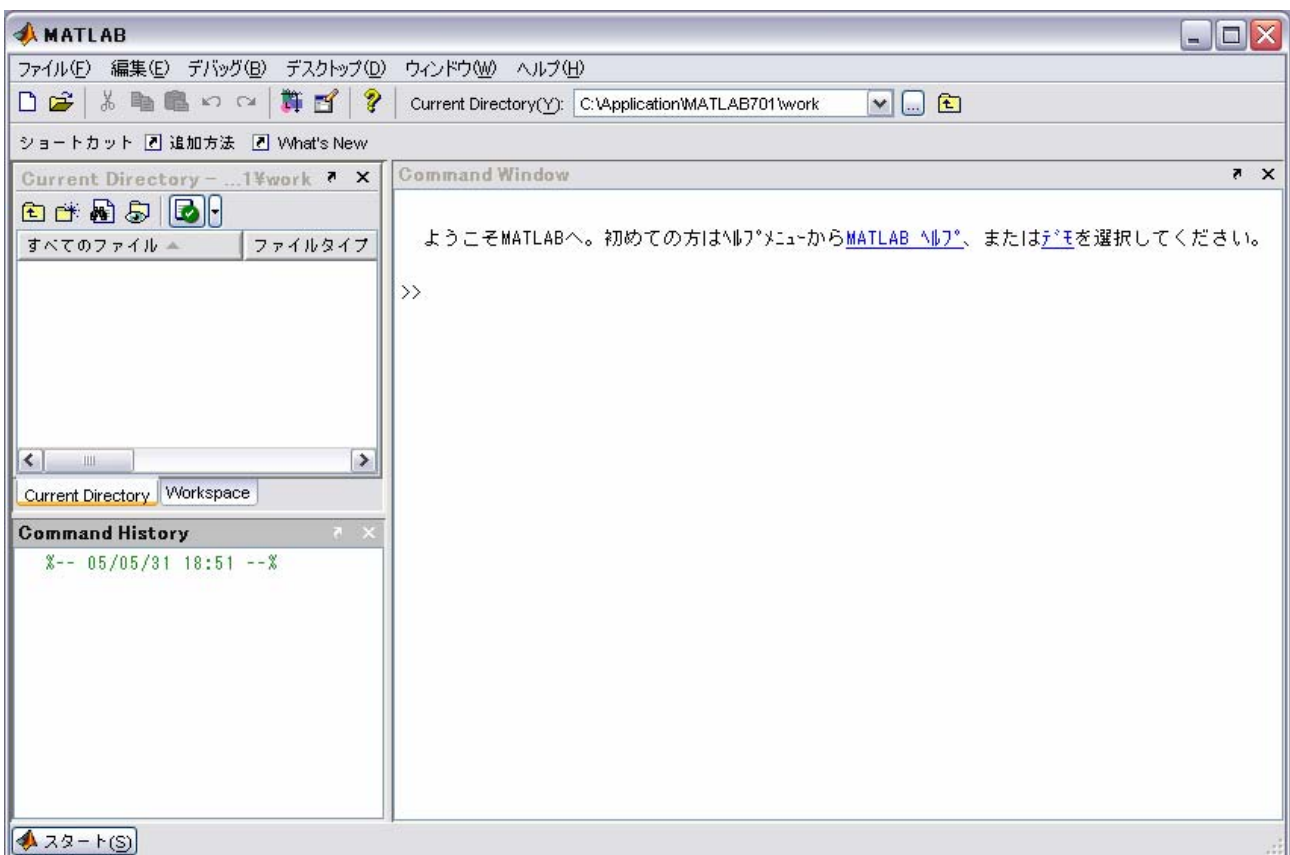
”High-Performance Numeric Computation and Visualization Software”

という位置付けで販売されています。その特徴として多彩な行列演算機能やグラフィック機能などが挙げられますが、これらの特徴を支えているのは用意されている豊富な関数群 (Function-file) です。

Function-file は MATLAB 本体に含まれているものの他にも数多く用意されており、それぞれ利用目的別に分類されて、Control Toolbox や Optimization Toolbox などのように Toolbox として販売されています。例えば Optimization Toolbox には、非線形最小二乗法や二次計画法など全 13 種類のプログラム (Function-file) が用意されています。Toolbox としては他に、System Identification Toolbox や Robust Control Toolbox などがあり、現在でも続々と新しい Toolbox が追加されています。

### 1.2. MATLAB のデモを実行しよう

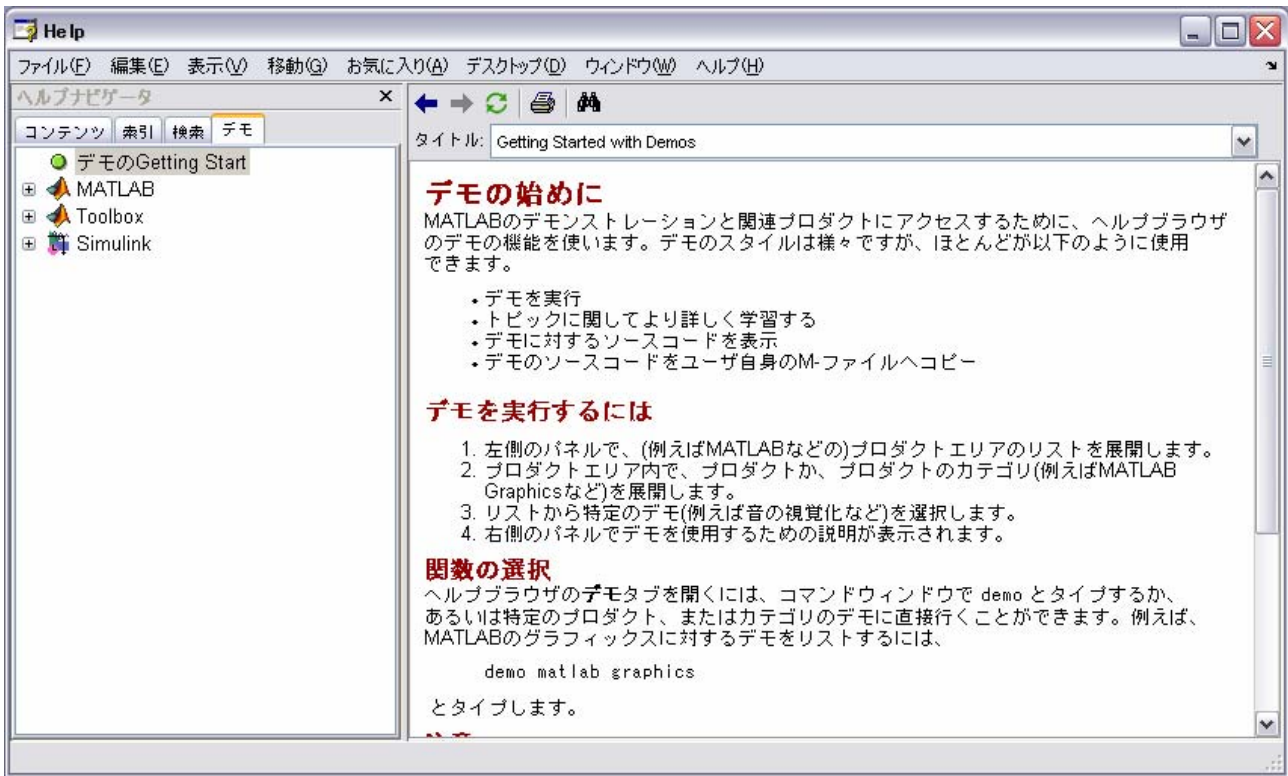
とにかく MATLAB を起動してみましょう。起動方法は利用する計算機の設定に依存しますので、わからないときはシステム管理者に聞いてください。起動時に開くウィンドウが MATLAB の Command Window となり、ここに種々のコマンドを入力することになります。



