令和4年度「専修学校による地域産業中核的人材養成事業」

■航空機設計・製造分野におけるDX人材養成事業■

令和4年度プロト教材資料

本プロト教材資料は、文部科学省の教育政策推進事業委託費による委託事業として、日本航空大学校が実施した令和4年度「専修学校による地域産業中核的人材養成事業」の成果物です。

制御プログラミング学習領域

MATLABによる制御プログラミング

制御プログラミング教材

目次

MATLAB 1．基礎

（1） MATLABの概要

MATLABは、**MAT**rix **LAB**oratoryを略したものであり、行列計算、ベクトル演算、グラフ化や3次元表示などの豊富なライブラリを持った、インタプリタ形式の高性能なテクニカルコンピューティング言語、環境としての機能を持つ。

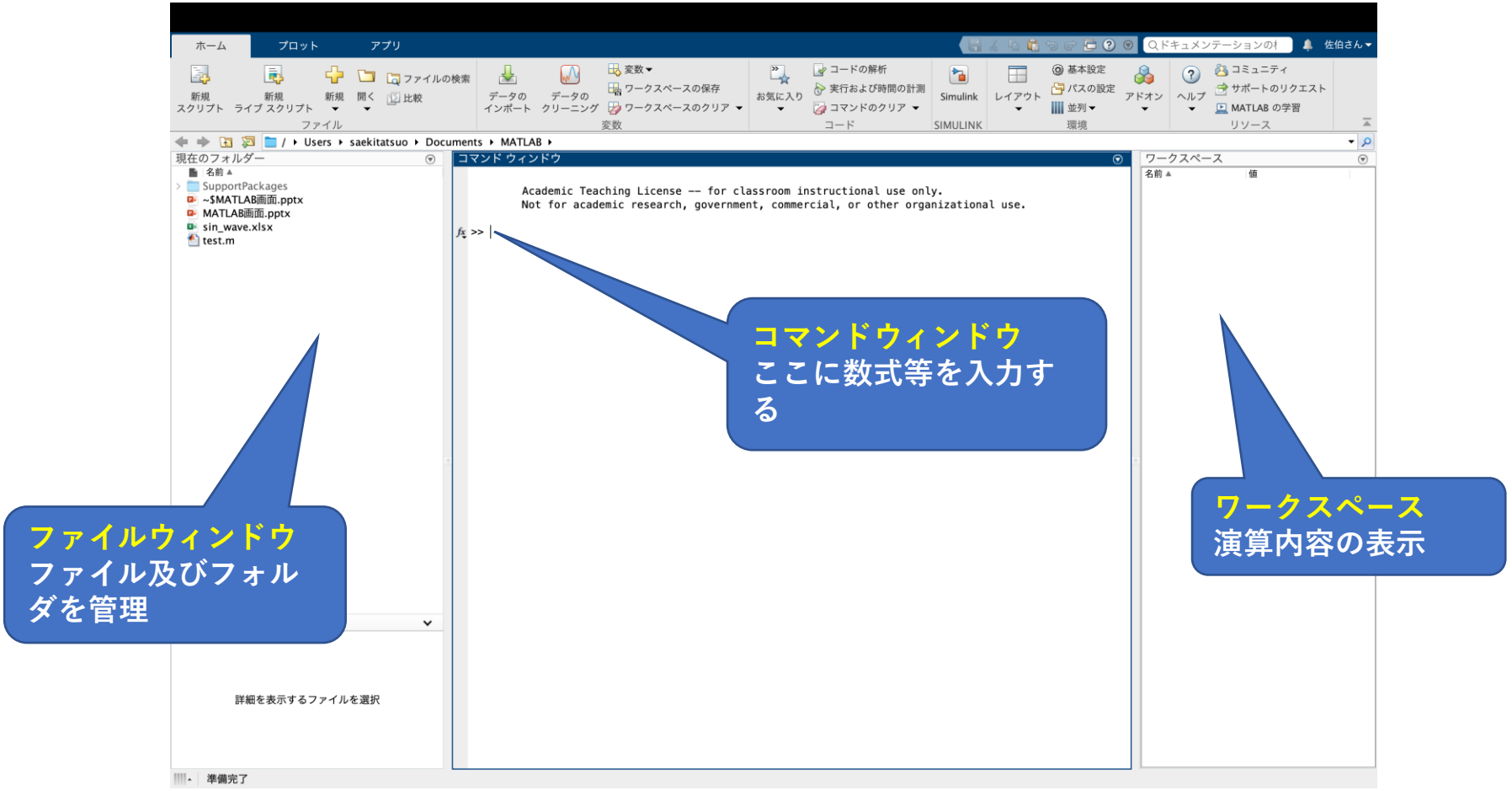
標準で数多くのライブラリを有しているが、それ以上のデータ解析や統計、アプリケーション展開などが必要な場合にはToolboxと呼ばれる拡張パッケージをインストールすることで、MATLABの機能拡張を図ることができる。MATLABとToolboxは総合してMATLABプロダクトファミリと呼ばれる。

このファミリの１つに制御システム開発のためのSIMULINKがある。

（2） MATLABの起動

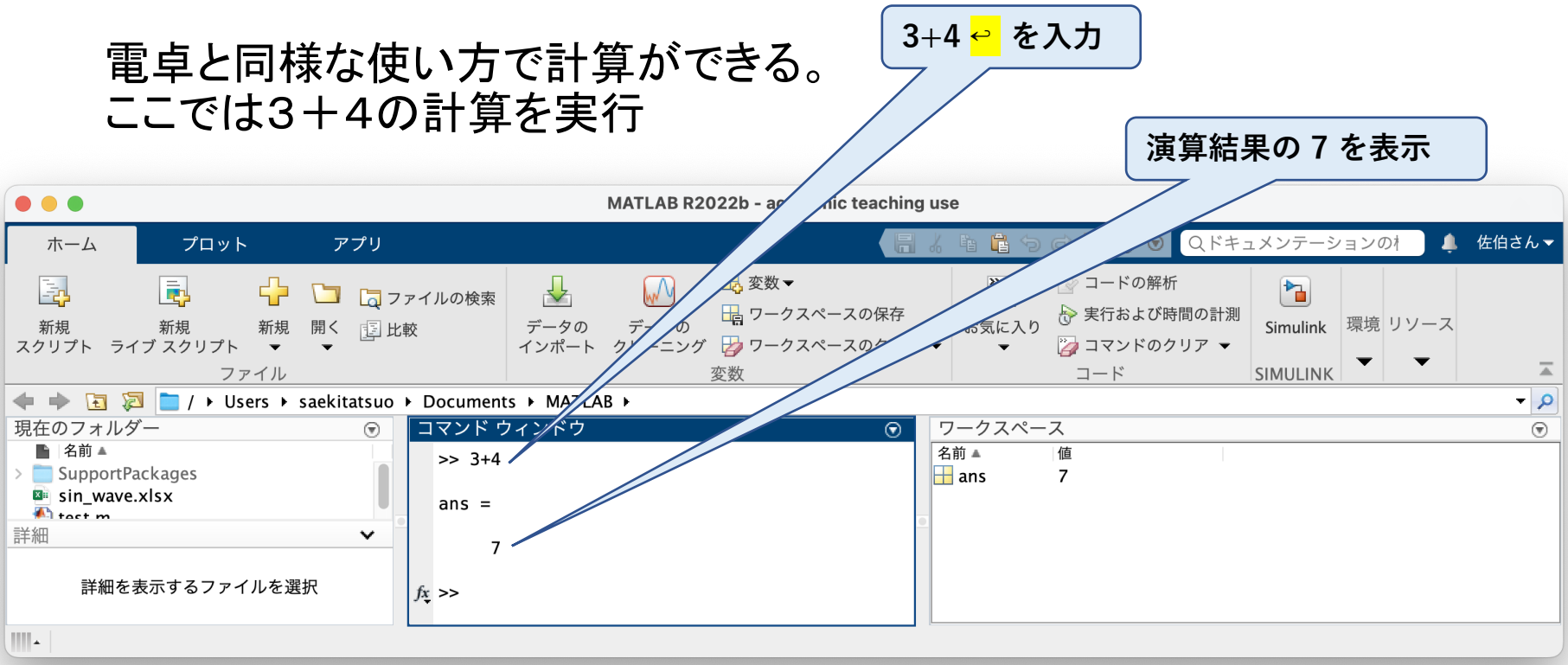
Windows又はMACで以下のアイコンをクリックしてMATLABを起動

（3） MATLAB画面

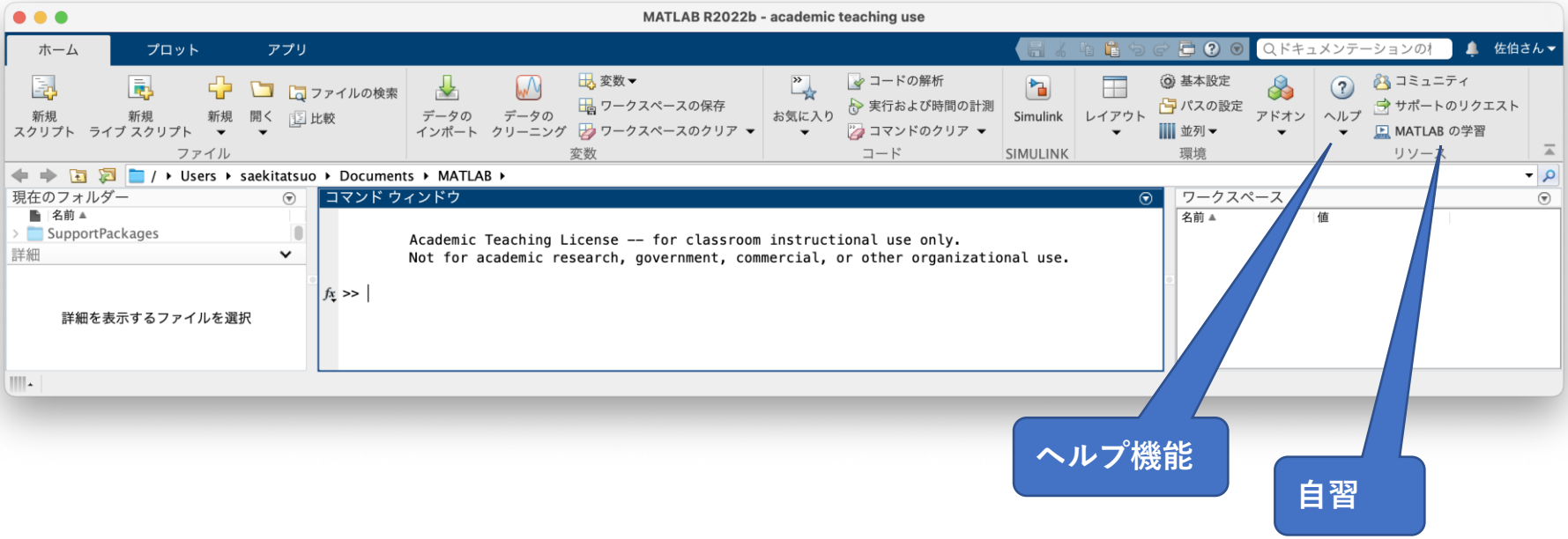

ホーム
プロット
アプリ
Users ▸ saekitatsuo ▸ Documents ▸ MATLAB ▸
現在のフォルダー
名前
SupportPackages
~$MATLAB画面.pptx
MATLAB画面.pptx
sin_wave.xlsx
test.m
詳細を表示するファイルを選択
コマンド ウィンドウ
Academic Teaching License -- for classroom instructional use only.
Not for academic research, government, commercial, or other organizational use.
>>
ワークスペース
名前
値
準備完了
ファイルウィンドウ
ファイル及びフォルダを管理
コマンドウィンドウ
ここに数式等を入力する
ワークスペース
演算内容の表示

（4） 単純計算

電卓と同様な使い方で計算ができる。
ここでは3＋4の計算を実行



（5） ヘルプ機能・学習機能


MATLAB R2022b - academic teaching use
ホーム
プロット
アプリ
ドキュメンテーションの
佐伯さん
新規 スクリプト
新規 ライブ スクリプト
新規
開く
ファイルの検索
比較
ファイル
データの インポート
データの クリーニング
変数
ワークスペースの保存
ワークスペースのクリア
変数
お気に入り
コードの解析
実行および時間の計測
コマンドのクリア
コード
Simulink
SIMULINK
レイアウト
基本設定
パスの設定
並列
環境
アドオン
ヘルプ
コミュニティ
サポートのリクエスト
MATLAB の学習
リソース
/ ▸ Users ▸ saekitatsuo ▸ Documents ▸ MATLAB ▸
現在のフォルダー
名前
SupportPackages
詳細
詳細を表示するファイルを選択
コマンド ウィンドウ
Academic Teaching License -- for classroom instructional use only.
Not for academic research, government, commercial, or other organizational use.
>>
ワークスペース
名前
値
ヘルプ機能
自習

（6） 課題

1．123+456

2．(3+4)×(5+6)

3．3/4

4．987−1024

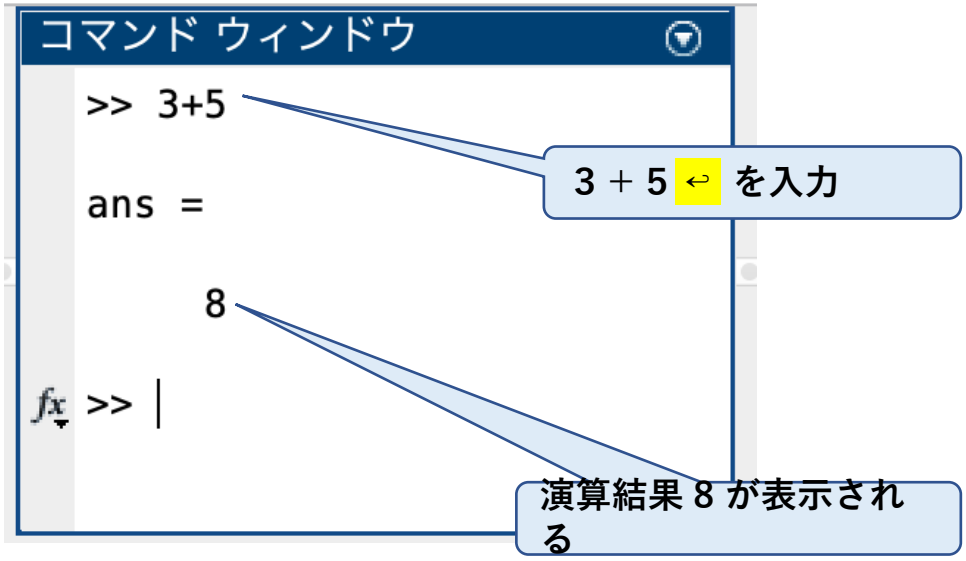
5．3.14159×2.3×2.3

MATLAB 2. 単純計算

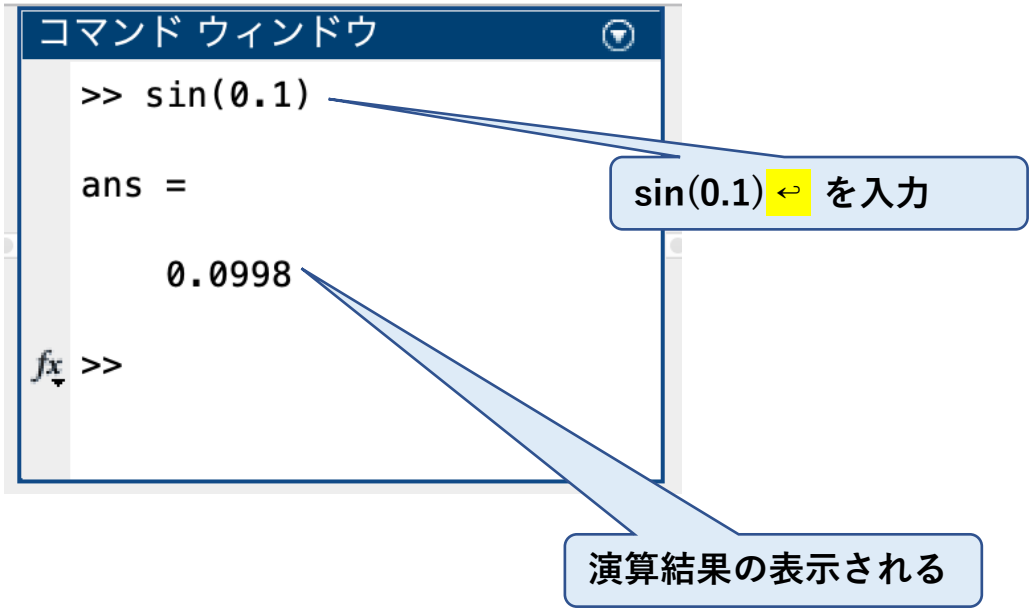
（1） 単純計算

MATLABは、電卓と同様な計算機能を有する。計算の手順は電卓と同様に数値及び演算子を入力し、最後に↩（Return／Enter）キーを押すことにより実行される。