MATLAB の使用方法の説明に移る前に、用意されているデモを実行してみましょう。

```
>> demo
```

と入力して下さい。Help window が表示されたら、デモタブにある MATLAB, Toolbox, Simulink に用意されている各種デモを実行してみてください。



### 1.3. コマンドラインからの入力

この節では、簡単な行列の入力・演算方法やグラフィック機能などを紹介します。ここで説明する内容は MATLAB を利用する上での基本中の基本となります。MATLAB の優れた特徴の一つはその多彩な行列演算機能にあります。ここでは、それらの機能をコマンドラインからの入力を通して利用する方法について説明します。

“論よりラン (Run) ”。以下の順で、コマンドラインに入力して下さい。

```
スカラー          >> x=3
列ベクトル        >> y=[1; 2; 3]
行ベクトル        >> z=[4 5 6]
ベクトルの積      >> y*z
                  >> z*y
ベクトルの和      >> [1 0 3]+[0 2 0]
```

行列                    >> A=[8 1 6; 3 5 7; 4 9 2]

このように，“[ ]” が行列（ベクトルやスカラーを含む）を，“;” が行の区切りを表します。また、行列の各要素は“スペース”で区別されます。

行列                    >> A=[8 1 6; 3 5 7; 4 9 2];

※同じコマンドを入力する場合には、カーソルキーを用いると便利です。

“↑”で過去に使用したコマンドを選択できます。

2つのコマンドの違いがわかりますか。そう、結果が表示されるか否かの違いです。プログラムを書くようになると、計算結果を全て画面に表示させるのは望ましくありません。結果を表示させたくない場合には、コマンドの最後に “;” を付けます。では、続けましょう。

転置                    >> B=A'

行列の和                >> A+B

行列の積                >> A\*B

要素の積                >> A.\*B

逆行列                  >> inv(A)

>> A\*inv(A)

固有値                  >> eig(A)

行列演算について理解できましたか。では、さらに続けましょう。行列演算に関連して、ある行や列など特定の要素を抜き出したいことがあります。

>> C=[1 2 3; 4 5 6; 7 8 9]

1行2列の抽出            >> C(1,2)

全行2列の抽出            >> C(:,2)

3行全列の抽出            >> C(3,:)

特定部分の抽出            >> C(1:2,2:3)

このように簡単にできてしまいます。他にも

各列の和                >> sum(C)

各列の平均                >> mean(C)

各列の標準偏差            >> std(C)

といった関数も用意されています。この他にも数多くの関数が用意されていますが、詳細はマニュアルを参照して下さい。さて、MATLAB で使用した変数に関する情報はワークスペースに保存されています。この情報を見るためには

```
変数のチェック      >> who
                   >> whos
```

というコマンドを使います。この2つのコマンドの違いは表示させる情報量の差にあります。必要に応じて使い分けるといいでしょう。現在ワークスペースに保存されている情報を全て消去するためのコマンドは、

```
クリア              >> clear
```

です。前述のコマンド “whos” を用いて、情報が消去されていることを確認しましょう。

ここまでは行列の入力と演算を中心に説明してきましたが、MATLAB の多彩なグラフィック機能についても触れておかなければなりません。以下に簡単な作図例を示します。

```
>> x=-10:2:10      ※ -10 から 10 まで 2 間隔で。
>> y=x.^2-5
>> plot(x,y)
>> plot(x,y,'o')  ※ 記号として “x”, “-”, “:” などがある。
>> bar(x,y)
```

ここで用いたベクトルの作成方法は有用ですので、是非覚えて下さい。

それでは最後に、データの保存と読み込み方法を説明します。現在ワークスペースに存在している変数を確認して下さい。いくつかの変数がありますね。それでは、この中のベクトル  $x$  と  $y$  をファイル “data” に保存しましょう。次のコマンドを実行して下さい。

```
>> save data x y
```

ファイル “data.mat” ができていることを確認して下さい。拡張子 (.mat) はデータファイルであることを示すものであり、自動的に付けられます。次に、保存したデータをファイルから読み込んでみましょう。

```
>> clear
>> whos
>> load data
>> whos
```

確かに、ベクトル  $x$  と  $y$  がワークスペースに読み込まれていますね。

## 1.4. M-file の作成 (プログラミング)

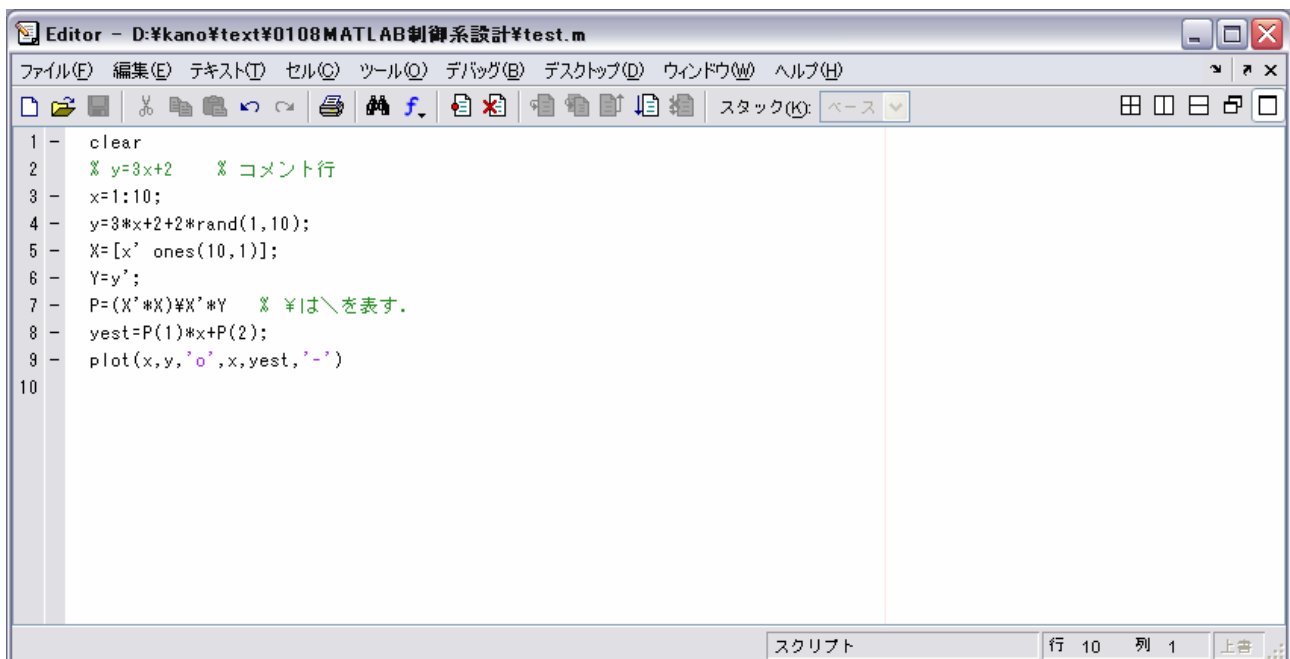
前節の演習で MATLAB のイメージはつかんでいただけたと思います。ここではさらに進んで、簡単なプログラムを作成してみましよう。プログラムの内容は最小二乗法を用いた未知パラメータの推定です。