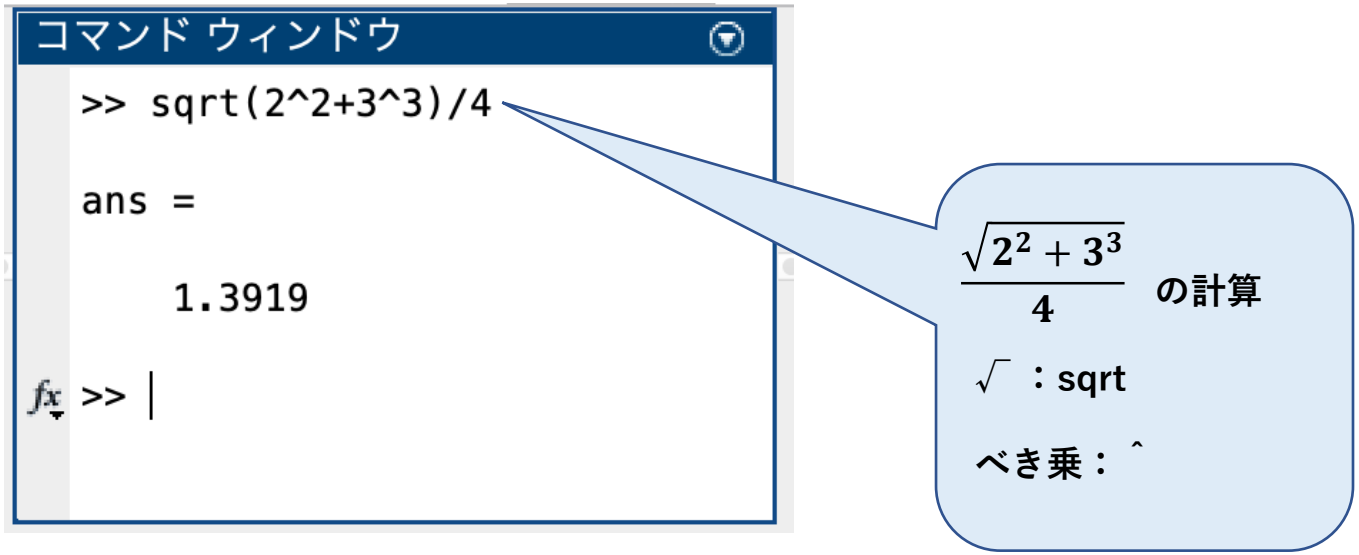
（2）3+5 の計算



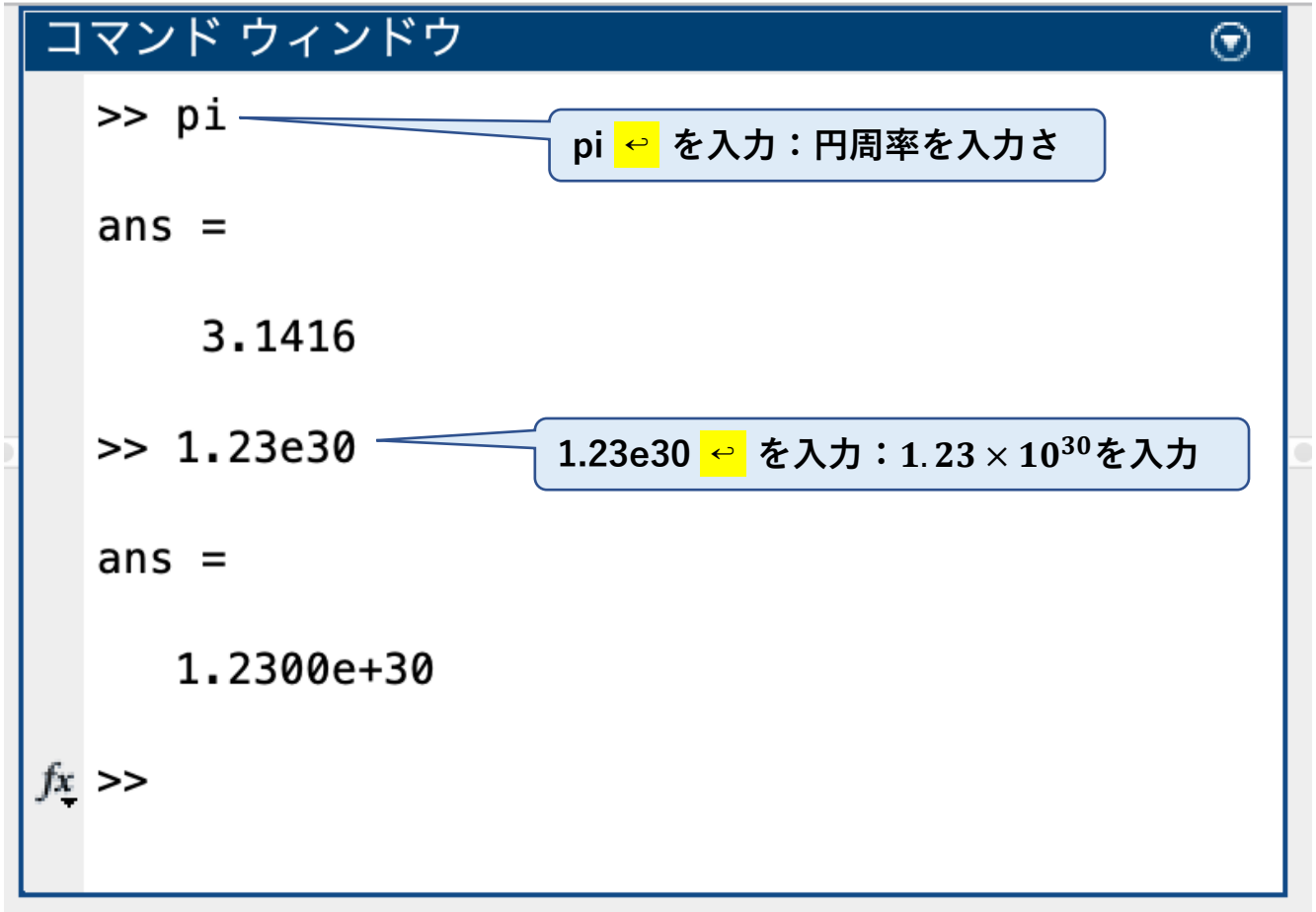
（3）sin(0.1) の計算



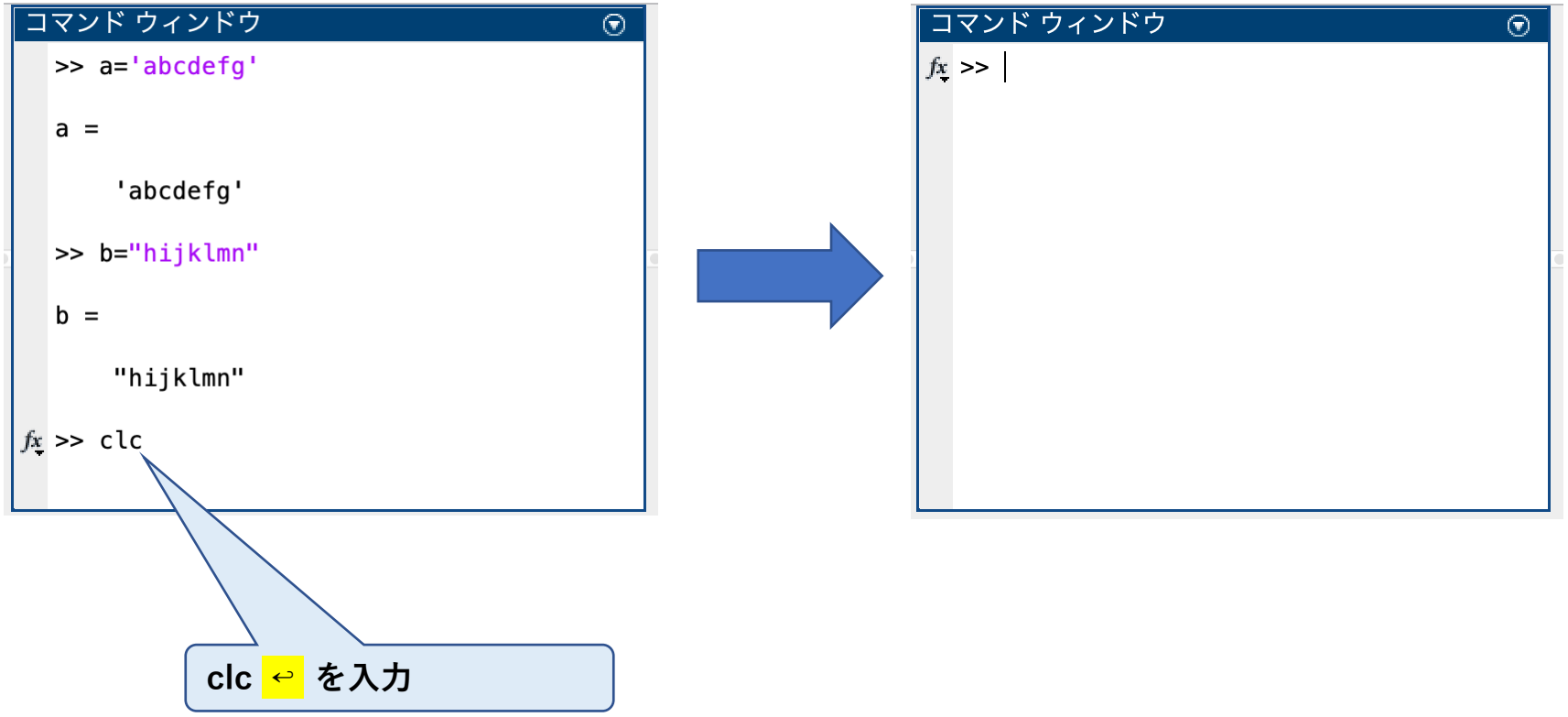
（4） √を含む計算



（5） その他の数値入力


コマンド ウィンドウ
>> pi
ans =
3.1416
>> 1.23e30
ans =
1.2300e+30
>>
pi ↩ を入力：円周率を入力さ
1.23e30 ↩ を入力：1.23×10^{30}を入力

（8） コマンドウィンドウの消去



（9） 代表的な演算子・関数

1. 四足演算他
   +, −, *, /

2. 三角関数
   sin, cos, tan, asin, acos, atan, atan2, sinh, cosh, tanh, asinh, acosh, atanh
   factorial

3. 対数関数
   exp, log, log10, ^[y^x], sqrt[$\sqrt{x}$]

4. 複素数
   abs, angle, imag, real

5. 行列
   zeros, eye, ones, inv, det

（10） 課題

1. 2^{100}

2. 100!（階乗）

3. sin(π)

4. cos(1)

5. $3 \times 10^5 \times 1.602 \times 10^{19}$

6. $\pi \times 3^2$

7. $\sqrt[3]{27}$

MATLAB ３．変数を用いた計算

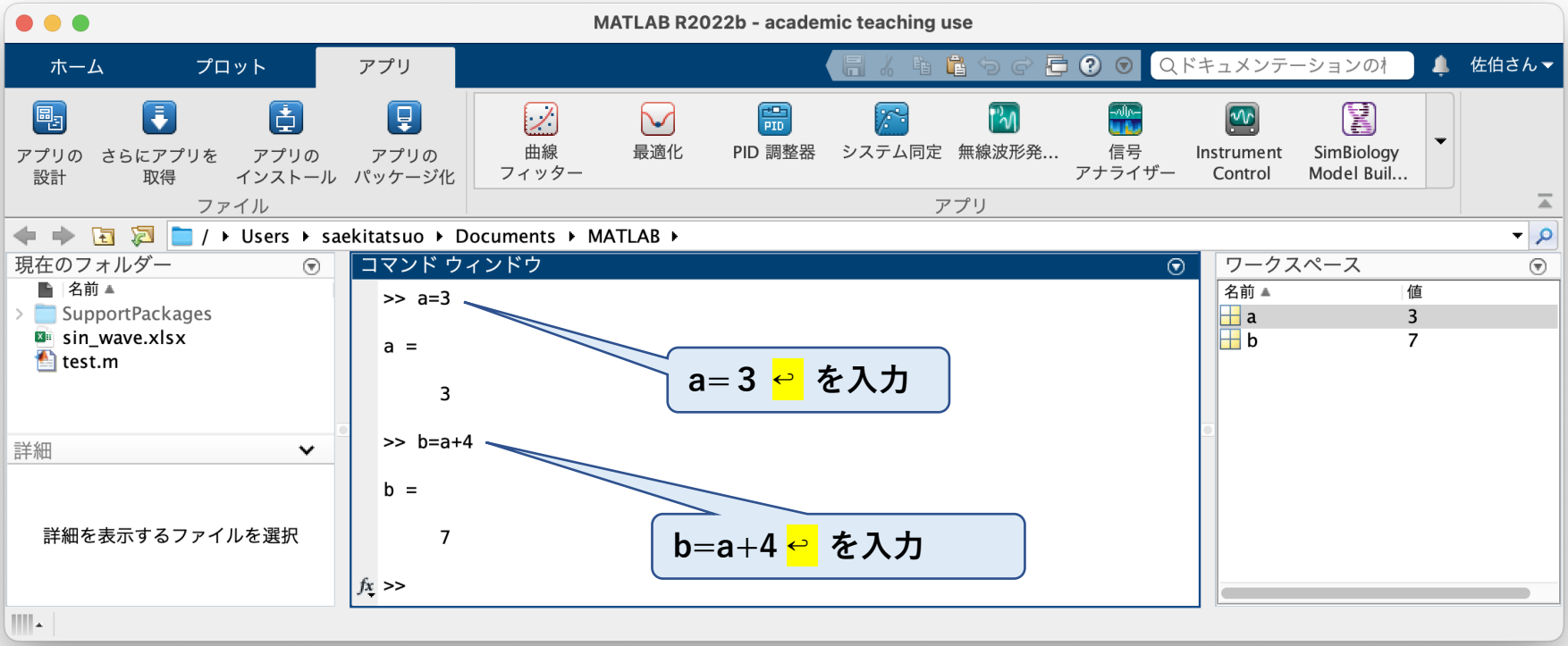
（１）変数を用いた計算概要

　MATLABは、変数を用いて計算することができる。変数は代入される側（＝の左側　例：a = 3）に置かれる場合は、C言語のような変数の宣言は必要無い。

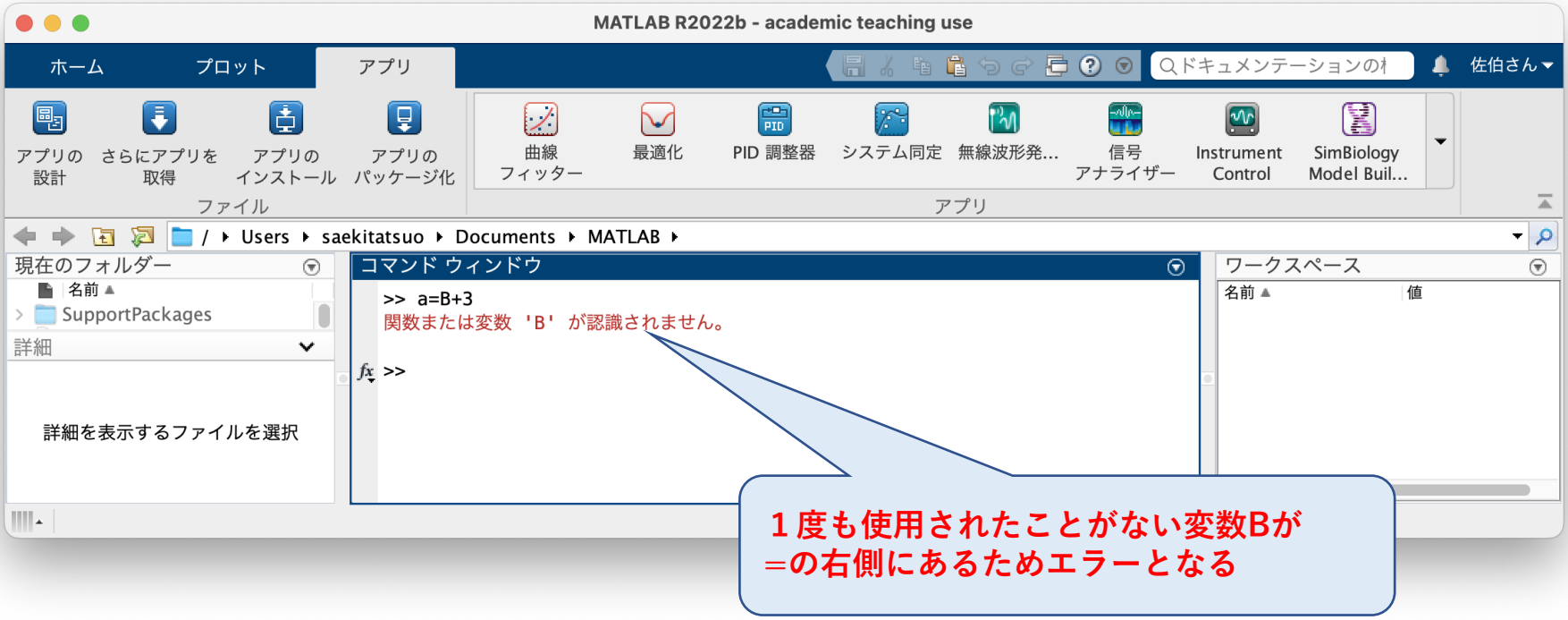
　又一度使用された変数は、代入する側（＝の右側）で使用することができる。

　１度も使用されていない変数を代入する側（＝の右側）で使用するとエラーとなることに注意。

（2）変数を用いた計算例


MATLAB R2022b - academic teaching use
ホーム
プロット
アプリ
ドキュメンテーションの
佐伯さん
アプリの設計
さらにアプリを取得
アプリのインストール
アプリのパッケージ化
ファイル
曲線フィッター
最適化
PID 調整器
システム同定
無線波形発...
信号アナライザー
Instrument Control
SimBiology Model Buil...
アプリ
/ ▸ Users ▸ saekitatsuo ▸ Documents ▸ MATLAB ▸
現在のフォルダー
名前
SupportPackages
sin_wave.xlsx
test.m
詳細
詳細を表示するファイルを選択
コマンド ウィンドウ
>> a=3
a =
3
>> b=a+4
b =
7
>>
a＝3 ↩ を入力
b=a+4 ↩ を入力
ワークスペース
名前
値
a
3
b
7

（3）間違った変数の使用例



（４）課題

１．Xに3×4の結果を代入し、さらにX^2を計算する。

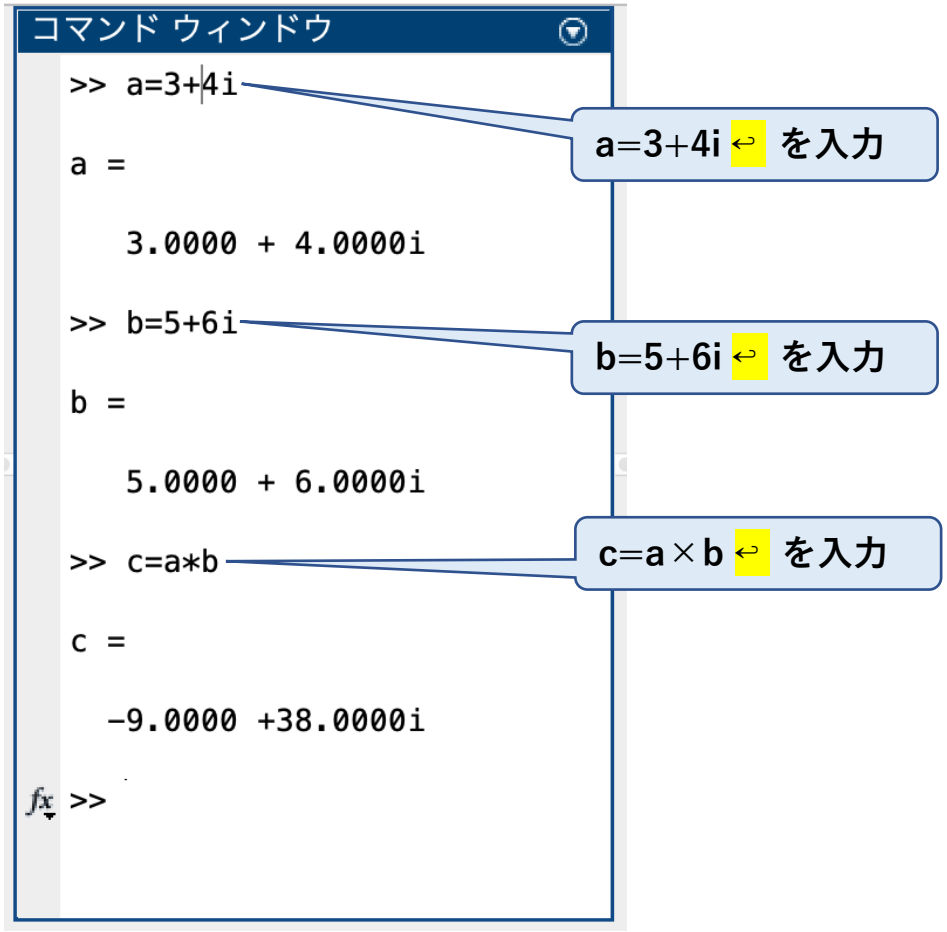
２．Xに3を代入し、Yに5を代入してX×Yを計算する。

MATLAB 4. 複素数計算

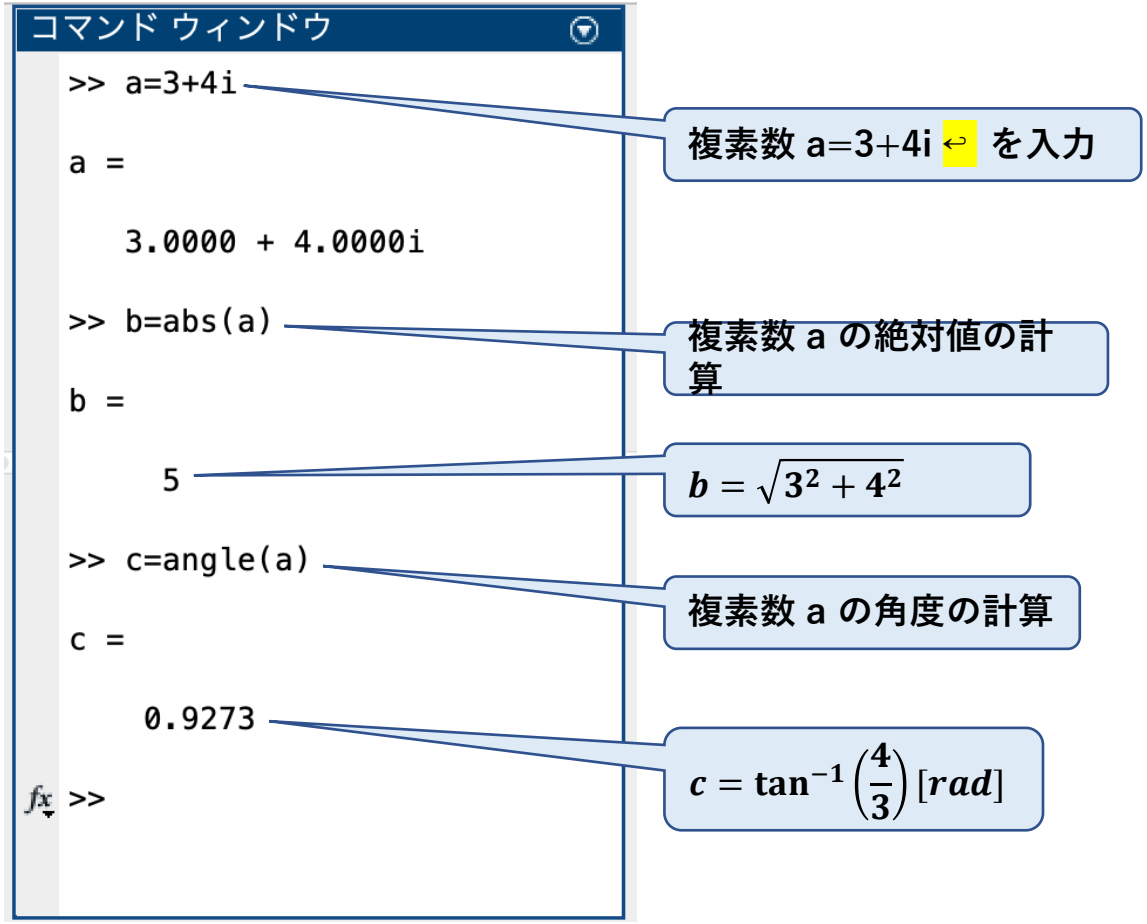
（1）概要

MATLABは、複素数も通常の数値と同様に計算することができる。
基本的な四則演算以外に、三角関数等の関数及び複素数の絶対値・角度の計算もすることができる。

（2）複素数の足し算・掛け算



（3）複素数の絶対値と角度



（4）複素数の演算機能

abs	絶対値と複素数の大きさ
angle	位相角
complex	複素数配列の作成
conj	複素共役
cplxpair	複素数を複素共役のペアに並べ替え
i	虚数単位
imag	複素数の虚数部
isreal	配列で複素数ストレージを使用するかどうかを判別
j	虚数単位
real	複素数の実数部
sign	符号関数 (関数 signum)
unwrap	位相角のシフト