MATLAB では、プログラムの書かれたファイルのことを “M-file” といい、filename.m というように拡張子 (.m) を付けて認識します。それでは、次の M-file を作成し、”test.m” というファイル名で保存して下さい。メニューの “ファイル” – “新規作成” – “M-ファイル” を選択すれば、テキストエディタが起動します。

```

行
1.      clear
2.      % y=3x+2          % コメント行
3.      x=1:10;
4.      y=3*x+2+2*rand(1,10);
5.      X=[x' ones(10,1)];
6.      Y=y';
7.      P=(X'*X)\X'*Y      % \は\を表す。
8.      yest=P(1)*x+P(2);
9.      plot(x,y,'o',x,yest,'-')

```



```

Editor - D:\kano\text\0108MATLAB制御系設計\test.m
ファイル(F) 編集(E) テキスト(T) セル(C) ツール(O) デバッグ(B) デスクトップ(D) ウィンドウ(W) ヘルプ(H)
1 - clear
2 - % y=3x+2 % コメント行
3 - x=1:10;
4 - y=3*x+2+2*rand(1,10);
5 - X=[x' ones(10,1)];
6 - Y=y';
7 - P=(X'*X)\X'*Y % \は\を表す。
8 - yest=P(1)*x+P(2);
9 - plot(x,y,'o',x,yest,'-')
10

```

プログラムの説明をしておきましょう。このプログラムは、10 点の  $(x, y)$  データを作成し、そのデータを (二乗誤差を最小にするという意味で) 最も良く表す直線を求めるためのものです。x は 1 から 10 までの整数であり、y は  $y=3x+2$  という関係を満たすものとします。ただし、ここでは関係式にランダムなノイズ ( $2*\text{rand}$ ) をのせています。別の言い方をすると、このプログラムの目的は、

$$y = \{ P(1) x + P(2) \}$$

の二乗和を最小にするパラメータ  $P(1), P(2)$  を求めることになります。このようなパラメータは、最小二乗法を用いて（正規方程式を解くことにより）簡単に求めることができます。最小二乗法を御存知ない方は勉強しておかれると良いでしょう。ここでは説明を省略します。

#### プログラムの説明

行

1. ワークスペースのクリア
- 3, 4. ベクトルの作成
- 5, 6. 正規方程式の準備
7. 解の計算
8. 推定値の計算
9. 結果の表示

それではこのプログラムを実行してみましょう。

```
>> test
```

と入力して下さい。結果が表示されますね。

### 1.5. Control System Toolbox の利用

ここでは、制御問題を取り扱う際に非常に強力な武器となる Control System Toolbox の利用方法について述べます。様々な関数を駆使する前に、MATLAB 上での伝達関数の取り扱いについて知っておく必要があります。Command Window に以下のように入力して下さい。

```
>> num=5
>> den=[10 1]
>> printsys(num, den)
>> num=[5 0]
>> printsys(num, den)
```

MATLAB では、この例のように伝達関数の分子分母をベクトルを用いて表します。

伝達関数のインパルス応答およびステップ応答をプロットしてみましょう。

```
>> num=[2]
```

```
>> den=[4 2 1]
>> impulse(num, den)
>> step(num, den)
```

もちろん、ボード線図やベクトル線図も描くことができます。

```
>> bode(num, den)
>> nyquist(num, den)
```

#### [演習問題 1]

伝達関数で表現されるシステムの安定性は、その特性方程式の根（極）を調べることによって判断できる。次の伝達関数

$$P_1(s) = \frac{2s+1}{4s^2+2s+1}$$

が安定であるかどうかを調べなさい。なお、多項式の根は関数 “roots” を用いて求めることができる。この関数の使用方法について知りたい場合は、

```
>> help roots
```

と入力しなさい。

次に、この伝達関数のステップ応答をプロットしなさい。

#### [演習問題 2]

伝達関数で表現されるシステムについて、その分子多項式の根（ゼロ）が 1 つ以上複素平面上で右半面に存在するとき、そのシステムは非最小位相系（NMP）であるという。次の伝達関数

$$P_2(s) = \frac{-2s+1}{4s^2+2s+1}$$

のステップ応答をプロットし、演習問題 1 の伝達関数  $P_1(s)$  のステップ応答と比較しなさい。



## 第2章 SIMULINK を使おう

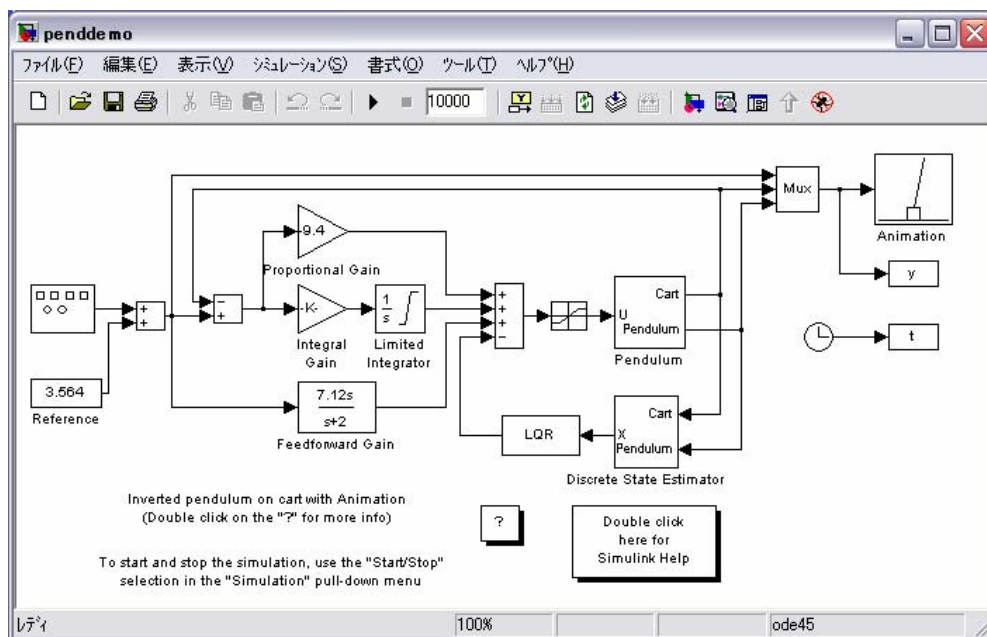
### 2.1. "SIMULINK" とは?

SIMULINK はダイナミックシミュレーションを行うためのソフトウェアです。このソフトの特徴は、ブロック線図を描くことによってシミュレーションプログラムを作成できる点にあります。このため、難しいプログラムを自分で作成する必要はありません。また、ブロックとして MATLAB の関数を利用できるため、複雑なシミュレーションも容易に実行することができます。特に制御系のシミュレーションを行う際には大変な威力を発揮します。

論よりラン。まずは、デモを見ることにしましょう。第1章で説明したように、

```
>> demo
```

と入力して、用意されているデモを実行してみましよう。Help window が開いたら、“Simulink” - “一般的なアプリケーション” - “倒立振り子とアニメーション” をクリックして下さい。右側にその説明が現れます。その中の“Open this model” をクリックして下さい。すると、“penddemo” という複雑なブロック線図からなる window が現れます。これが SIMULINK のプログラムです。



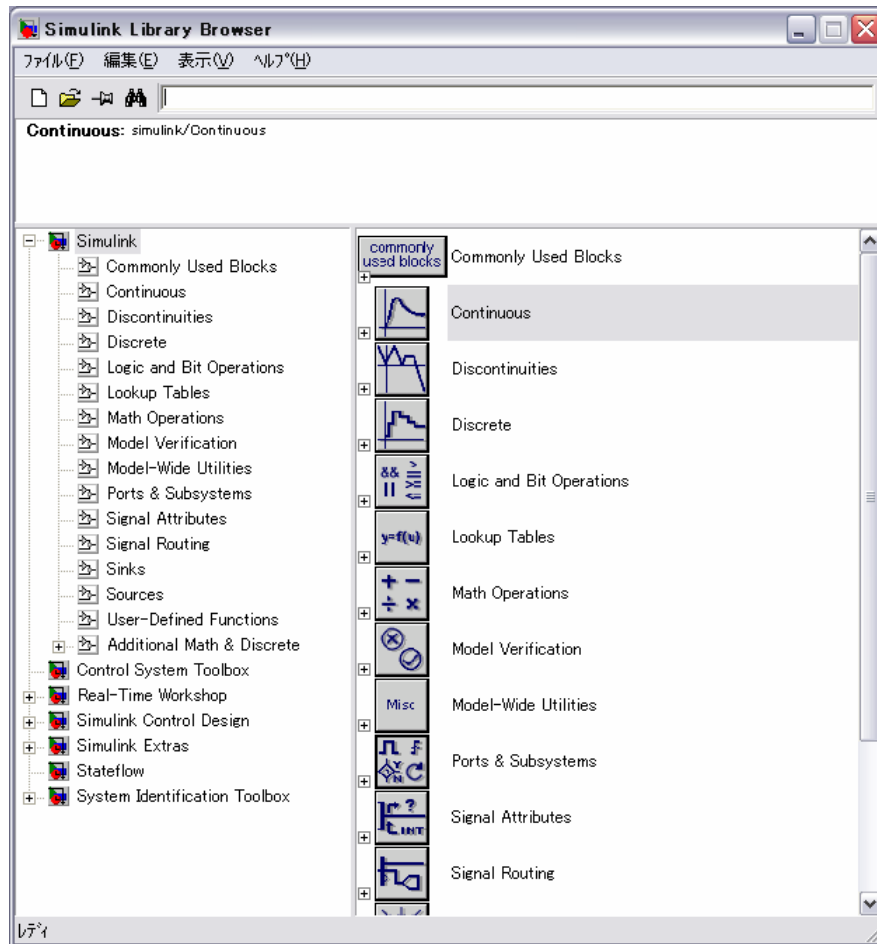
各ブロックの内容については後から理解することにして、とにかくシミュレーションを実行してみましよう。メニューの“シミュレーション”-“開始”を選択すると、シミュレーションが実行されます。このとき現れるアニメーション画面のスライダーを利用して、倒立振り子の位置を変化させることができます。このデモのようなアニメーション付きシミュレーションのプログラムを作成するためには、かなりの熟練を必要としますが、通常はアニメーションなしで十分でしょう。以下では、実際に SIMULINK を用いてシミュレーションプログラムを作成します。とりあえず、不必要な window は全て閉じておきましょう。

## 2.2. 簡単なプログラムの作成と実行

MATLAB の Command Window 上で

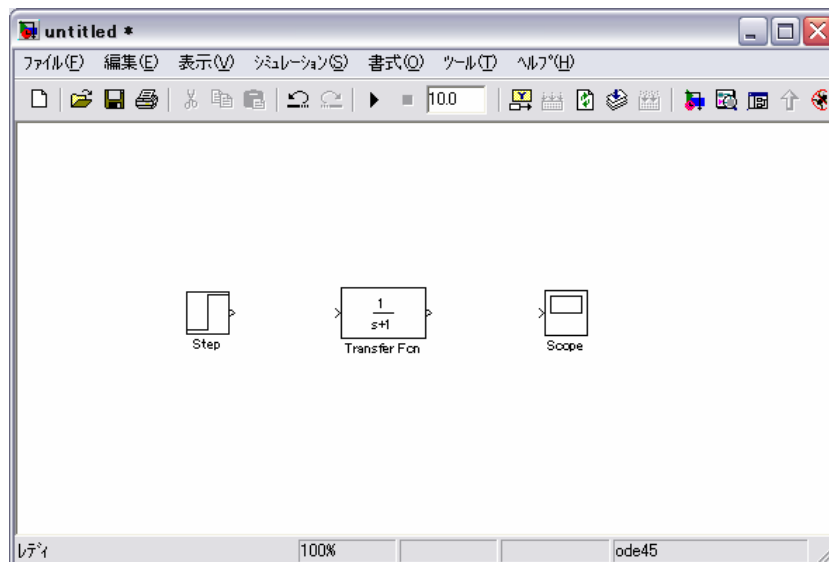
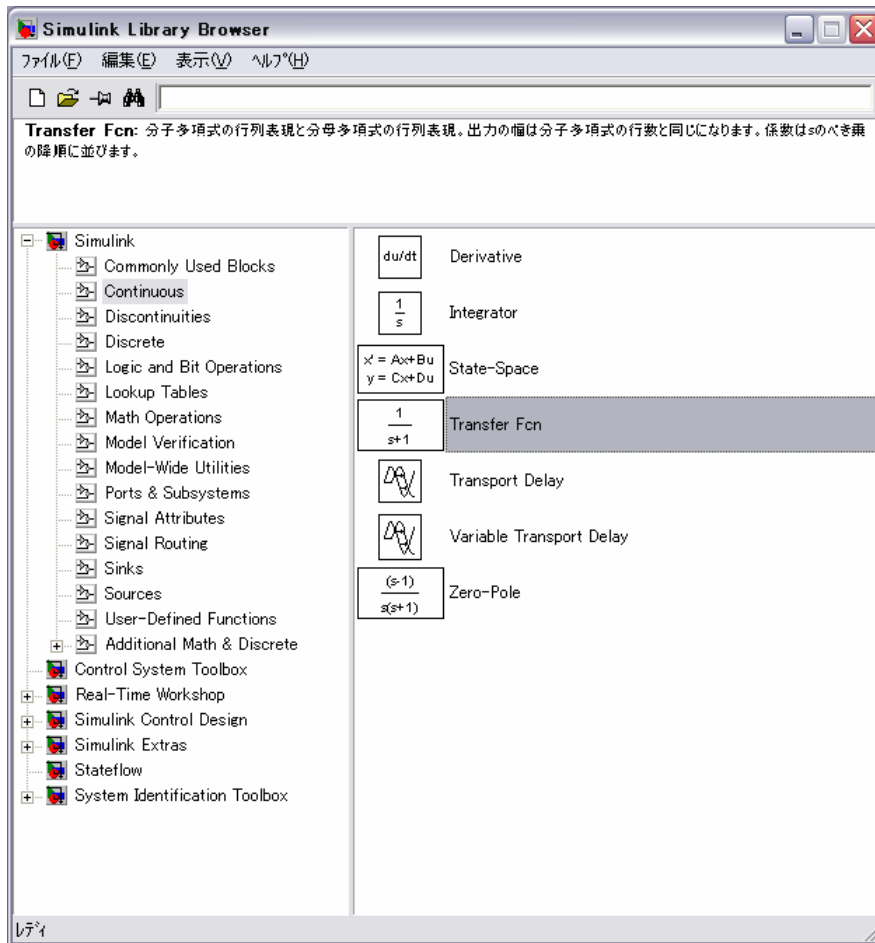
```
>> simulink
```

と入力して下さい. すると, “Simulink Library Browser” というウィンドウが現れます.