（５）課題

１．５＋６iの絶対値を求めよ

２．５＋６iの位相角を求めよ

３．５＋６iから実数部分を取り出せ

４．５＋６iから虚数部分を取り出せ

５．５＋６iの複素共役を求めよ

MATLAB 5. 行列計算

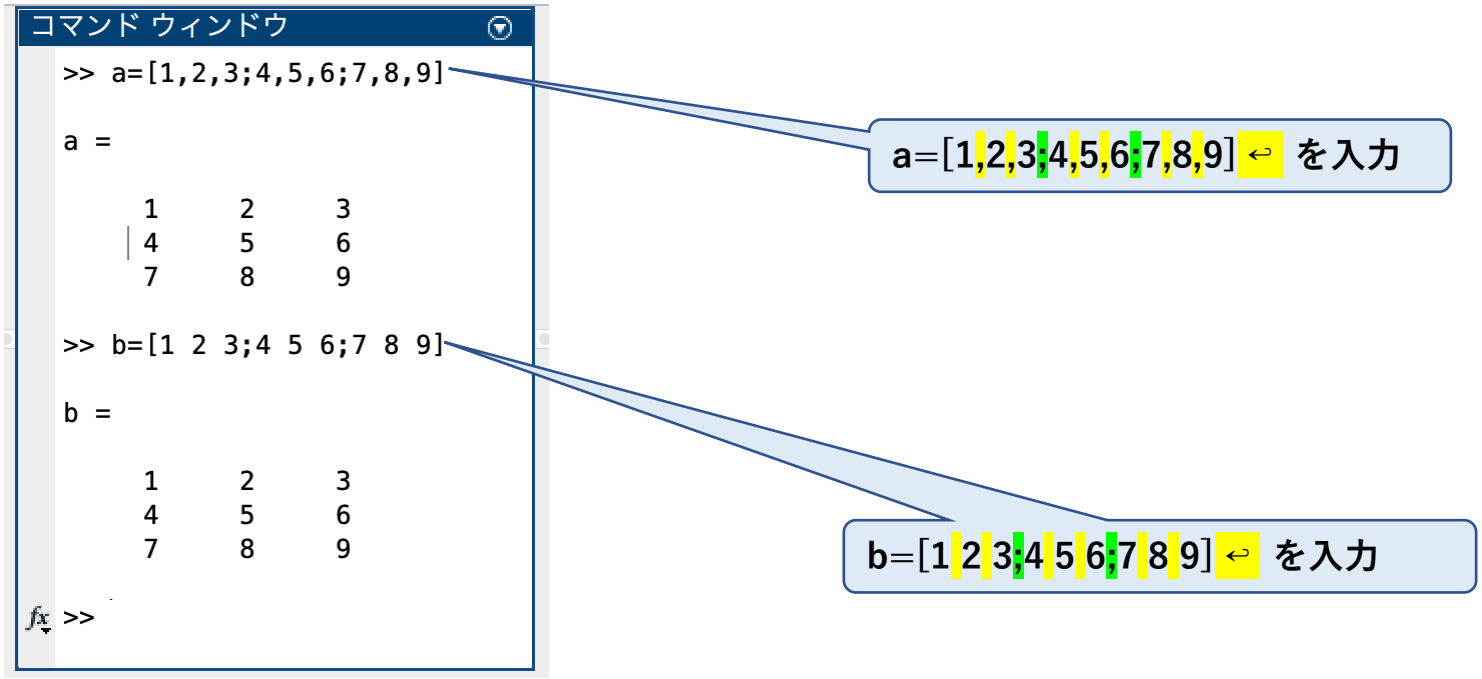
（1） 概要

MATLABは、行列の計算を行うことができる。変数を使用すれば行列同士の＋－×／と言った四則演算から、三角関数等の関数、又複素数を含む値も通常の変数と同様に計算することができる。

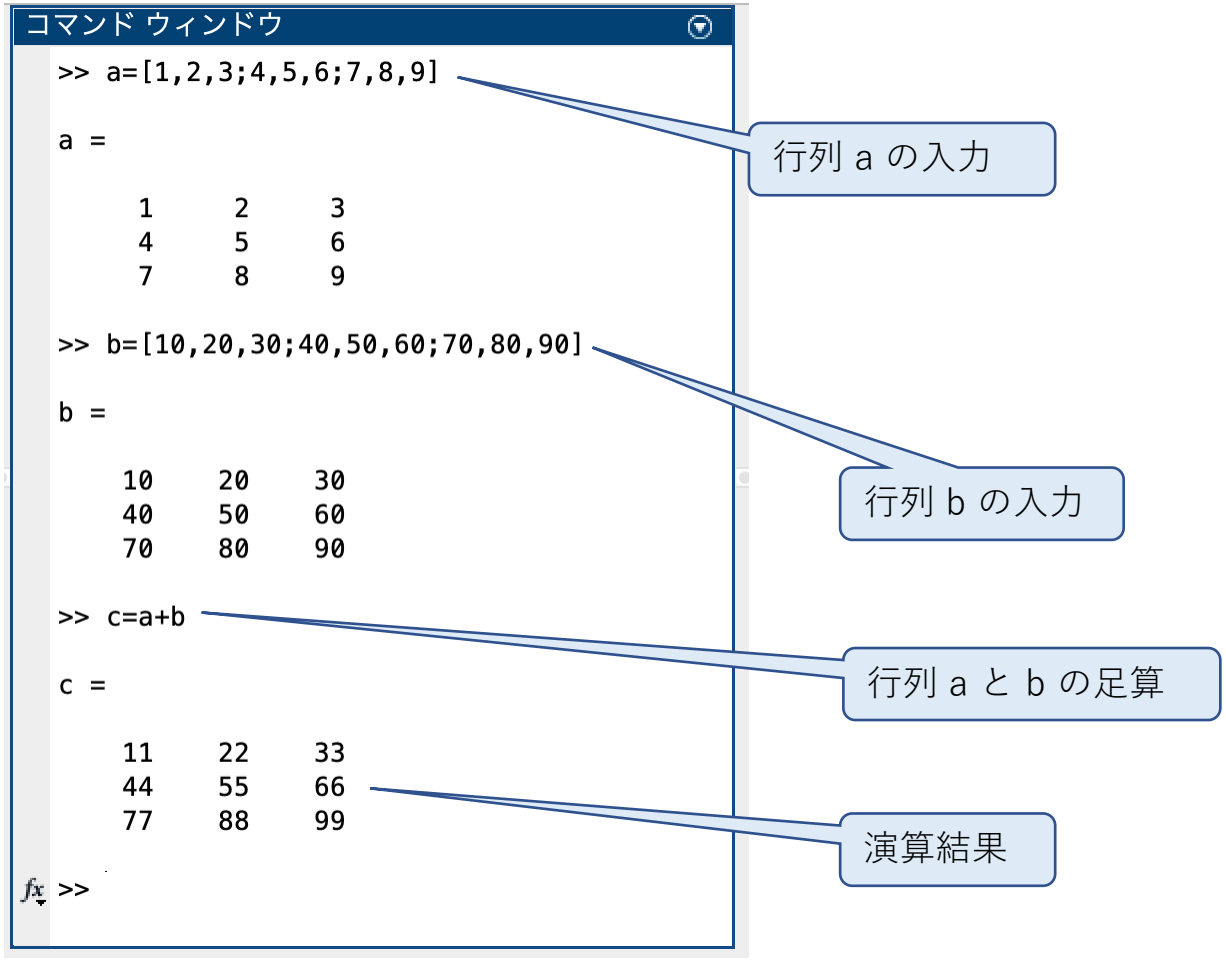
その他、逆行列や行列の値の計算等行列固有の計算も可能。

（2） 行列の入力

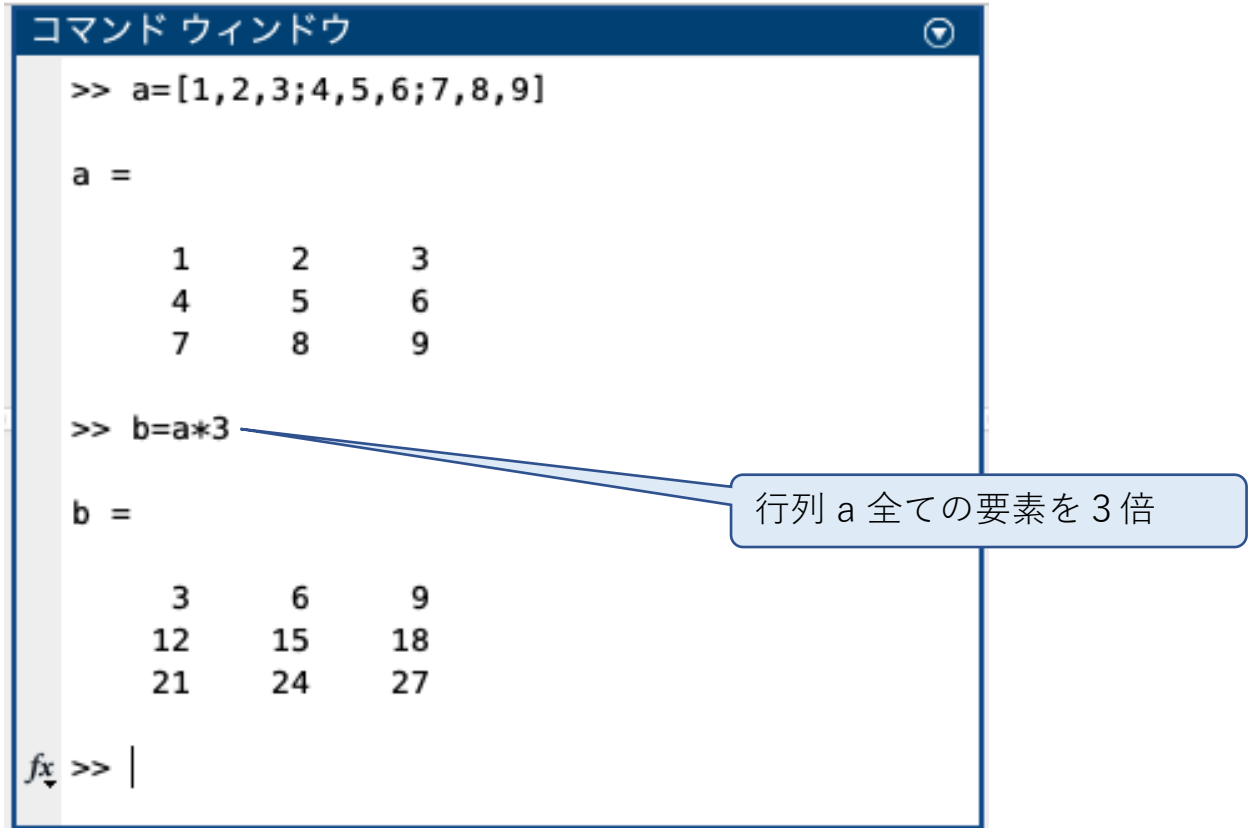
行列の値は[ ] で囲み、行の要素は , 又はスペースで分離する。さらに行の終わりに ; を入れる。



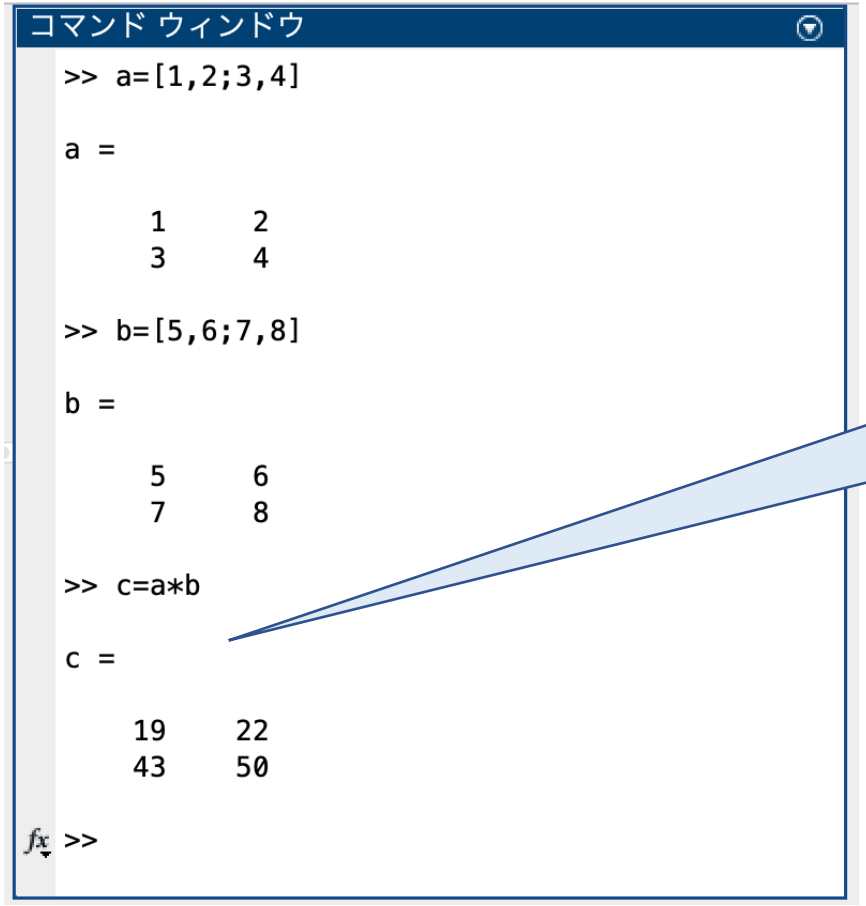
（3） 行列の足算


コマンド ウィンドウ
>> a=[1,2,3;4,5,6;7,8,9]
a =
1 2 3
4 5 6
7 8 9
>> b=[10,20,30;40,50,60;70,80,90]
b =
10 20 30
40 50 60
70 80 90
>> c=a+b
c =
11 22 33
44 55 66
77 88 99
>>
行列 a の入力
行列 b の入力
行列 a と b の足算
演算結果

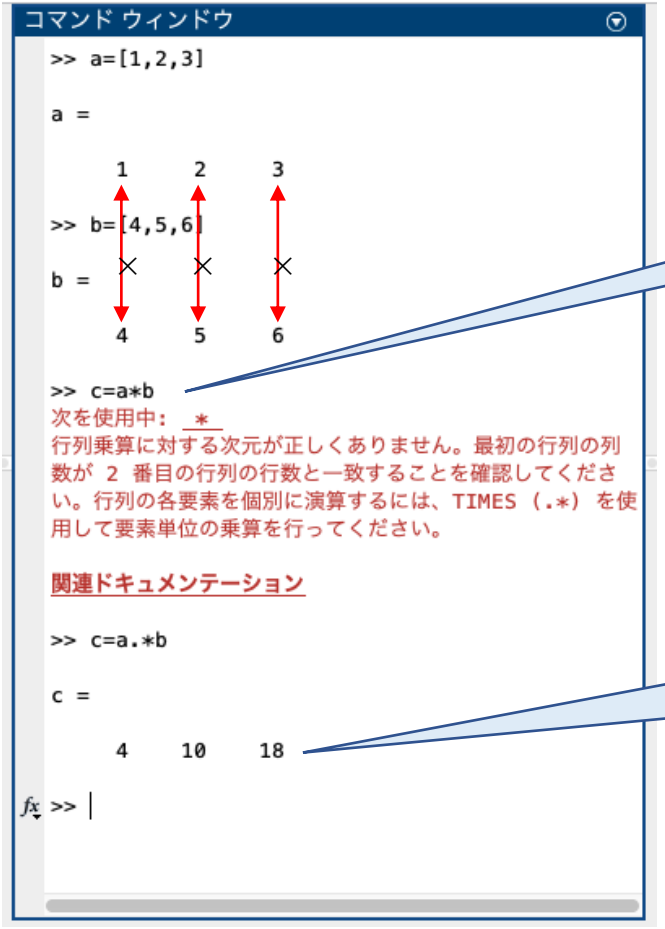
（4） 行列と数値の掛け算


コマンド ウィンドウ
>> a=[1,2,3;4,5,6;7,8,9]
a =
1 2 3
4 5 6
7 8 9
>> b=a*3
b =
3 6 9
12 15 18
21 24 27
>>
行列 a 全ての要素を 3 倍