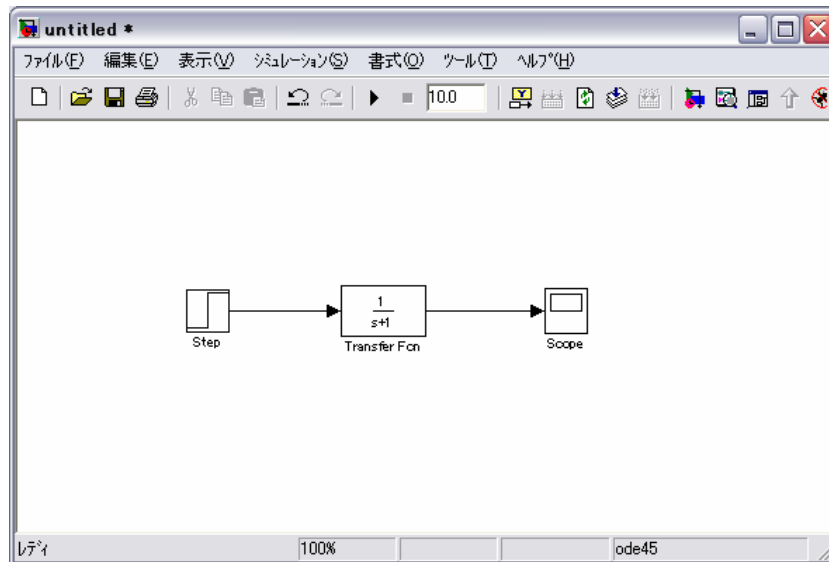


新しくプログラムを作成するときは, Simulink Library Browser のメニューの “ファイル” - “新規作成” - “モデル” を選択し, 新たに現れる untitled window にブロック線図を描きます. ここでは, 簡単な例として, 1 次遅れ要素のステップ応答を求めるシミュレーションプログラムを作成してみましょう.

まず, 必要なブロックを untitled window に貼り付けます. ここで必要になるブロックは, ステップ入力, 伝達関数, グラフ出力の 3 つです. ステップ入力は Simulink Library Browser の “Sources” ブロック内にあるので, Sources ブロックを開いて下さい. この中の “Step” がステップ状入力を作成するためのブロックです. このブロックを Untitled window に貼り付けるためには, 目的地までドラッグ & ドロップすればいいだけです. 続けて, 他の 2 つのブロックも貼り付けましょう. “Continuous” ブロックの中にある “Transfer Fcn” と “Sinks” ブロックの中にある “Scope” です. 貼り付け方は “Step” と同じです.



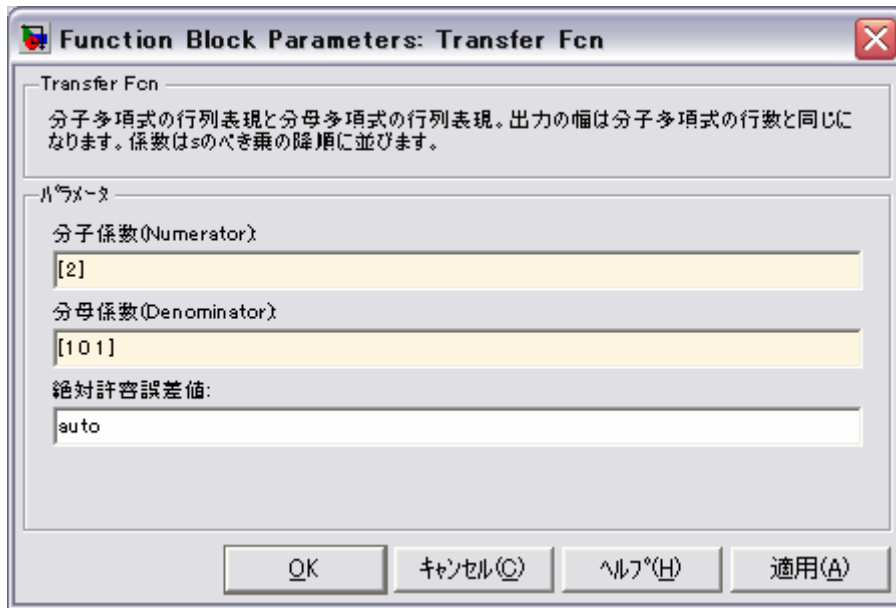
次に、各ブロックを接続しましょう。“Step”の“>”をクリックし、そのまま“Transfer Fcn”の“>”までカーソルを動かした後、ボタンを離します。同様の手順で“Transfer Fcn”と“Scope”も接続します。



それでは、各ブロックの設定に移ります。対象とするブロックをダブルクリックすることにより、設定画面が現れます。まずは、“Step” の設定です。ここでは、以下のように設定しておきましょう。10 ステップに大きさ 1 のステップ入力を与えます。

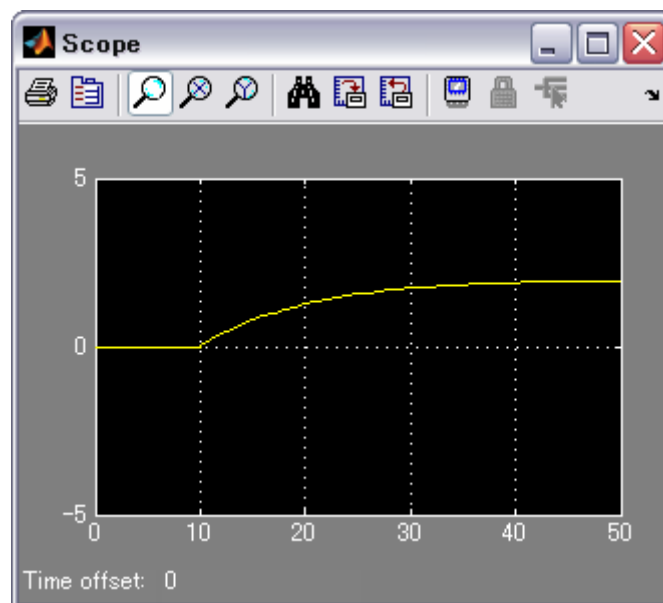


“Transfer Fcn” については、以下のように設定しておきましょう。これは、対象とする一次遅れ要素の伝達関数が  $\frac{2}{10s+1}$  で表されることを意味します。“Scope” については何も設定しなくて構いません。



あとは、シミュレーション条件の設定です。untitled window の “シミュレーション” - “コンフィギュレーションパラメータ” を選択し、“Stop Time” を 50.0 に設定して下さい。その他の項目はそのままで結構です。

それではいよいよシミュレーションの実行です。その前に Scope を開いておきましょう。Scope ブロックをダブルクリックして下さい。続いて、“シミュレーション” - “開始” を選択すると、シミュレーションが開始され、グラフが出力されます。確かに 1 次遅れ要素のステップ応答になっていますね。



計算機を利用するときには大切なのは、とにかく保存することです。今作成したシミュレーションプログラムもとにかく保存しておきましょう。ファイル名は “simu.mdl” としておきましょう。少なくとも、最後に “.mdl” を付けることを忘れないで下さい。