（5） 行列と行列の掛け算−1



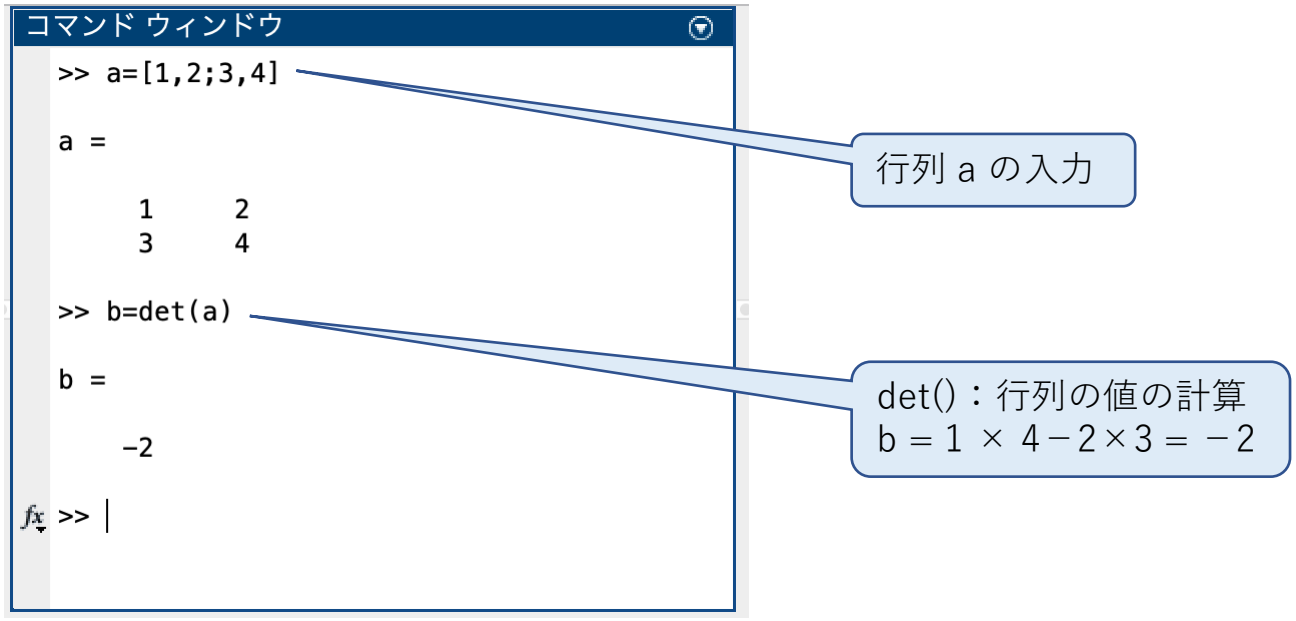
（6） 行列と行列の掛け算−2



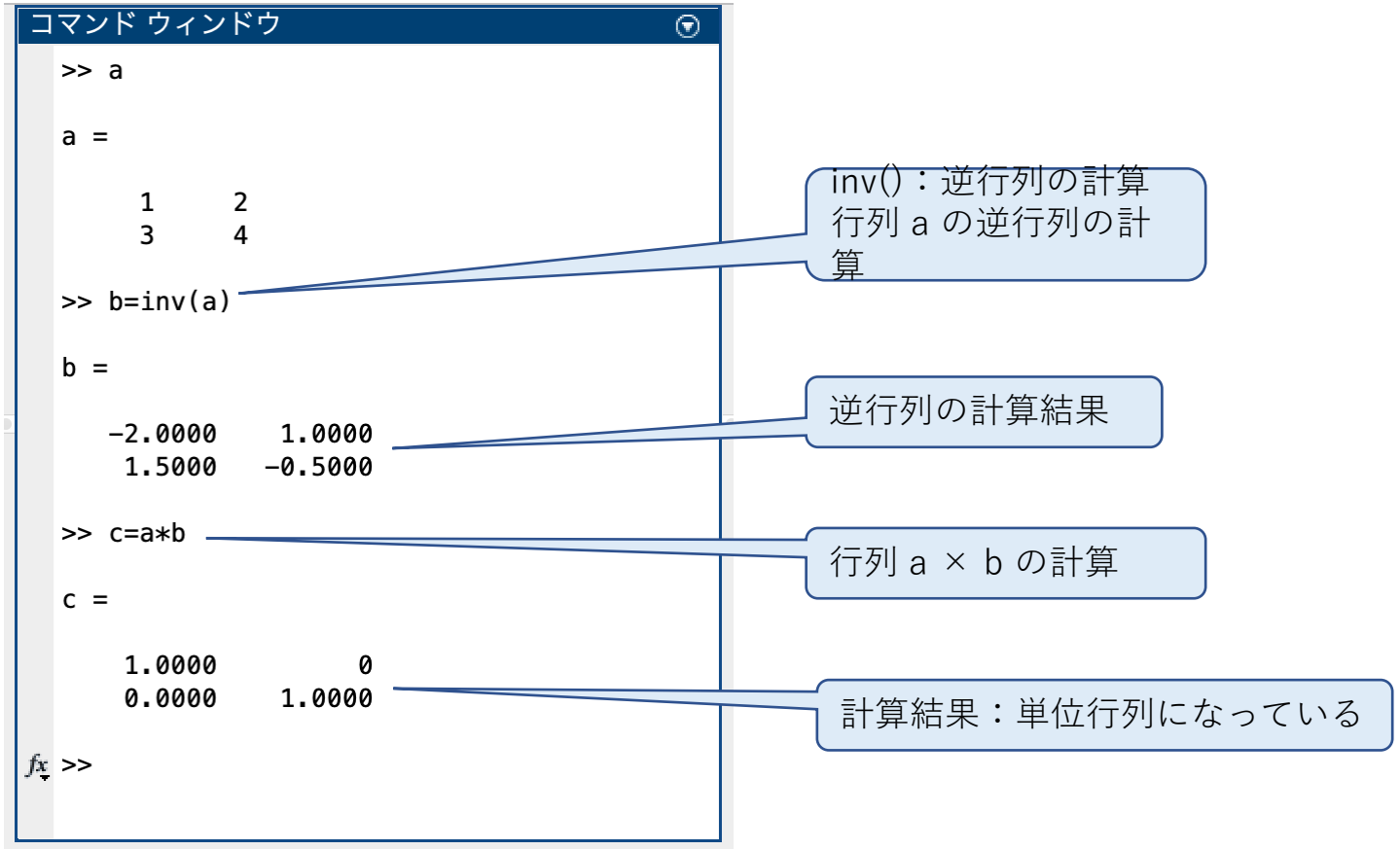
この場合、
行列 a × 行列 b
は計算できない

行列 a .× 行列 b
なら以下のように計算
4=1×4
10=2×5
18=3×6

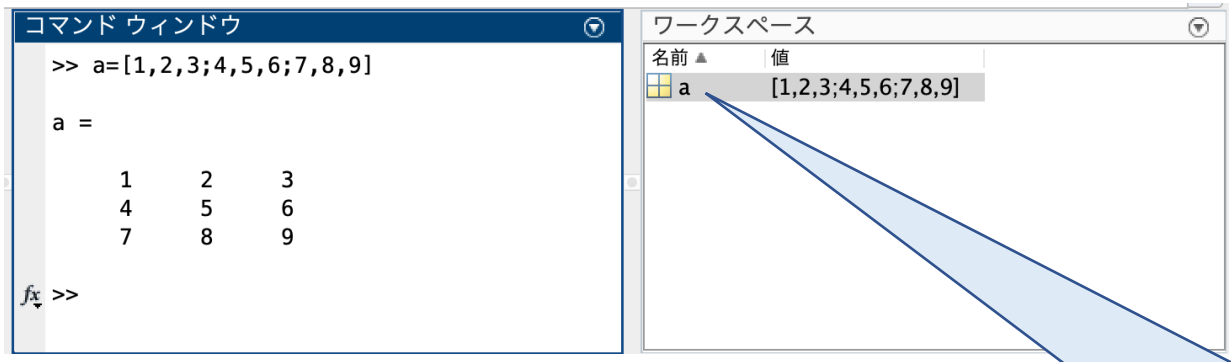
（7） 行列の値の計算



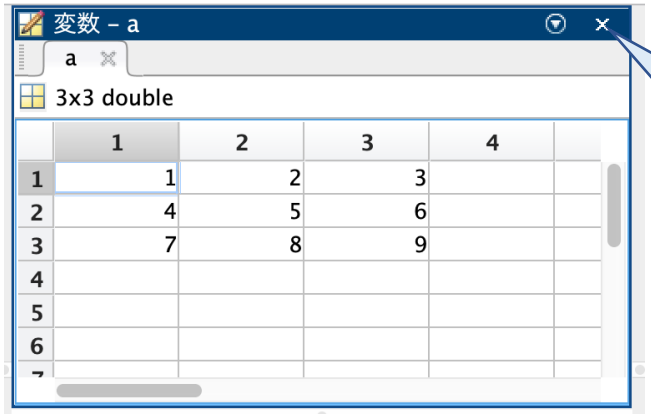
（8） 逆行列の計算


コマンド ウィンドウ
>> a
a =
1 2
3 4
>> b=inv(a)
b =
-2.0000 1.0000
1.5000 -0.5000
>> c=a*b
c =
1.0000 0
0.0000 1.0000
>>
inv()：逆行列の計算
行列 a の逆行列の計算
逆行列の計算結果
行列 a × b の計算
計算結果：単位行列になっている

（9） 行列の変数エディター



ワークスペースの変数をダブルクリック



行列の変数エディターが開かれるので数値を編集。
編集終了後、右上の×をクリックしてエディターを終了

（10） 行列の演算機能概略

＋ － × ／	四則演算
Det	行列の値を計算
Inv	逆行列の計算
.*	要素単位の乗算
.^	要素単位のべき乗
‘	行列の転置
sin	各要素のsin

（11） 課題

1. 行列 $a = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$, $b = \begin{bmatrix} 11 & 22 & 33 \\ 44 & 55 & 66 \\ 77 & 88 & 99 \end{bmatrix}$ を設定せよ

2. $c = a \times b$ を求めよ

3. aとb の要素単位の積を求めよ

4. 行列 c の値を求めよ

5. 行列 c の逆行列を求めよ

MATLAB 6. 関数

（1） 概要

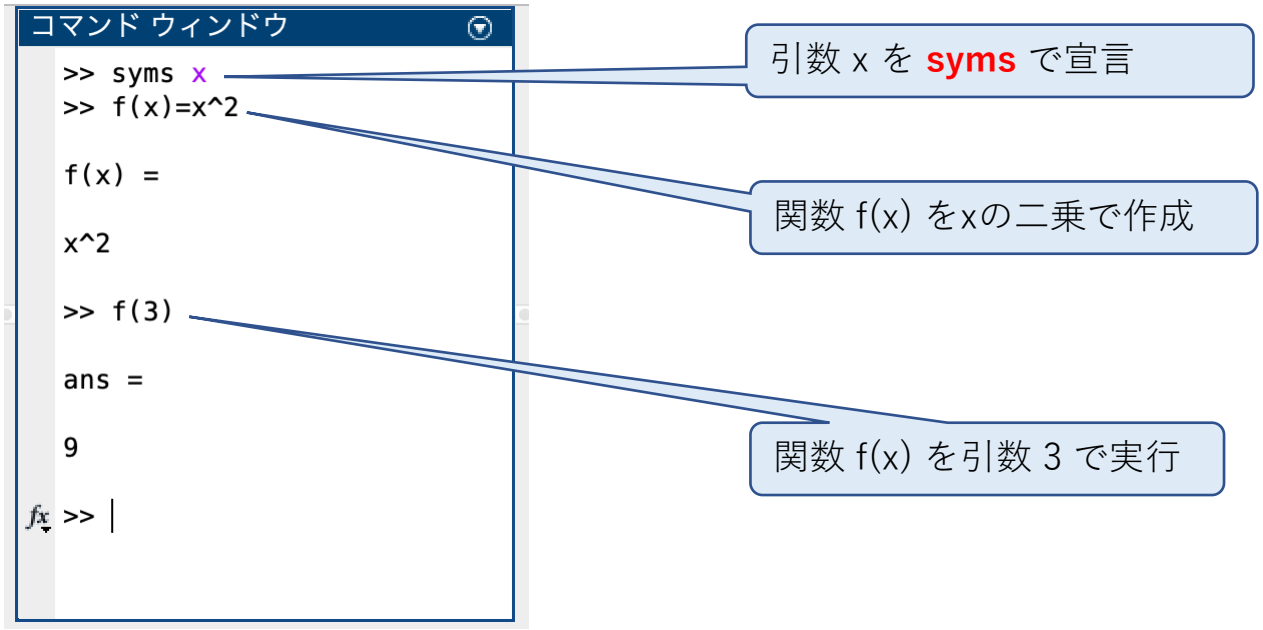
MATLABでは変数を宣言することと同様に関数を宣言することができる。変数と同様に＝の左側に新しい関数を記述し、右側に組み込み関数又は既に宣言されている関数等を記述する。

関数で使用される変数は、使用される前に syms によって宣言されている必要がある。

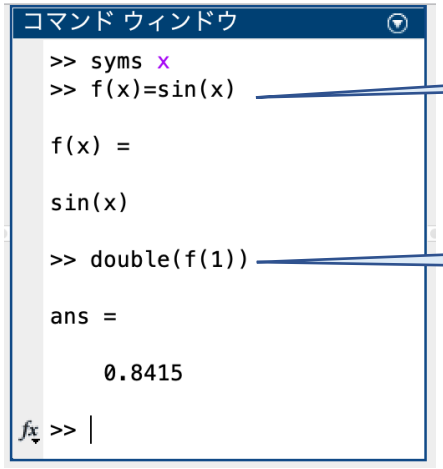
組込関数を含む関数を作成した場合、数値計算する場合 double を用いて結果を数値化する必要がある。

（2） 関数の作成と実行

関数を作成する前に、引数（下例では x ）を **syms** で宣言する。

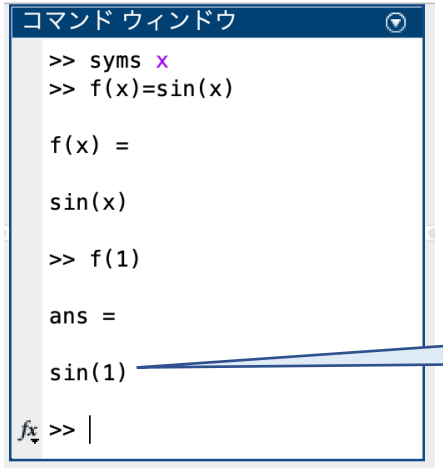


（3） 組込関数を含む関数の作成と実行



関数 f(x) をsin(x)で作成

関数 f(x) の実行結果を **double** により数値化



関数 f(x) の実行結果は **double** が無いので式で表示

（4） 課題

1. $\sqrt{x^2+y^2}$ を関数にして $x=3$、$y=4$ を計算せよ。

2. $\sqrt{\sin(x)^2+\cos(y)^2}$ を関数にして $x=1$、$y=2$ を計算せよ。

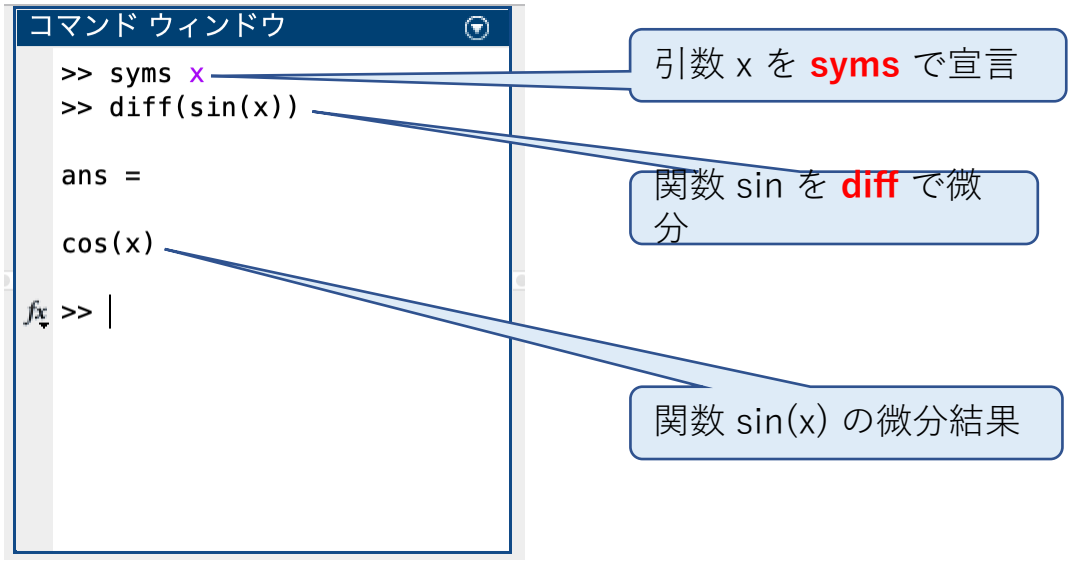
MATLAB 7. 微分計算

（1） 概要

MATLABでは、微分計算を **diff** により数式で行うことができる。
微分計算を行う前に、微分する関数の引数を **syms** によって宣言しておく必要がある。

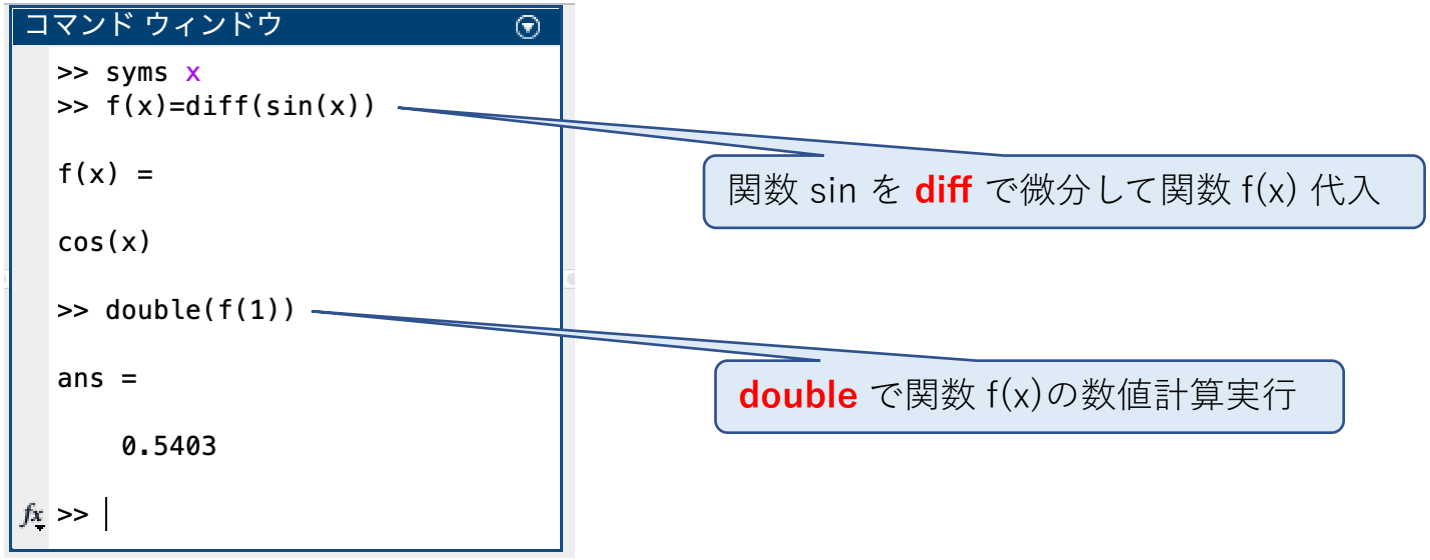
（2） sin の微分

関数を微分する前に、引数（下例では x ）を **syms** で宣言する。



（3） sin の微分と数値計算

微分結果を f(x) に代入し、数値計算を double で実行。



MATLAB 8. 積分計算

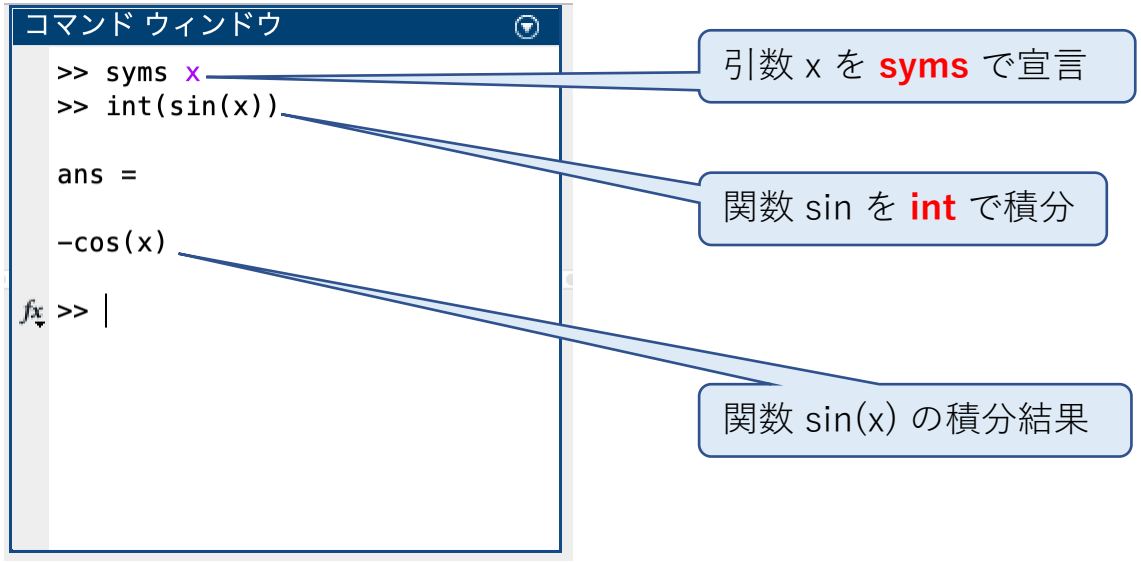
（1） 概要

MATLABでは、積分計算を **int** により数式で行うことができる。
微分計算を行う前に、微分する関数の引数を **syms** によって宣言しておく必要がある。