### 2.3. シミュレーションデータの取り扱い

ここでは、MATLAB と SIMULINK との間でのデータの受け渡しについて説明します。まず、全ての simulink に関連する window を閉じて下さい。画面上には MATLAB の Command window のみが残ります。

MATLAB から直接プログラム "simu.mdl" を起動してみましょう。

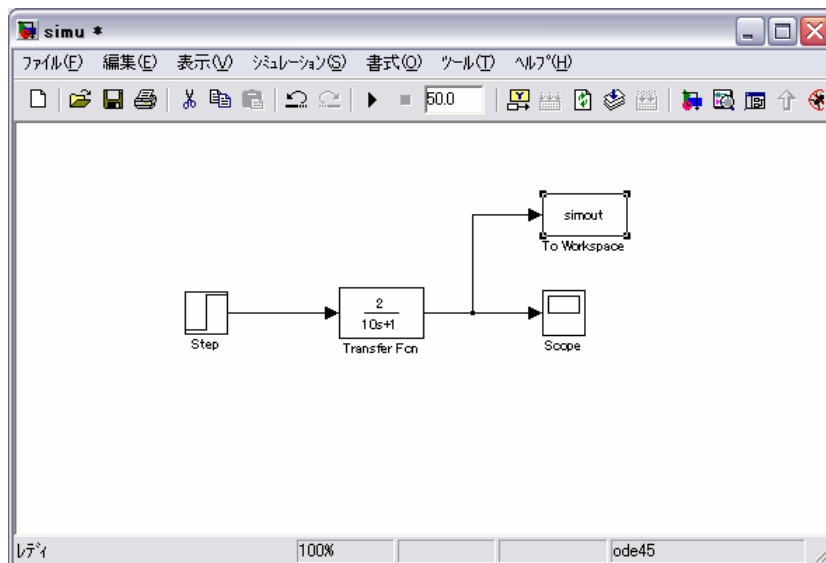
```
>> simu
```

と入力して下さい。前節で作成したプログラムが現れますね。ところが、これだけの操作では Simulink Library Browser は現れません。今後必要となりますので、

```
>> simulink
```

と入力します。

さて、simu.m をそのまま実行してもシミュレーションが行われるだけで、データは消え去ってしまいます。そこで、"Sinks" 内の "To Workspace" を利用して、MATLAB の workspace (作業領域) にデータを移す方法について説明します。とりあえず、"To Workspace" を simu.mdl に貼り付け、"Transfer Fcn" と接続して下さい。

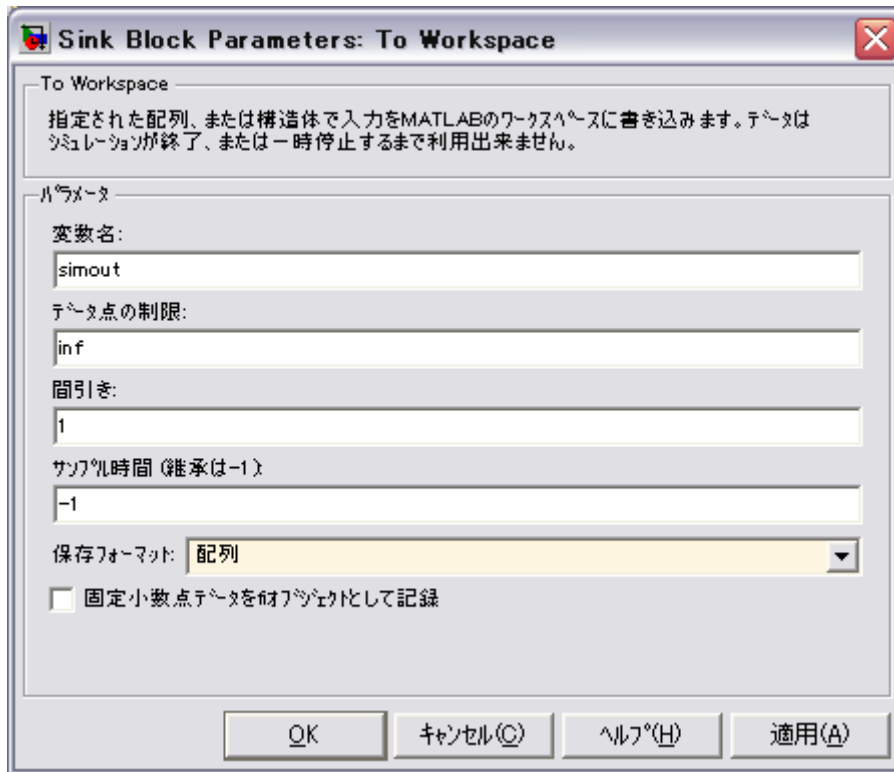


"To Workspace" の設定も行いましょう。一番下にある "保存フォーマット" を "配列" にしておきます。設定が済んだらシミュレーションを実行し、MATLAB の Command Window で "simout" と "tout" というデータが生成されていることを確認して下さい。ここで、tout は時間です。さらに、このデータがステップ応答データであることを確認するために、一度プロットしておきましょう。

```
>> plot(tout, simout)
```

と入力して下さい。

このようにして workspace に移されたデータは、MATLAB のコマンドを用いて容易に編集あるいは保存することができます。



## 2.4. MATLAB によるシミュレーション条件の設定

本格的にシミュレーションを行うようになると、SIMULINK 上で条件設定するのが大変面倒な作業になります。ここでは、MATLAB 上でシミュレーション条件を設定する方法を述べます。例として、種々の伝達関数のステップ応答を求めるという問題を取り上げます。



まず、伝達関数のゲインと時定数をそれぞれ変数 K, T で表すことにします。“Transfer Fcn” の設定

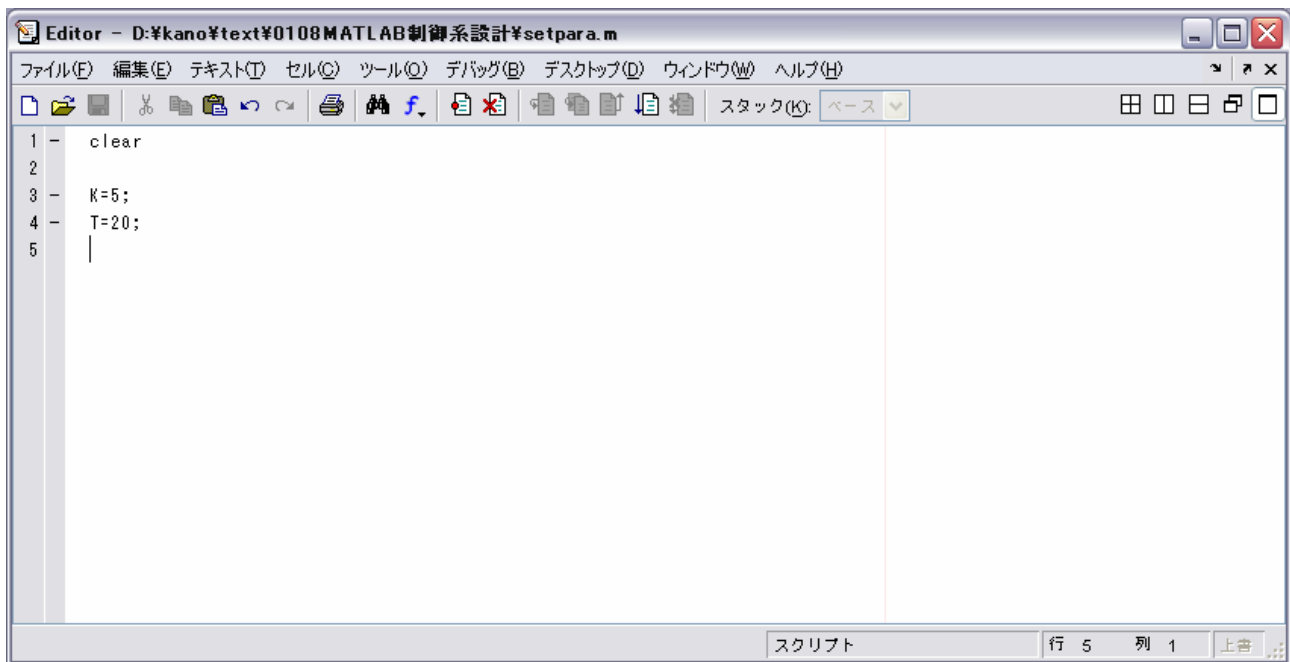
を変更して下さい。これらの変数の値を MATLAB から与えればいいわけです。そのために、新しい M-file を作成しましょう。ファイル名は `setpara.m` とします。K, T には適当な値を代入して下さい。ファイルを保存したら、Command Window に

```
>> setpara
```