積分は数式で解を求める不定積分と、数値を求める定積分がある。

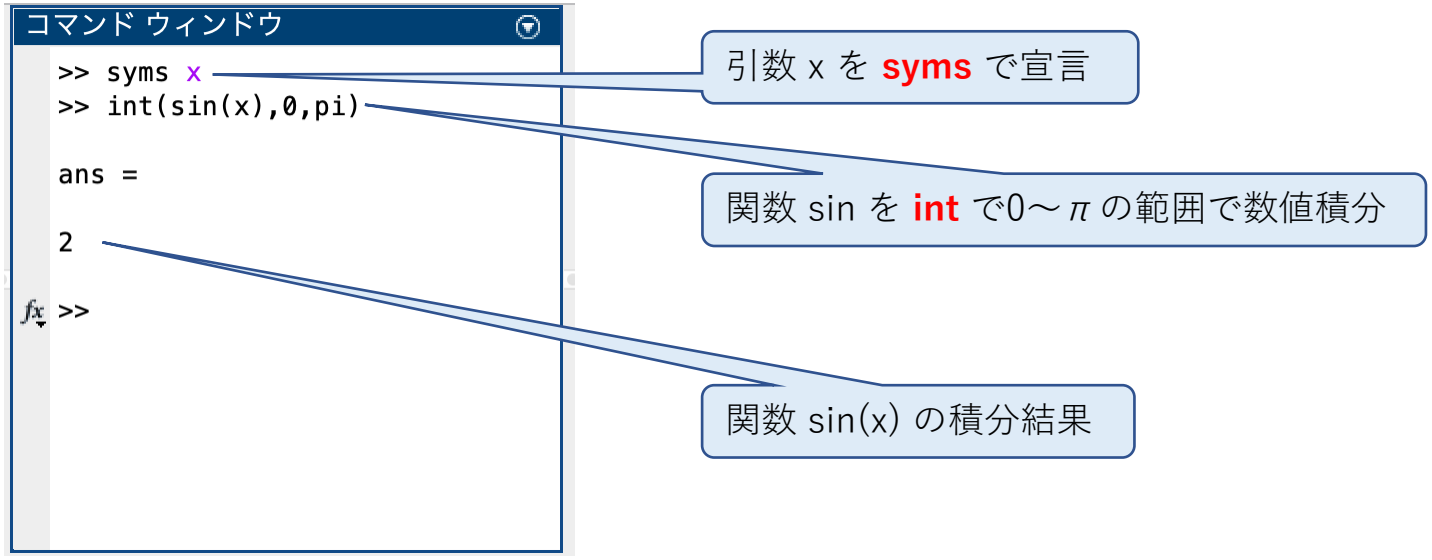
（2） 不定積分

関数を積分する前に、引数（下例では x ）を **syms** で宣言する。
int により関数を数式で積分することができる。



（3） 定積分

int により関数を範囲を決めて数値的に積分することができる。



MATLAB ９．Excelとの連携

（１） 概要

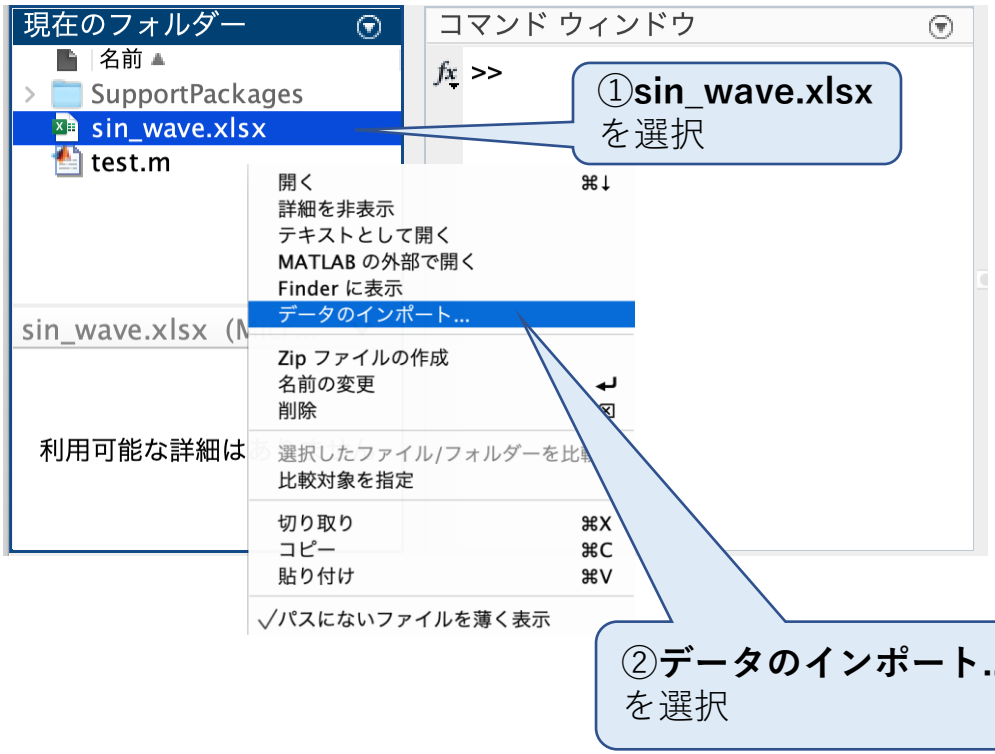
　MATLABでは、Excelのデータファイルをそのまま読み込み数値処理及びグラフ化することができる。

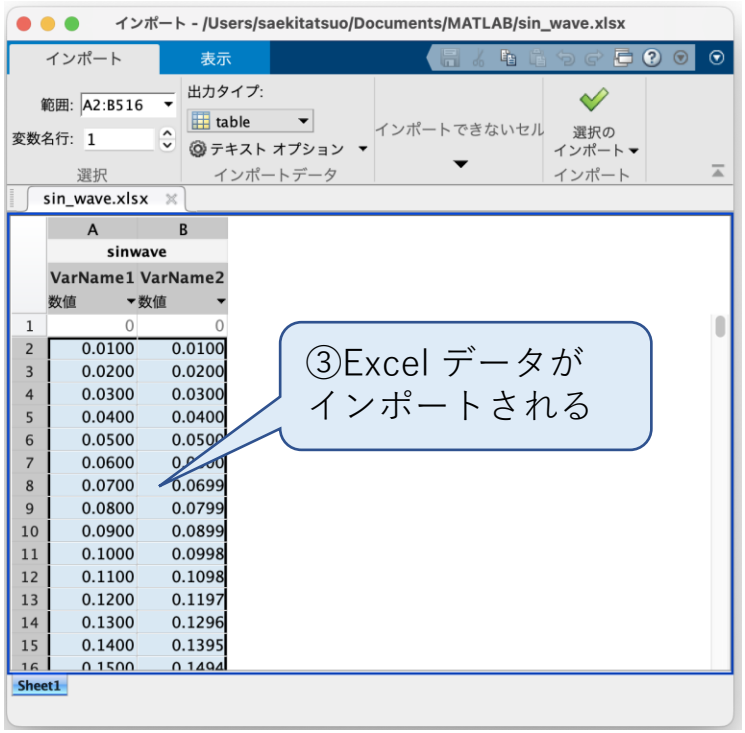
　これを実施するためには、前もってExcelデータファイルをMATLABの作業フォルダーに用意しておく必要がある。

　ここではsin波形のデータ sin_wave.xlsx を読み込んで処理を実施する。

（2） Excel データのインポート　1

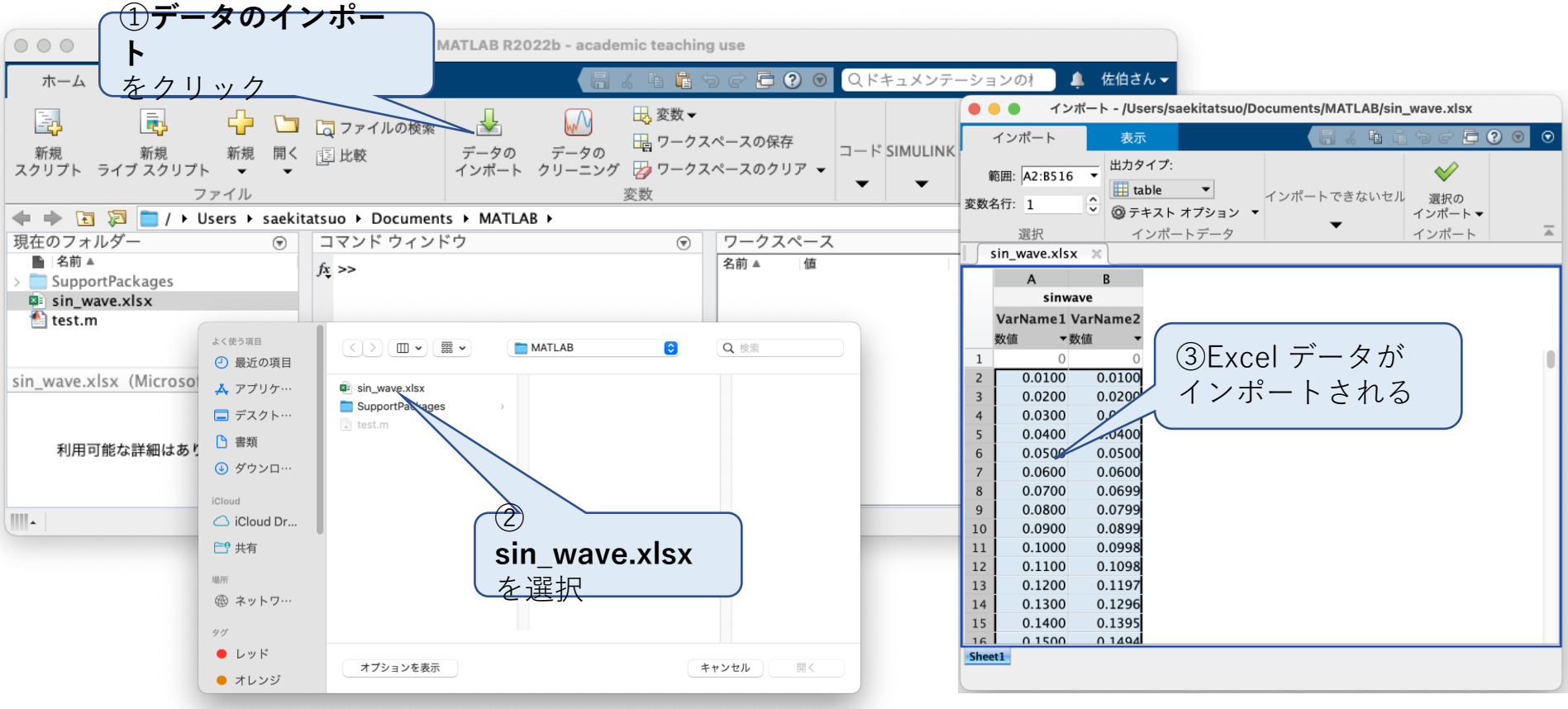
以下の手順でExcelデータをインポートできる。



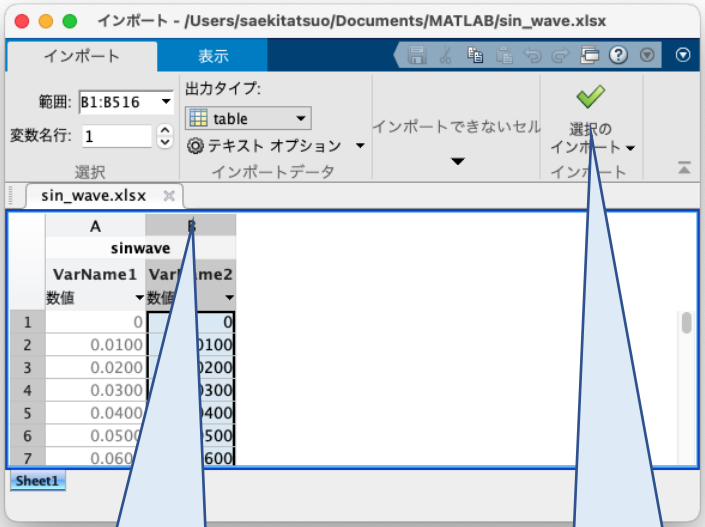


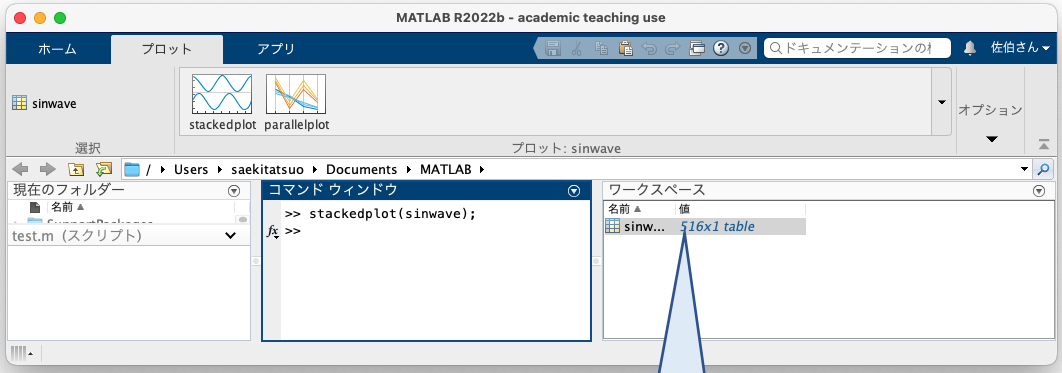
（3） Excel データのインポート　2

以下の手順でもExcelデータをインポートできる。



（4） インポートデータのワークスペースへの取り込み



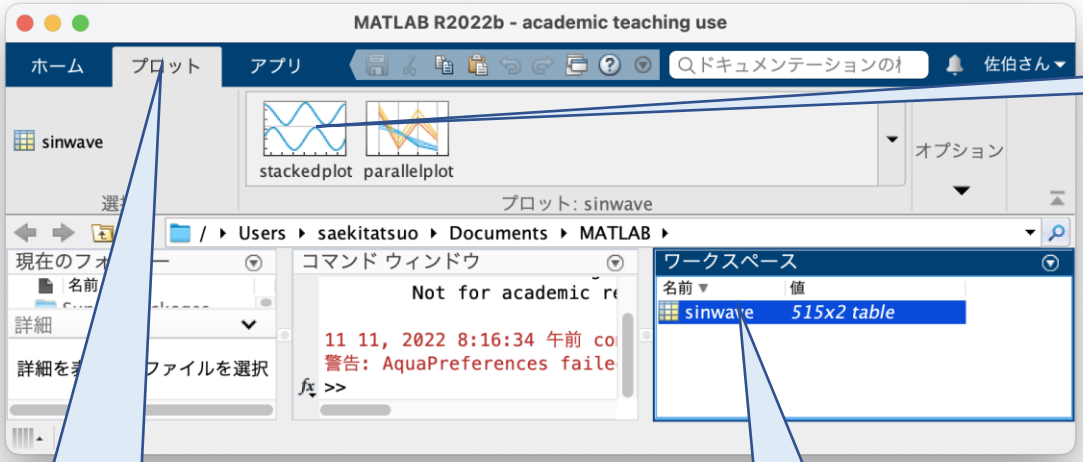


①プロットデータを選択

②**選択のインポート**をクリック

③ワークスペースに選択したデータが表示される

（5） ワークスペースのデータのグラフ化

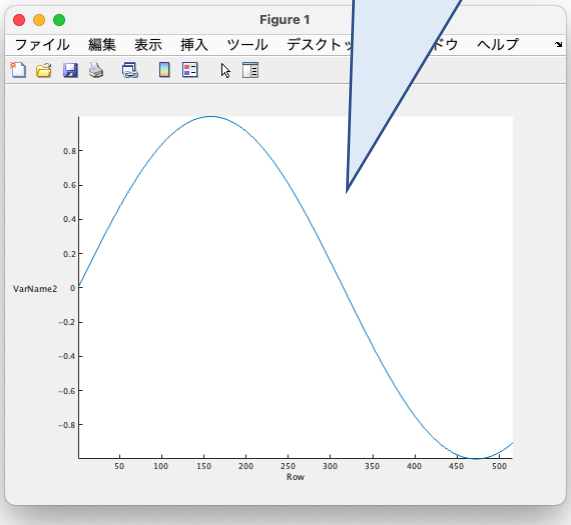


①**プロット**をクリック

②ワークスペースの
データをクリック

③**stackedplot**をクリック

④データのグラフが表示される



MATLAB 10. M-ファイル

（1） 概要

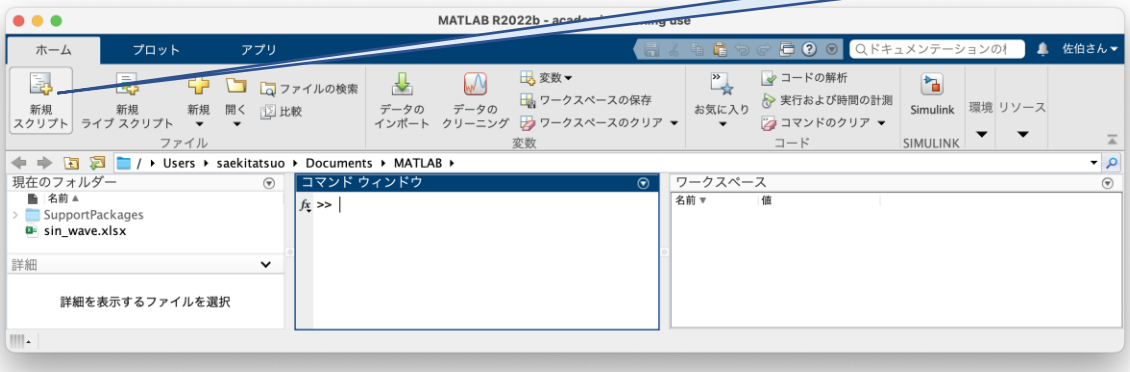
MATLABの数値処理はコマンドラインで実行できるが、コマンドラインとしてスクリプトを一旦入力すると、後で一部修正することができない、同じ処理を繰り返ことができない等不便なことがある。

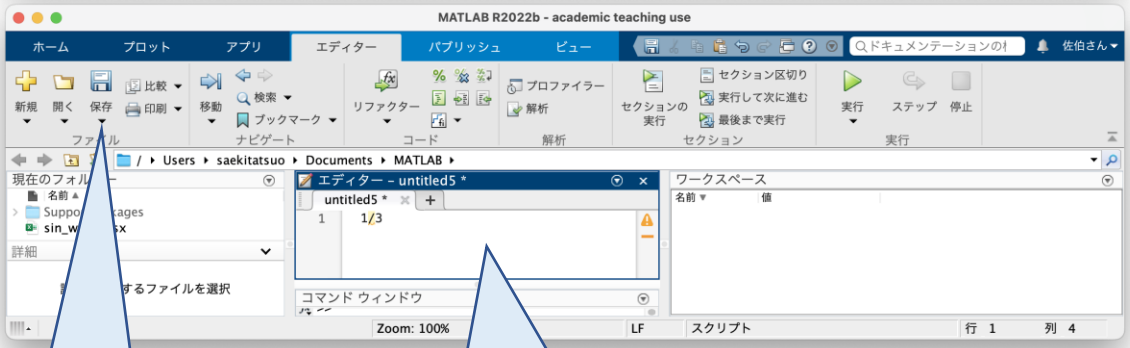
この問題を解決するために、MATLABにはスクリプトをM-ファイルとして保存する機能がある。

尚、M-ファイルには、M-ファイルと関数M-ファイルの２種類がある。

（2） M-ファイルの新規作成と保存

①**新規スクリプト**をクリック

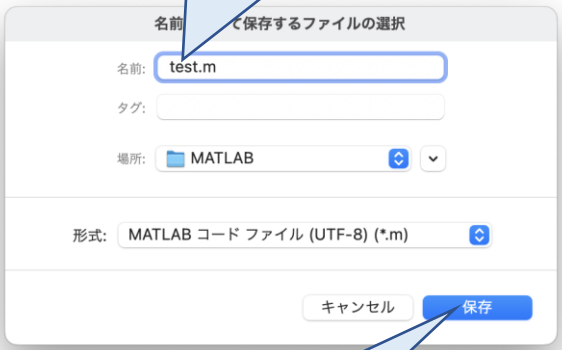




③**保存**をクリック

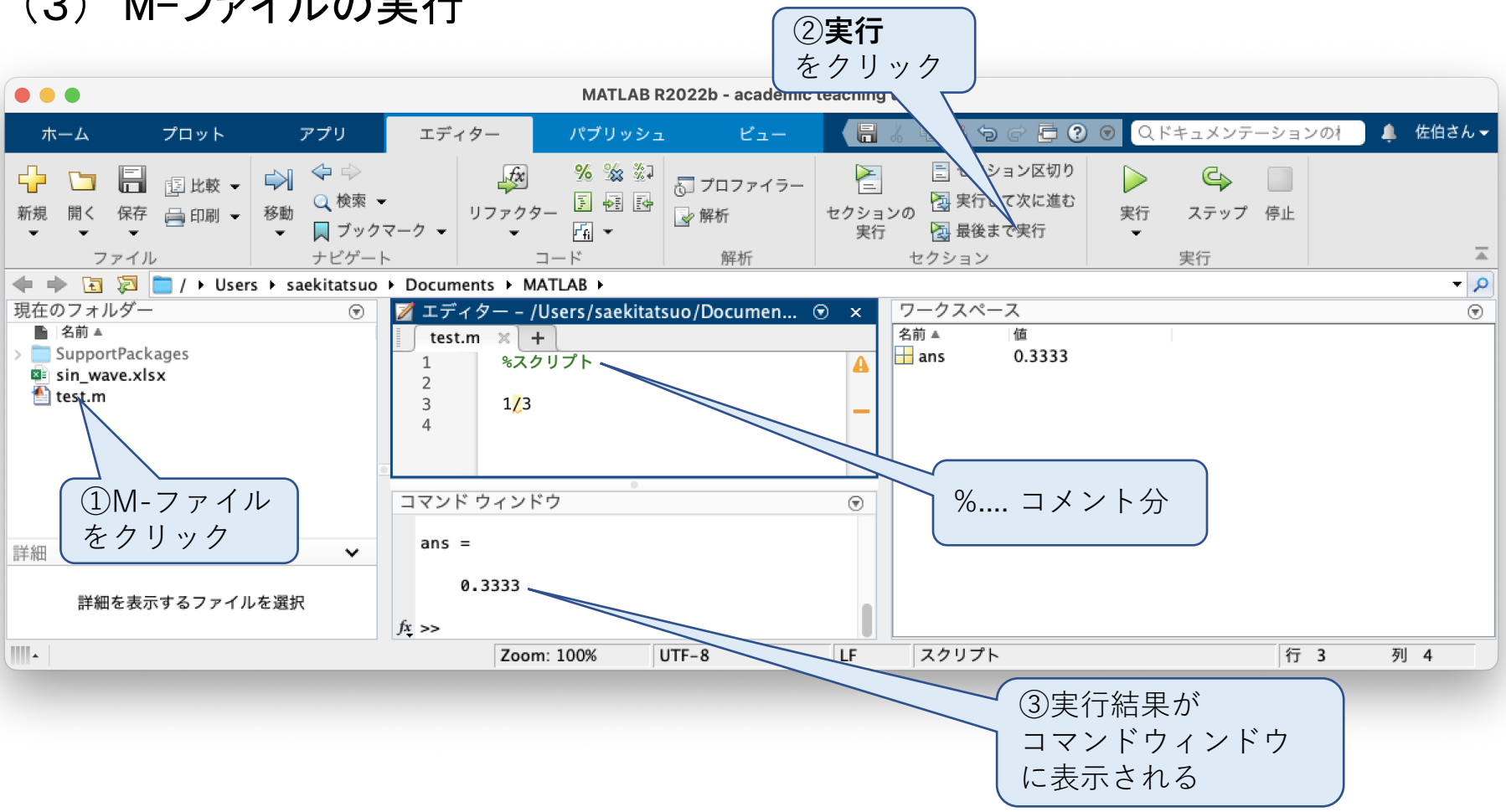
②スクリプトの入力

④M-ファイル名を入力



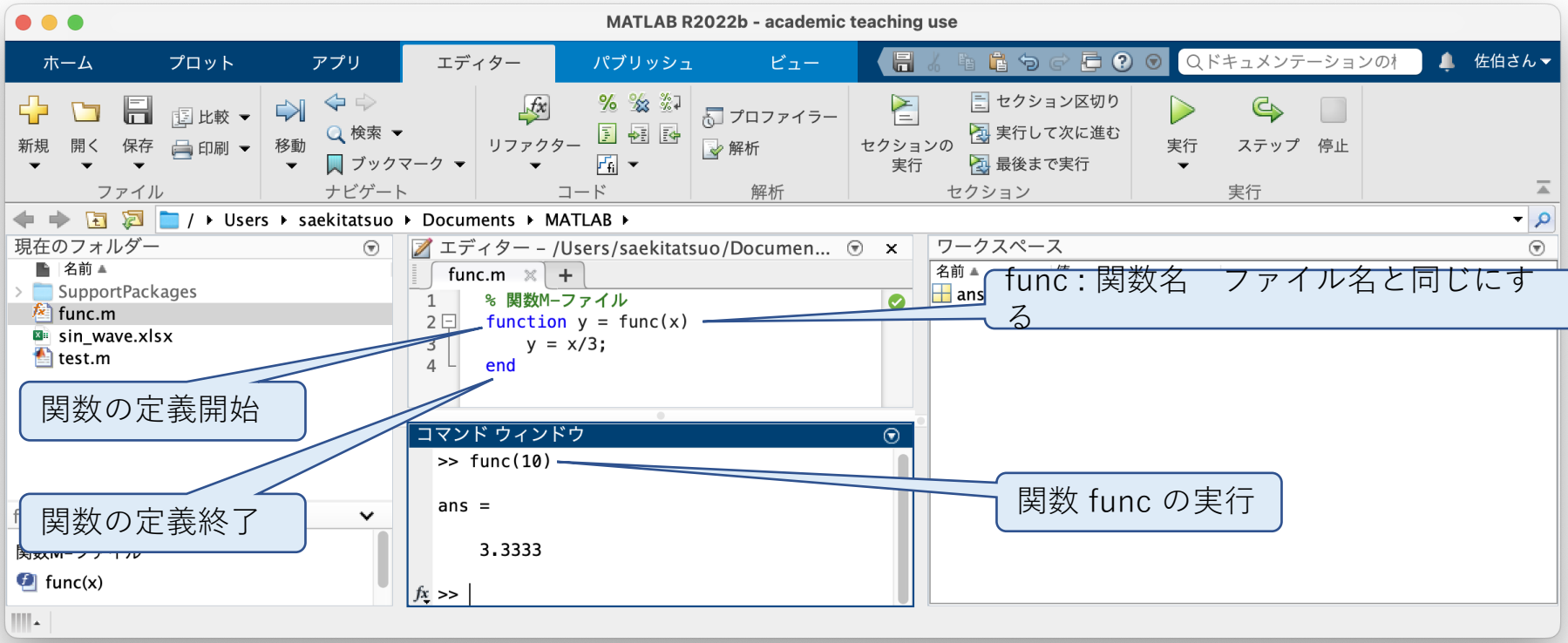
⑤**保存**をクリック

（3） M-ファイルの実行


②実行
をクリック
MATLAB R2022b - academic teaching
ホーム
プロット
アプリ
エディター
パブリッシュ
ビュー
佐伯さん
新規
開く
保存
比較
印刷
移動
検索
ブックマーク
ファイル
ナビゲート
リファクター
コード
プロファイラー
解析
解析
セクションの
実行
実行して次に進む
最後まで実行
セクション
実行
ステップ
停止
実行
/ ▸ Users ▸ saekitatsuo ▸ Documents ▸ MATLAB ▸
現在のフォルダー
名前
SupportPackages
sin_wave.xlsx
test.m
①M-ファイル
をクリック
詳細
詳細を表示するファイルを選択
エディター - /Users/saekitatsuo/Documen...
test.m
1 %スクリプト
2
3 1/3
4
%.... コメント分
コマンド ウィンドウ
ans =
0.3333
>>
ワークスペース
名前
値
ans
0.3333
③実行結果が
コマンドウィンドウ
に表示される
Zoom: 100%
UTF-8
LF
スクリプト
行 3
列 4

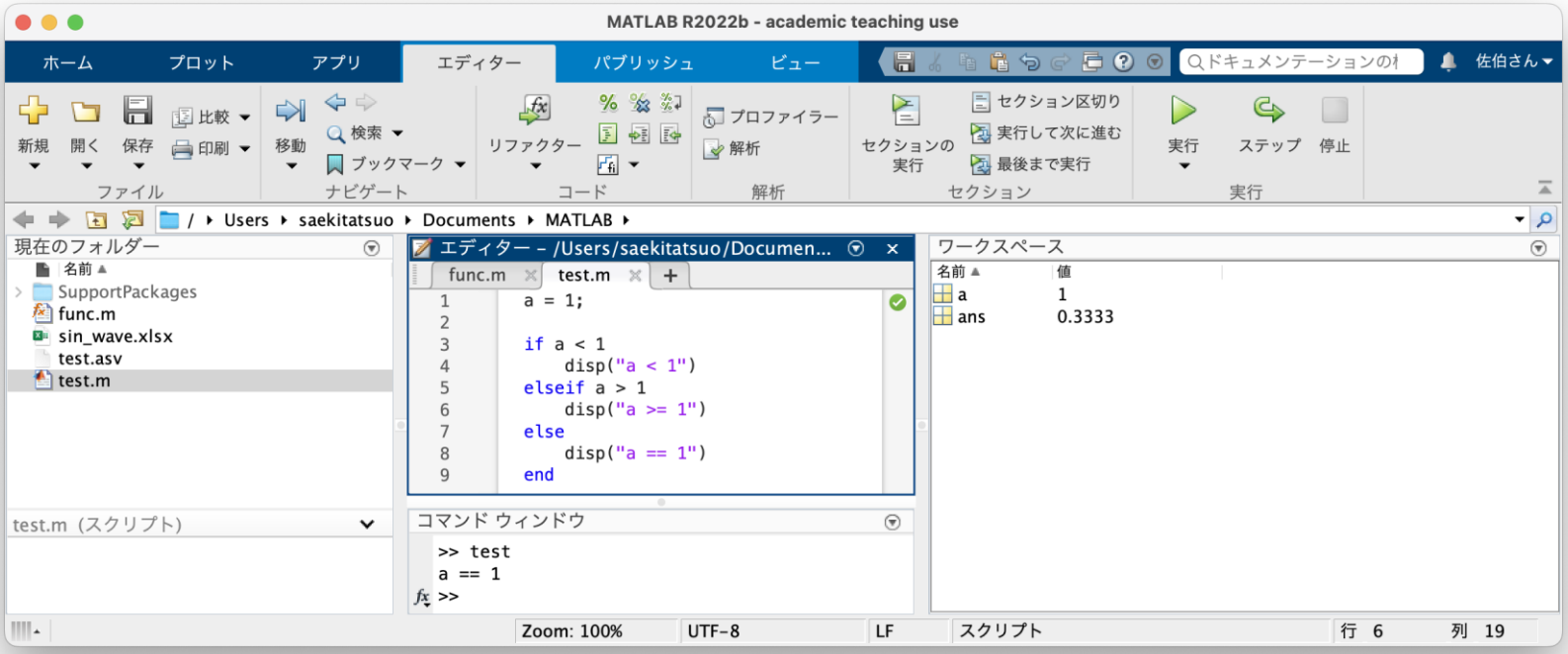
（4） 関数M−ファイル

　スクリプトで呼び出しができる関数を作成し、関数M−ファイルとして保存することができる。



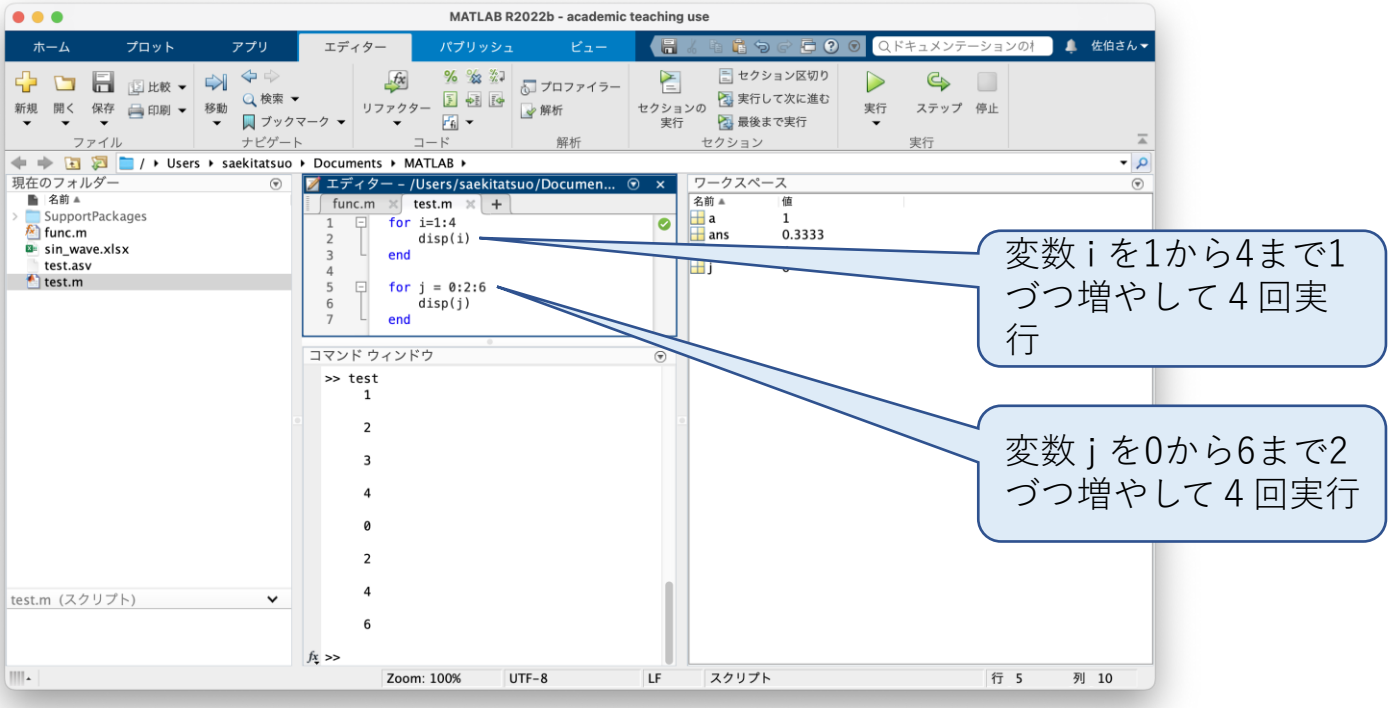
（5） M-ファイルの制御構造　比較演算 if

　他のプログラム言語のようにM-ファイルのスクリプトで if, else if, else を使うことができる。比較演算の最後は end で終了する
　比較演算子には ==, <, >, <=, >=, ~= がある。



（6） M-ファイルの制御構造　繰り返し for

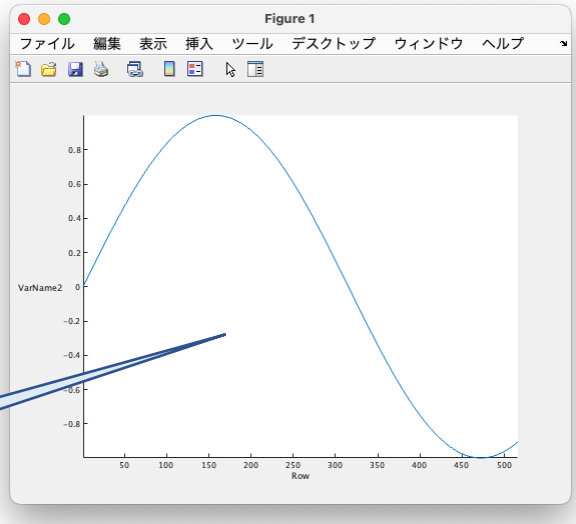
　他のプログラム言語のようにM-ファイルのスクリプトで for, while を使うことができる。繰り返しの最後は end で終了する。



MATLAB 11. グラフィックス

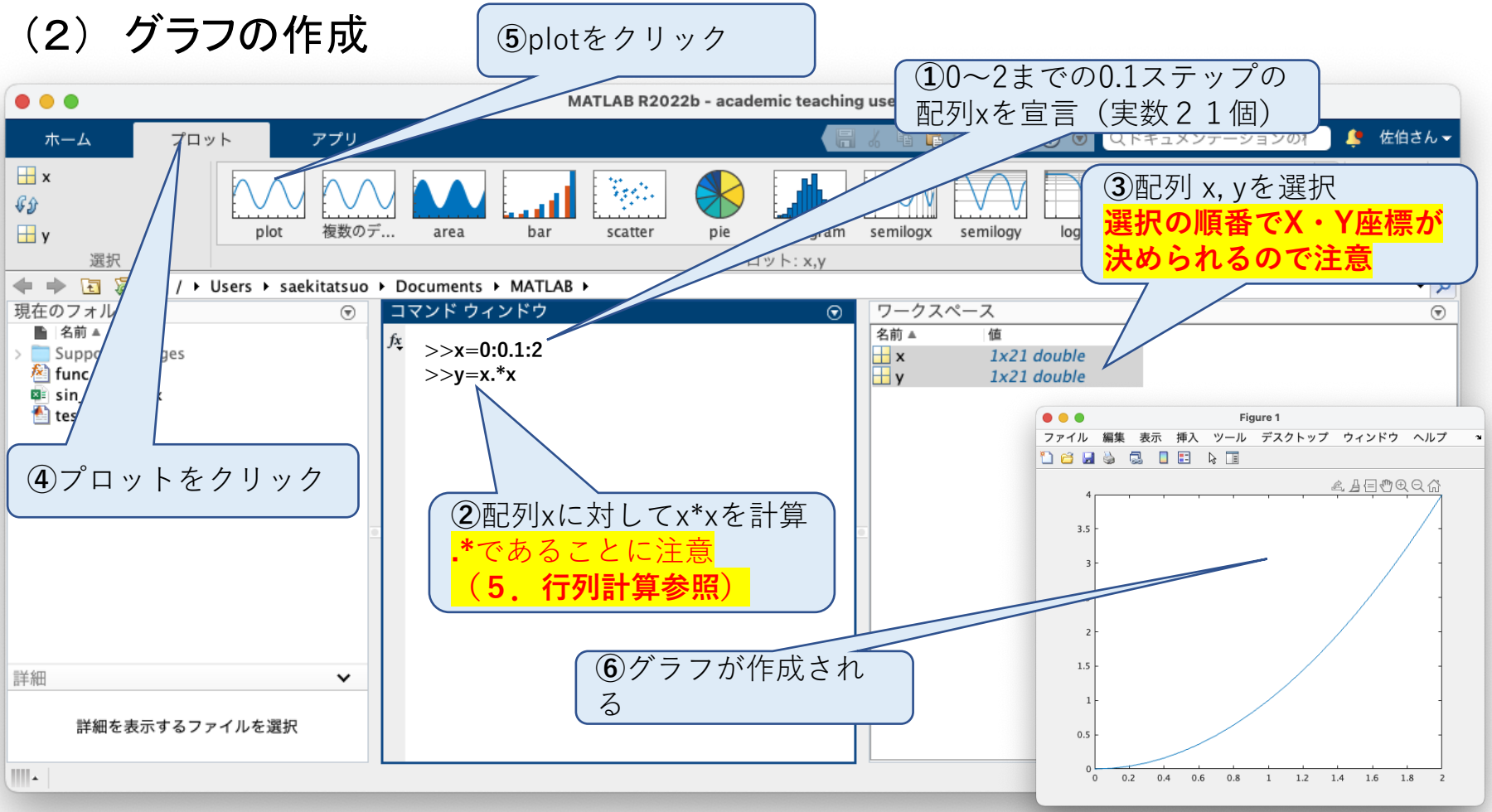
（1） 概要

　MATLABは、演算結果をグラフに描いて可視化する機能がある。
又演算結果をアニメーションにして動きのある状態で観察することも可能となっている。
　MATLABはVisualizationを謳い文句にしていた時期があったくらい可視化の機能が充実していて、
多くのコマンド、関数群が用意されている。



グラフ例

（2） グラフの作成


⑤plotをクリック
①0～2までの0.1ステップの
配列xを宣言（実数２１個）
③配列 x, yを選択
選択の順番でX・Y座標が
決められるので注意
MATLAB R2022b - academic teaching use
ホーム
プロット
アプリ
佐伯さん
x
y
選択
plot
複数のデ...
area
bar
scatter
pie
semilogx
semilogy
ロット: x,y
Users ▸ saekitatsuo ▸ Documents ▸ MATLAB ▸
現在のフォル
名前
コマンド ウィンドウ
>>x=0:0.1:2
>>y=x.*x
ワークスペース
名前
値
x
1x21 double
y
1x21 double
④プロットをクリック
②配列xに対してx*xを計算
.*であることに注意
（５．行列計算参照）
⑥グラフが作成され
る
詳細
詳細を表示するファイルを選択
Figure 1
ファイル 編集 表示 挿入 ツール デスクトップ ウィンドウ ヘルプ