と入力します。このようにして workspace 上で定義された変数は、SIMULINK 内でも有効になります。シミュレーションを実行してみましょう。

```
>> simu
```

確かに実行されますね。それでは、伝達関数の設定をいろいろと変化させて、シミュレーションを実行してみてください。



### [演習問題 3]

次の伝達関数

$$P(s) = \frac{2s+1}{4s^2+2s+1}$$

で表現されるシステムに正弦波 (Amplitude:1, Frequency:0.5) を入力として与えた場合の応答を求めなさい。

### [演習問題 4]

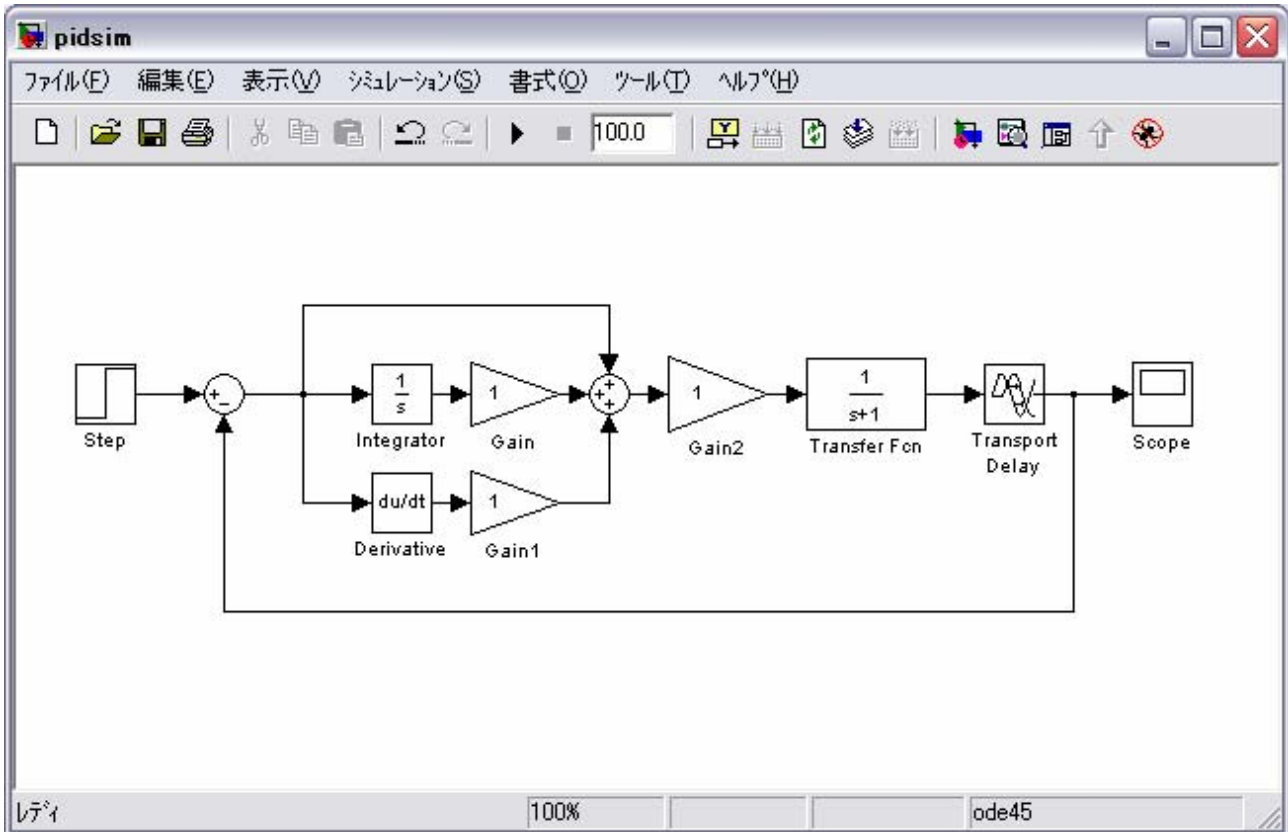
演習問題 3 の周波数応答を MATLAB の Work Space へ移し、ファイル名 `freqres` に保存しなさい。次に、この周波数応答の 2 周期分 (だいたいよい) をプロットしなさい。



## 第3章 PID 制御

### 3.1. SIMULINK による PID 制御シミュレーション

難しいことは後回しにして、とにかくシミュレーションプログラムを SIMULINK を用いて作成してみましょう。まずは、以下のようなブロック線図を作成し、“pidsim.mdl” というファイル名で保存して下さい。あるいは、お料理番組のように、用意したファイルを利用してもらっても構いません。

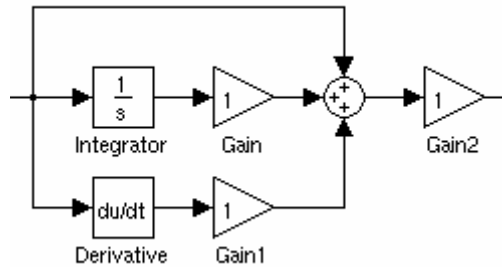


ここで各ブロックの設定を行います。

ブロック名	設定	設定値
Step	Step time	10
	Final value	10
Gain	Gain	1/20
		積分時間
Gain1	Gain	0
		微分時間
Gain2	Gain	0.8
		比例ゲイン
Transfer Fcn	Numerator	[5]
	Denominator	[10 1]
Transport Delay	Time delay	3
		むだ時間