（7） 課題

１．０～πの範囲でステップ0.01でsin関数のグラフを作成せよ。

２．０～10の範囲でステップ0.1で y=3*x*x*x+2*x*x+x+1のグラフを作成せよ。

３．０～2πの範囲でステップ0.01でsin*sinのグラフを作成せよ。

SIMLINK １．制御システムとは

（１） 自動車のオートクルーズコントローラで考える制御システム

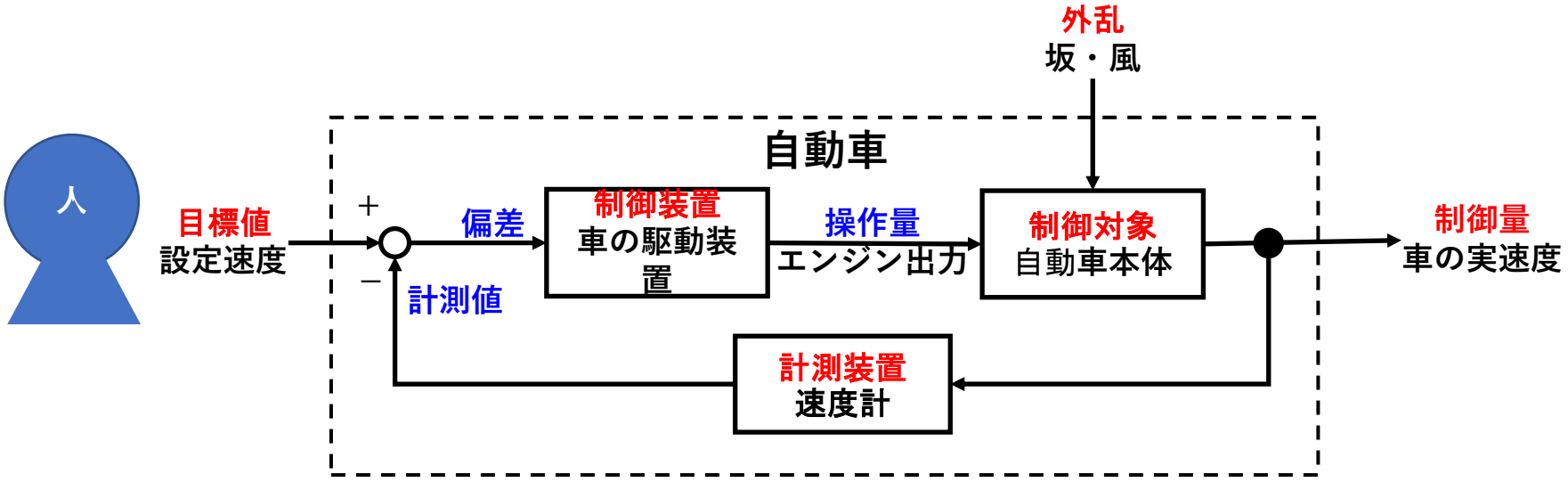
自動車のオートクルーズコントローラは、ドライバーが車の速度を設定すると、車は自動で加減速して設定速度で走行するシステムである。

ここで、ドライバーによって設定された速度を**目標値**、車の実際の速度を**制御量**、車を**制御対象**、エンジン等の駆動装置を**制御装置**、速度を計測する物を**検出装置**と言う。

又、車が一定速度で走る場合、平坦地であれば問題ないが上り坂・下り坂又風等によって加減速が発生するが、これを**外乱**と言う。

上記内容の制御ブロック線図を次頁に示す。

（2） 自動車のオートクルーズコントローラの制御ブロック線図



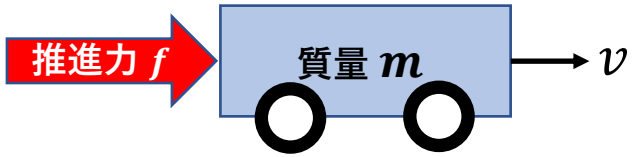
ブロック図中、操作量はエンジンの出力であり、計測値は実際の車の速度、偏差は目標値と計測値の差分であり、以下の式で表される。

偏差＝目標値－計測値

制御システムでは、この偏差が0となるよう制御される。

（3） 自動車の制御例　1

例1：抵抗の無い自動車

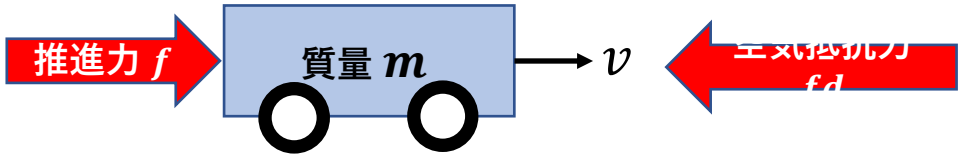


右図のように抵抗の無い車の速度を制御する場合、速度0から発進して設定値100[km/h]を目標に加速した場合を考える。

推進力は偏差に比例するもとのすると、始めは偏差が大きいため大きな推進力で加速するが、速度が上がるに従い偏差が減少（推進力が減少）し、速度が100[km/h]に達した時に力が0となり後は惰性で速度100[km/h]を維持することになる。

（4） 自動車の制御例 ２

例2：空気抵抗の有る自動車



右図のように空気抵抗が有る車の速度を制御する場合、速度0から発進して設定値100[km/h]を目標に加速するのは前頁と同じである。

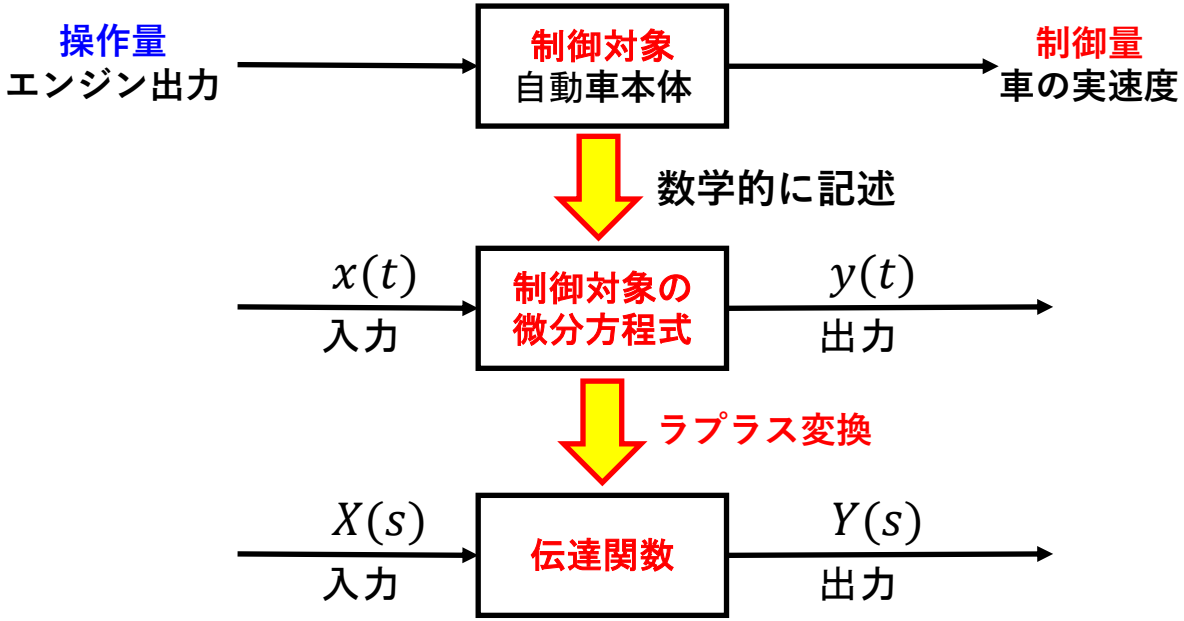
この場合では速度が上がるに従って空気抵抗力が速度の二乗に比例して増大するため、偏差が0（速度が100[km/h] ）になる前にある速度で推進力と空気抵抗力が釣り合ってしまい、いつまで待っても目標値100[km/h]に達しないことになる。

制御システムはこのような問題を解決するために発達してきた。

（5） 車の伝達関数

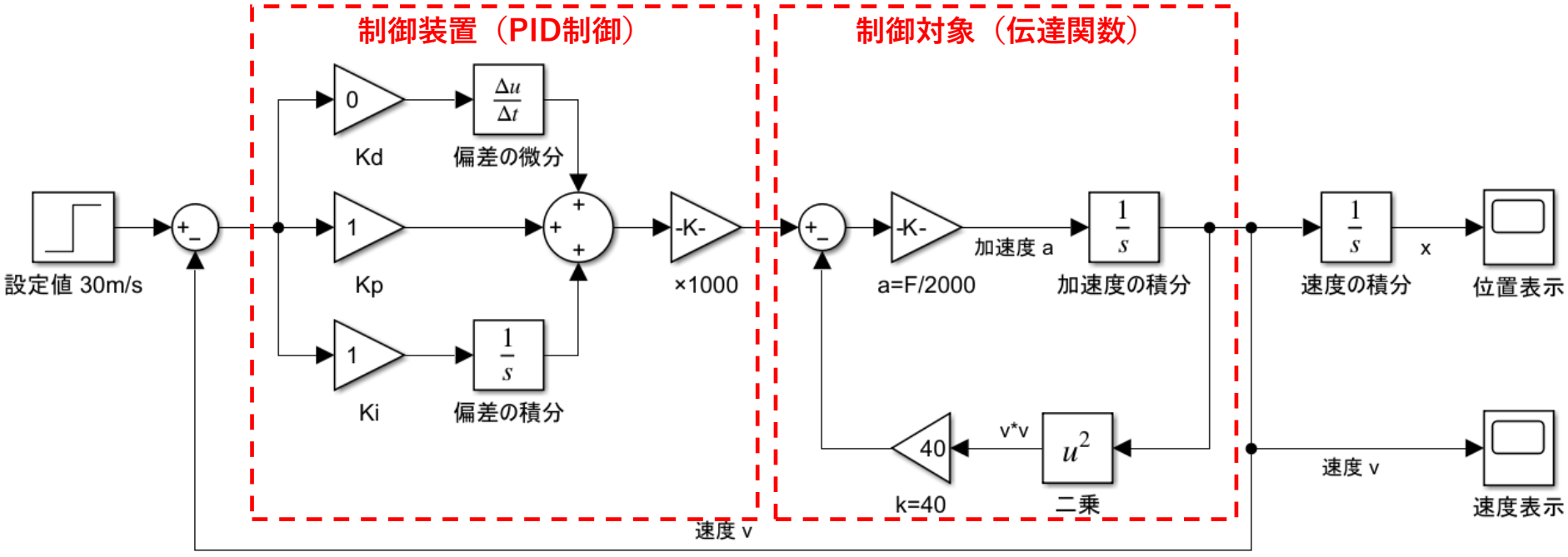
制御対象を数学的に記述したものが制御対象の微分方程式となり、これをラプラス変換すると伝達関数が得られる。

SIMULINKでは、制御対象の代わりに伝達関数を用いて記述する。



（6） SIMULINKにおけるブロック線図例

SIMULINKと伝達関数を用いれば、複雑な制御システムでも下図のように視覚的に設計し、シミュレーションを実施できる。



SIMLINK 2. ラプラス変換と伝達関数

（1） 概要

SIMULINKでは、制御対象をラプラス変換した伝達関数でモデルを作成し、数値計算でシミュレーションを実施する機能を持つ。

ラプラス変換とは変数変換の一種であり、線形微分方程式を解析するために用いられる。

一般に線形微分方程式を解析的に解くのは非常に難しいが、ラプラス変換すれば単純な四則演算で計算することができる。又、ラプラス変換はラプラス変換表を見れば比較的に容易に行うことができる。

尚、上記計算結果を逆ラプラス変換したものが微分方程式の解となる。この逆変換は非常に難しいが、SIMULINKを用いて数値計算を行えば逆ラプラス変換は不要となる。SIMULINKで計算するために、伝達関数を求める必要がある。

（2） 微分方程式の解法

解析的解法

1．微分方程式を解析的に求める	大変難しい

ラプラス変換

1．微分方程式をラプラス変換	簡単
2．上記の結果から伝達関数を求める	簡単
3．上記の結果を逆ラプラス変換	難しい

SIMULINK

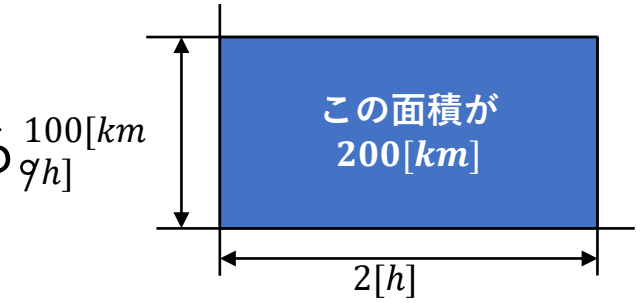
1．微分方程式をラプラス変換	簡単
2．上記の結果から伝達関数を求める	簡単
3．SIMULINKで数値シミュレーション	簡単

（3） 運動の微分と積分

①速度100[km/h]で2時間走ると、走行距離は200[km]となり、計算式は下式となる。図中の青い面積が走行距離となる

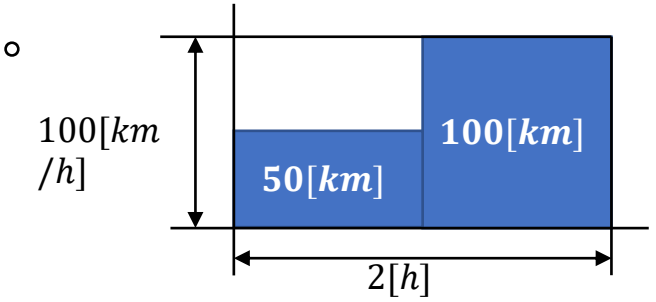
$$100 \times 2 = 200[km]$$

走行距離xとは速度v×時間tで計算される



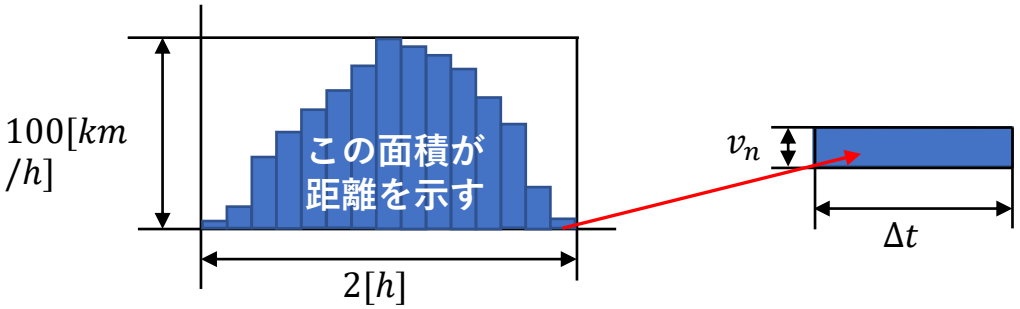
②速度50[km/h]で1時間、速度100[km/h]で1時間走ると、走行距離は150[km]となり、計算式は下式となる。

$$50 \times 1 + 100 \times 1 = 150[km]$$



③速度が時間的に変化する場合、2時間のなかで速度が細かく変化するため、走行距離 x は速度 v 毎に$v \times \Delta t$ を求めその合計で計算する必要があり、計算式は下式となる。尚Δtは微小時間（短い時間）を示す。

走行距離 $x = v_1 \times \Delta t + v_2 \times \Delta t + \cdots + v_n \times \Delta t$



④上記式を積分で表現すると下式となり、これは速度を時間で積分すると距離になることを示している。

走行距離 $\boldsymbol{x} = v_1 \times \Delta t + v_2 \times \Delta t + \cdots + v_n \times \Delta t = \sum_{i=1}^{n} v_i \times \Delta t = \int \boldsymbol{v}\, \boldsymbol{dt}$

⑤積分と逆の計算をするのが微分であり、下式となる

$$x = \int v dt \quad \Leftrightarrow \quad \frac{dx}{dt} = v$$

上式は速度vを積分すると距離xとなり、又距離xを微分すると速度vになることを示している。

⑥ここまでは距離と速度を考えていたが、速度vと加速度aも同じ関係にあり、同様に計算できる

$$v = \int a dt \quad \Leftrightarrow \quad \frac{dv}{dt} = a$$

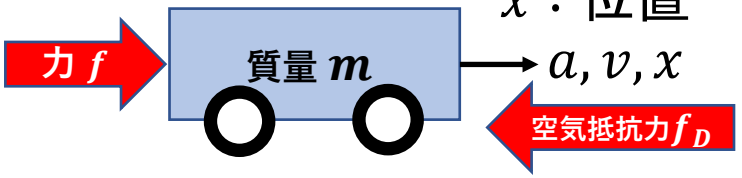
上式は加速度aを積分すると速度vとなり、又速度vを微分すると加速度aになることを示している。

（4） 微分方程式

物理現象を微分方程式を使って数式で表すことができる。

a：加速度
v：速度
x：位置

例：質量mの車を力fで押す場合



物体に働く力の合計 $\boldsymbol{F = ma} = f - f_D$

$ma = f - f_D$　ここで空気抵抗力 $f_D = Kv^2$ ：$K = \frac{1}{2}\rho C_d S$ とすると

$$ma = \boldsymbol{m\frac{dv}{dt} = f - Kv^2}$$ ：速度の微分方程式

$$ma = \boldsymbol{m\frac{d^2x}{dt^2} = f(t) - K\left(\frac{dx}{dt}\right)^2}$$ ：位置の微分方程式

（5） ラプラス変換

右表を用いて微分方程式をラプラス変換することができる。

ラプラス変換する場合には時間的に変化する量、例えば力fや位置x等の変数を、係数等の固定値と区別するために$f(t), x(t)$と記述する。

変換前	変換後
$f(t)$	$F(s)$
1	$1s$
$\int f(t)$	$\frac{F(s)}{s}$
$\frac{df(t)}{dt}$	$sF(s)-f(0)$
$\frac{d^2f(t)}{dt^2}$	$s^2F(s)-sf(0)-f(0)$
$f(t)\times g(t)$	$\frac{1}{2\pi j}\int_{Br} F(s-\sigma)G(\sigma)d\sigma$
t	$\frac{1}{s^2}$

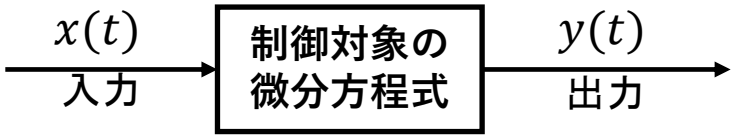
変換例

ラプラス変換

$$x(t)+3+\int x(t)\,dx+\frac{dx(t)}{dt}+\frac{d^2x(t)}{dt^2} \Rightarrow X(s)+3s+\frac{X(s)}{s}+sX(s)-f(0)+s^2X(s)-sf(0)-f(0)$$

（6） 伝達関数

制御システムの解析とは、下図のようにある制御対象へ入力$x(t)$を加えた時、どのような出力$y(t)$となるかを調べることである。例えば水槽に熱量を加えた時水温が何度上昇するか、ある物体を押した時どのような動きをするか等。