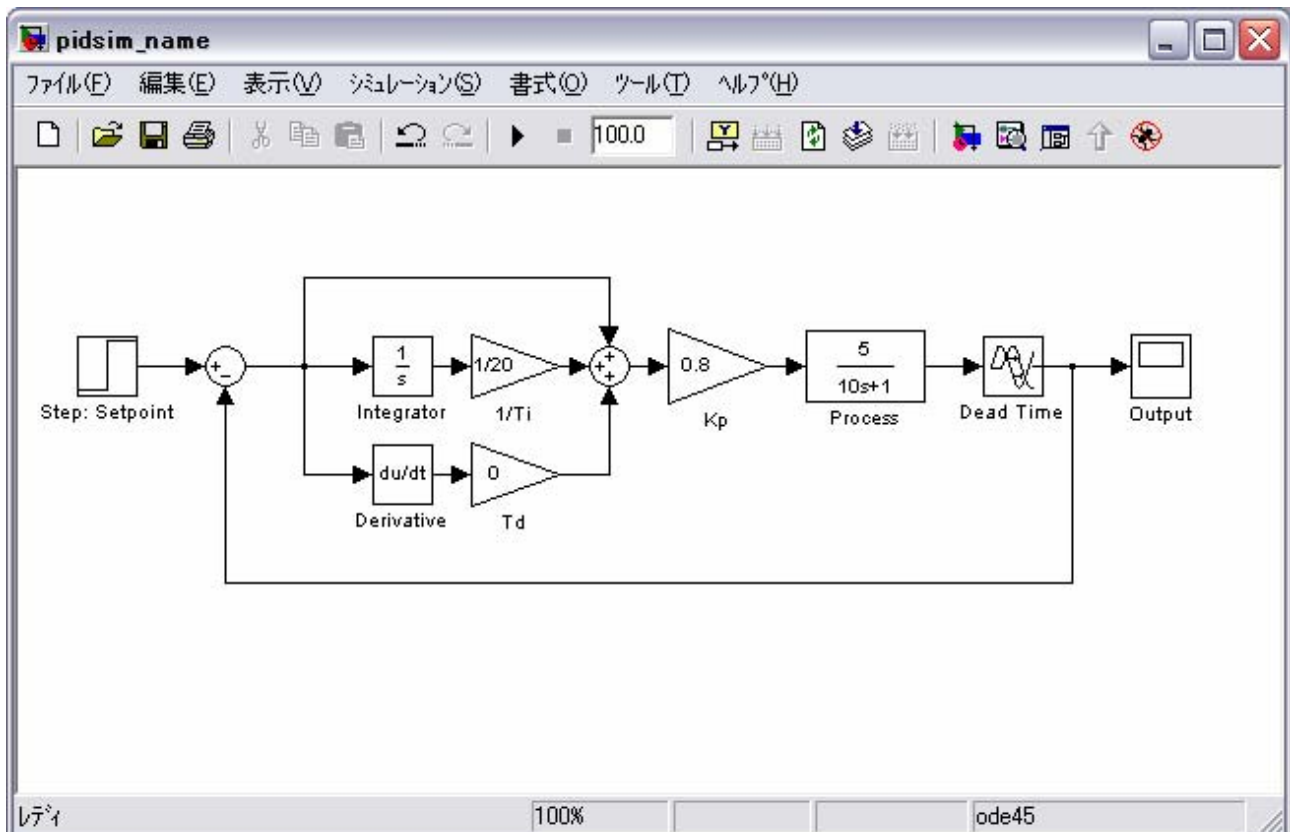
次に、シミュレーション条件の設定を行います。“シミュレーション”-“コンフィギュレーションパラメータ”内の“Stop Time”を100にしておきます。以上の設定が完了したら、シミュレーションを実行して下さい。出力が振動しながら10に近づいていくことが確認できます。

それでは、作成したプログラムの内容についてみていきましょう。もちろんPID制御系になっているのですが、外乱については考慮せず、設定値変更のみを扱っています。設定値に関する情報は“Step”ブロックで定義されています。このプログラムの中心であるPIDコントローラは、以下のようなブロック線図で表現できます。



PID controller

3種類のパラメータ（比例ゲイン、積分時間、微分時間）を“Gain”ブロックで定義しています。先程のシミュレーションでは“Gain1”を0に設定していましたから、実際にはPI制御のシミュレーションを行っていたこととなります。制御対象の伝達関数は“Transfer Fcn”と“Transport Delay”で与えています。この例では、プロセスは1次遅れ要素とむだ時間としています。



最後に、各ブロックに適切な名前を付けておきましょう。例えば、上図のように名前を書きおくと、 $K_p$ ,  $T_i$ ,  $T_d$  が比例ゲイン、積分時間、微分時間で、プロセスは 1 次遅れとむだ時間からなることが一目でわかります。

### 3.2. 限界感度法によるチューニング

ここでは、Ziegler-Nichols の限界感度法を用いて、PID コントローラのパラメータを決定する方法について述べます。制御対象の伝達関数は

$$P(s) = \frac{5}{10s+1} e^{-3s}$$

で与えられると仮定します。

まず、このプロセスを比例制御した場合にちょうど安定限界に達する比例ゲイン（限界感度）の値を求めます。安定限界に達するということは、制御系の一巡伝達関数  $C(s)P(s)$  のゲイン余裕がゼロになることと同じです。そこで、以下のような M-file を作成します。

```

行
1.      clear, clc
2.      process = tf(5, [10 1], 'InputDelay', 3);
3.      figure(1); step(process)
4.      figure(2); margin(process)

```

プログラムの説明をしましょう。2 行目で伝達関数を定義しています。3 行目でそのステップ応答をプロットしています。8 行目でボード線図を描き、ゲイン余裕および位相余裕を求めています。

実際にこのプログラムを実行すると、ゲイン余裕は  $G_m=1.42\text{dB}$ 、そのときの角周波数は  $0.58\text{rad/sec}$  であることがわかります。もし計算結果が異なる場合は、プログラムをチェックして下さい。

ゲイン余裕が  $1.42\text{dB}$  であることを式で表現すると、

$$-1.42 = 20\log|P(j\omega)| \quad \text{where } \omega = 0.58$$

となります。いま求めたいのは一巡伝達関数  $C(s)P(s) = K_C P(s)$  のゲイン余裕がちょうどゼロになるときの比例ゲイン  $K_C$  です。これを式で表すと、

$$0 = 20\log K_C |P(j\omega)| \quad \text{where } \omega = 0.58$$

となります。得られた 2 式を連立させて比例ゲイン  $K_C$  について解くと、

$$K_C = 10^{1.42/20} = 1.18$$

が得られます。一方、安定限界における振動周期  $T_C$  は、位相交点角周波数から

$$T_C = 2\pi / 0.58 = 10.8$$

となります。

得られた限界感度と振動周期を、限界感度法による調整則に代入してみましょう。例えば、PI 制御を行う場合、制御パラメータは

$$K_p = 0.45K_C = 0.531$$

$$T_I = 0.833T_C = 9.00$$

となります。PID 制御を行う場合は、

$$K_p = 0.6K_C \quad T_I = 0.5T_C \quad T_D = 0.125T_C$$

から計算することができます。

### 3.3. 制御シミュレーション

前節で得られた制御パラメータを用いて、PI 制御のシミュレーションを実行してみましょう。さらに、適当に制御パラメータの値を変化させて、制御シミュレーションを実行してみてください。PID 制御も実行しておきましょう。制御パラメータが制御性能に及ぼす影響を実感して下さい。

#### [演習問題 5]

プロセスの出力側にステップ状外乱が入るという設定でのシミュレーションを行いなさい。なお、プロセスの伝達関数はそのままとし、設定値変更は行わないとする。

#### [演習問題 6]

プロセスが次の伝達関数

$$P(s) = \frac{2s+1}{4s^2+2s+1}$$

で表されると仮定し、限界感度法を用いて PI コントローラを設計しなさい。さらに、ステップ状設定値変更のシミュレーションを行いなさい。

おわり