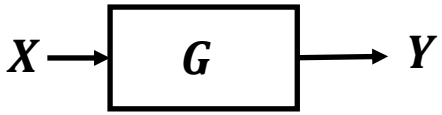
上記制御システムをラプラス変換すると、下図のように変換される。

伝達関数は次式で求めることができる。 $\boldsymbol{G(s) = Y(s)/X(s)}$

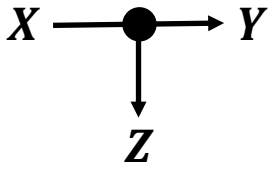
（7） 伝達関数の演算

伝達関数をブロック図として表記すると、下記のように四則演算で計算できる。

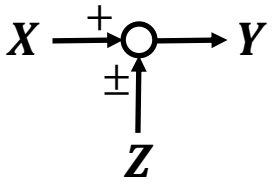
$X \times G = Y$



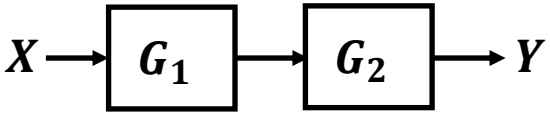
$X = Z = Y$



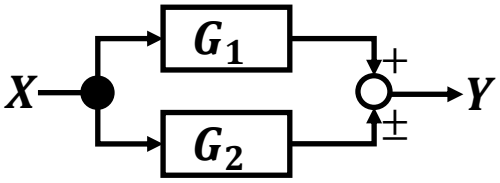
$X \pm Z = Y$



$X \times G_1 \times G_2 = Y$

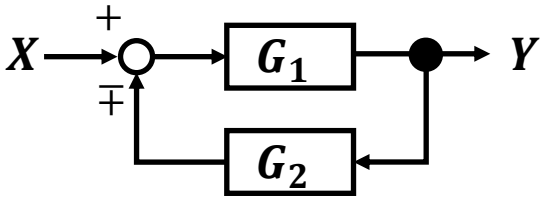


$X \times G_1 \pm X \times G_2 = Y$



次頁参照

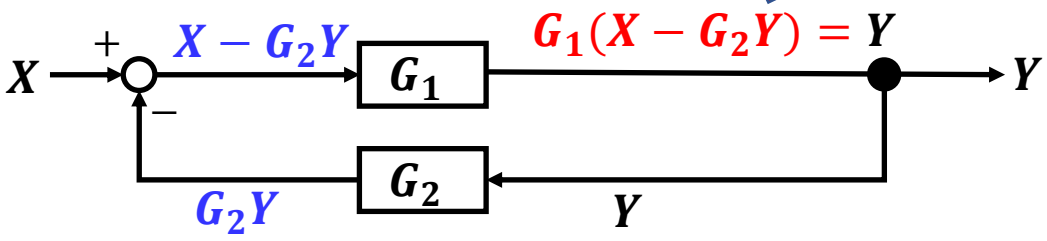
$$X \times \left(\frac{G_1}{1 \pm G_1 \times G_2} \right) = Y$$



フィードバック

（8） フィードバックの伝達関数の計算

フィードバックの伝達関数はブロック線図から
下記のように四則演算で計算できる。



$$G_1(X - G_2Y) = Y$$

$$G_1X - G_1G_2Y = Y$$

$$G_1X = Y(1 + G_1G_2)$$

$$\left(\frac{G_1}{1 + G_1G_2}\right)X = Y$$

$$G = \frac{G_1}{1 + G_1G_2} = \frac{Y}{X}$$

（9） 逆ラプラス変換

伝達関数は下表を用いて微分方程式の解を求めることができるが、かなり難しい。

尚、**SIMULINKを用いれば数値シミュレーションを実行できるため、本変換は必要ない。**

変換前	変換後
$\frac{1}{s+a}$	e^{-at}
$\frac{\omega}{s^2+\omega^2}$	$sin(\omega t)$
$\frac{s}{s^2+\omega^2}$	$cos(\omega t)$
$\frac{n!}{s^{n+1}}$	t^n
$\frac{\omega}{(s+a)^2+\omega^2}$	$e^{-at}sin(\omega t)$
$\frac{s+a}{(s+a)^2+\omega^2}$	$e^{-at}cos(\omega t)$

（10） 空気抵抗の無い車を力fで加速した時の加速度変化の伝達関数

物体に働く力の合計[N] $= \boldsymbol{F = ma}$

$$f(t) = ma(t)$$

ラプラス変換

$$F(s) = mA(s)$$

車に対して力fという入力を加えた時の速度vという出力を求める

出力

$$\text{伝達関数：} \boldsymbol{G(s) = \frac{A(s)}{F(s)} = \frac{1}{m}}$$

入力

（11） 空気抵抗の無い車を力fで加速した時の速度変化の伝達関数

物体に働く力の合計[N] $= \boldsymbol{F = ma}$

$ma(t) = f(t)$　　$m\dfrac{dv(t)}{dt} = f\ (t)$

⇩ ラプラス変換

$msV(s) = F(s)$

伝達関数：$\boldsymbol{G(s) = \dfrac{V(s)}{F(s)} = \dfrac{1}{m} \times \dfrac{1}{s}}$

出力

入力

車に対して力fという入力を加えた時の速度vという出力を求める

（12） 空気抵抗の無い車を力fで加速した時の位置変化の伝達関数

物体に働く力の合計[N] $= \boldsymbol{F = ma}$

$ma(t) = f(t)$　　$m\dfrac{d^2x(t)}{dt^2} = f\ (t)$

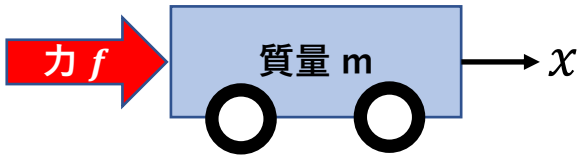
ラプラス変換

$ms^2X(s) = F(s)$

伝達関数：$\boldsymbol{G(s) = \dfrac{X(s)}{F(s)} = \dfrac{1}{m} \times \dfrac{1}{s^2}}$

出力

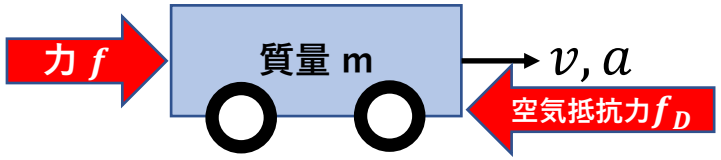
入力



車に対して力fという入力を加えた時の速度vという出力を求める

（13） 空気抵抗の有る車を力fで加速した時の速度変化の伝達関数

物体に働く力の合計[N] $= \boldsymbol{F = ma}$ = 力－空気抵抗力



$$ma(t) = f(t) - f_D(t) = f(t) - K(v(t))^2 \quad K = \frac{1}{2}\rho C_d S$$

ここで計算を簡単にするため空気抵抗を$Kv(t)$と置いて、速度を求める式は次式となる。

$$m\frac{dv(t)}{dt} = f(t) - Kv(t)$$

⇩ ラプラス変換

$$msV(s) = F(s) - KV(s) \rightarrow msV(s) + KV(s) = F(s) \rightarrow V(s) = \frac{F(s)}{m} \times \frac{1}{s + K/m}$$

伝達関数：$\boldsymbol{G(s) = \frac{V(s)}{F(s)} = \frac{1}{m} \times \frac{1}{s + K/m}}$

（14） 空気抵抗の有る車を力fで加速した時の速度変化の伝達関数―2

前頁では空気抵抗力を$Kv(t)$としたが、これをKv^2とした場合のラプラス変換を行う。

$$m\frac{dv(t)}{dt} = f(t) - Kv^2(t)$$

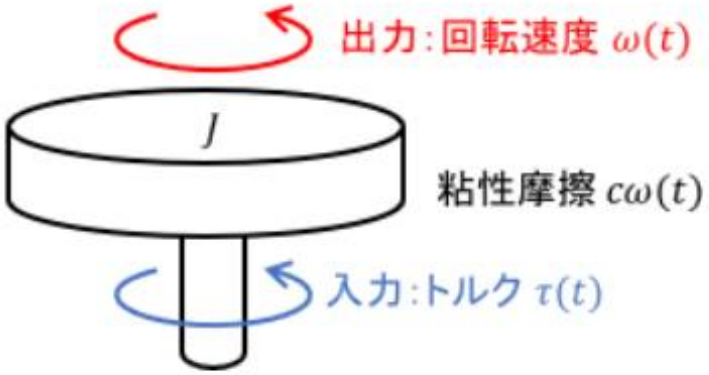
⇩ ラプラス変換

$$msV(s) = F(s) - \frac{K}{2\pi j}\int_{Br} V(s-\sigma)V(\sigma)d\sigma$$

この場合はラプラス変換で伝達関数を求めることは、非常に困難であることがわかる。

（15） 回転運動系の伝達関数

慣性モーメント:$\boldsymbol{J}$ ⇔ $\boldsymbol{m}$:質量
角速度:$\boldsymbol{\omega}$ ⇔ $\boldsymbol{v}$:速度
トルク（モーメント）:$\boldsymbol{\tau}$ ⇔ $\boldsymbol{F}$:力



$$J\frac{d\omega(t)}{dt} = \tau(t) - c\omega(t) \quad \Longleftrightarrow \quad \left(\text{参考：} ma(t) = m\frac{dv(t)}{dt} = f(t) - kv(t)\right)$$

⇩ ラプラス変換

$$Js\Omega(s) = \mathrm{T}(s) - c\Omega(s)$$

$$\Omega(s)(Js + c) = \mathrm{T}(s)$$

$$\text{伝達関数：}\boldsymbol{G(s)} = \frac{\boldsymbol{\Omega(s)}}{\boldsymbol{T(s)}} = \frac{\boldsymbol{1}}{\boldsymbol{J}} \times \frac{\boldsymbol{1}}{\boldsymbol{s + c/J}}$$

（16） ばねダンパ系の伝達関数

$$ma(t) = f(t) - cv(t) - kx(t)$$

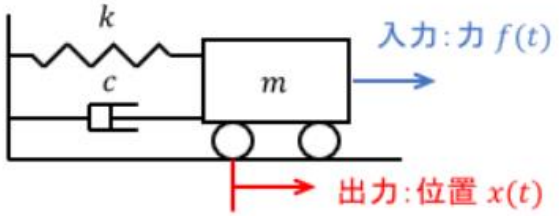
$$m\frac{d^2x(t)}{dt^2} = f(t) - c\frac{dx(t)}{dt} - kx(t)$$

ラプラス変換

$$ms^2X(s) = F(s) - csX(s) - kX(s)$$

$$X(s)(ms^2 + cs + k) = F(s)$$

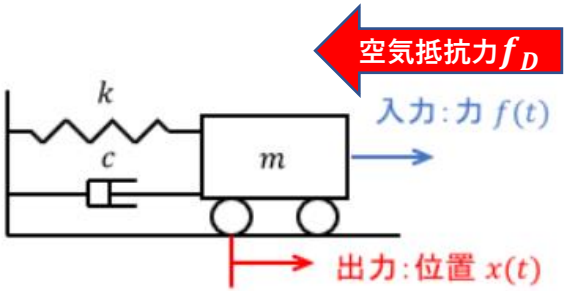
伝達関数：$\boldsymbol{G(s) = \frac{X(s)}{F(s)} = \frac{1}{ms^2 + cs + k}}$



k：ばね定数
c： ダンパの減衰係数
m：車の質量
a：車の加速度
$f(t)$：車に働く力
$x(t)$：車の位置

（17） 課題　1

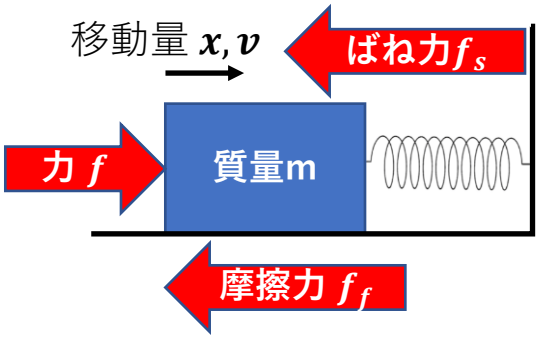
　空気抵抗の有るばねダンパ系の伝達関数を求めよ。尚、空気抵抗力はKvとする。



（18） 課題 ２

質量m[kg]の物体に力f[N]を加えた時の移動量x[m]の伝達関数を求めよ。尚、ばねの定数をK、物体の摩擦係数をD[N s/m]で、下式とする。

ばね力 $\boldsymbol{f_s = Kx}$　摩擦力 $\boldsymbol{f_f = Dv}$

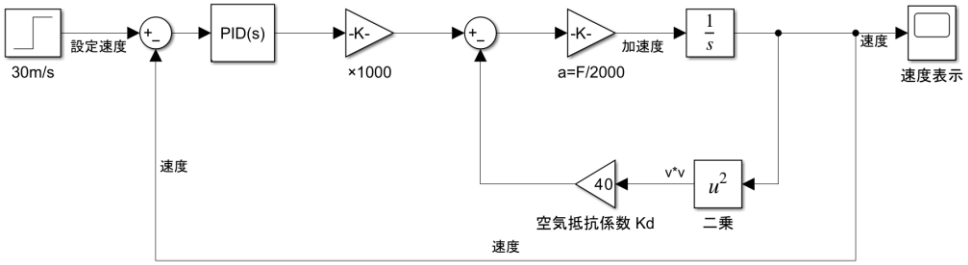


SIMLINK ３．ブロック線図

（１） 概要

SIMULINKはMBD(Model Based Design)で制御システムの開発を行う環境を提供している。

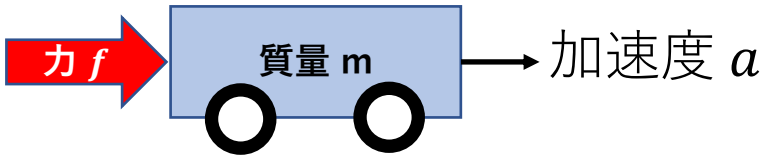
具体的には下図のようなブロック線図により制御システムの設計と数値シミュレーションが可能となっている。尚、各ブロックは微分方程式をラプラス変換した要素で構成される。



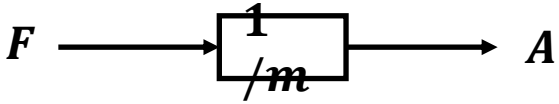
ブロック線図例

（2） 空気抵抗の無い車を力fで加速した時の加速度変化のブロック線図

$$f(t) = ma(t)$$



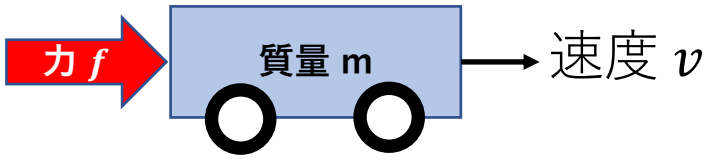
伝達関数：$\boldsymbol{G_{fa}(s) = \frac{A(s)}{F(s)} = \frac{1}{m}}$



（3） 空気抵抗の無い車を力fで加速した時の速度変化のブロック線図

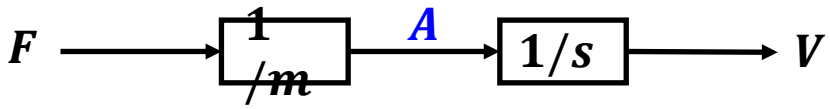
力と加速度の伝達関数

伝達関数：$\boldsymbol{G_{fa}(s) = \frac{A(s)}{F(s)} = \frac{1}{m}}$

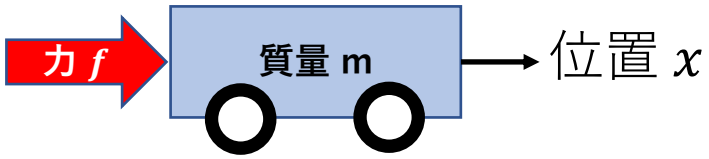


力と速度の伝達関数

伝達関数：$\boldsymbol{G_{fv}(s) = \frac{V(s)}{F(s)} = \frac{1}{ms} = \frac{1}{m} \times \frac{1}{s}}$ ： $\boldsymbol{V(s) = A(s) \times \frac{1}{s}}$

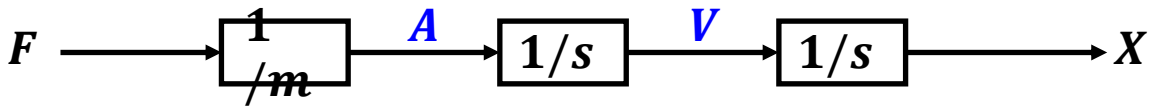


（4） 空気抵抗の無い車を力fで加速した時の位置変化のブロック線図



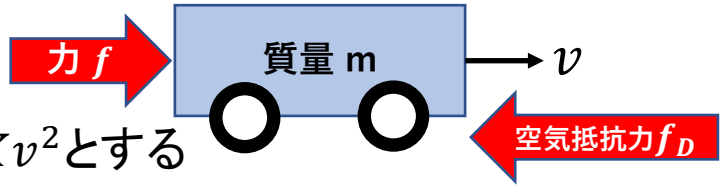
力と位置の伝達関数

伝達関数：$\boldsymbol{G(s)} = \dfrac{\boldsymbol{X(s)}}{\boldsymbol{F(s)}} = \dfrac{\boldsymbol{1}}{\boldsymbol{ms^2}} = \dfrac{\boldsymbol{1}}{\boldsymbol{m}} \times \dfrac{\boldsymbol{1}}{\boldsymbol{s}} \times \dfrac{\boldsymbol{1}}{\boldsymbol{s}}$



（5） 空気抵抗の有る車を力fで加速した時の速度変化のブロック線図―1

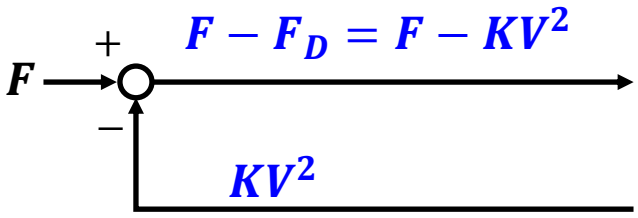
1．まず自動車に働く力の合計を考える



力合計 $f_t = f - f_D = f - Kv^2$　　空気抵抗力をKv^2とする

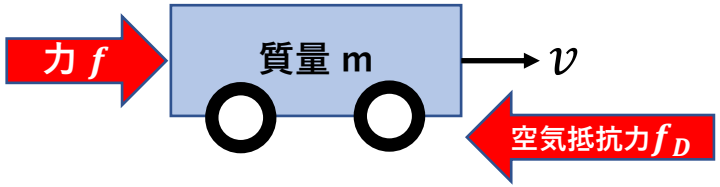
⇩ ラプラス変換

力合計 $\boldsymbol{F_t(s) = F(s) - F_D(s) = F(s) - KV^2(s)}$



（5） 空気抵抗の有る車を力fで加速した時の速度変化のブロック線図ー2

2．力から加速度を求める

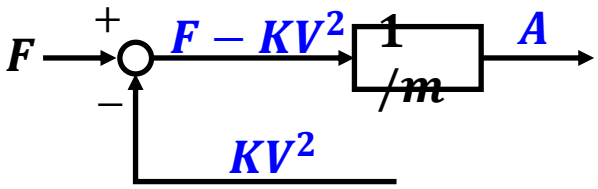


$$f_t = ma$$

ラプラス変換

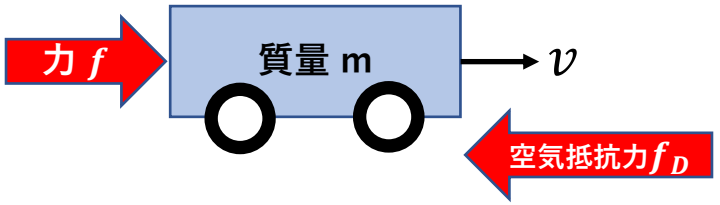
$$F_t(s) = F(s) - kV(s) = mA(s)$$

$$\frac{\boldsymbol{F_t(s)}}{\boldsymbol{m}} = \frac{\boldsymbol{F(s) - KV(s)}}{\boldsymbol{m}} = (\boldsymbol{F(s) - KV^2(s)}) \times \frac{\boldsymbol{1}}{\boldsymbol{m}} = \boldsymbol{A(s)}$$



（5） 空気抵抗の有る車を力fで加速した時の速度変化のブロック線図ー3

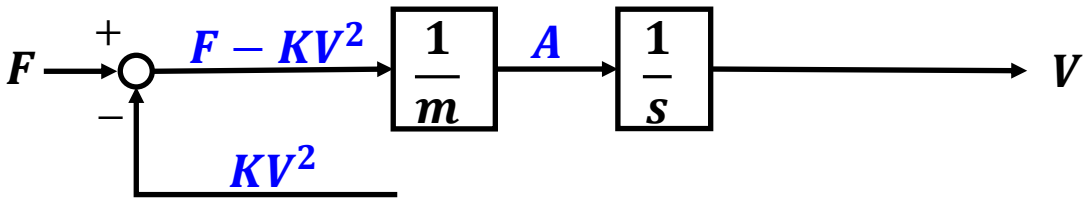
3．加速度から速度を求める



$$a(t) = \frac{dv(t)}{dt}$$

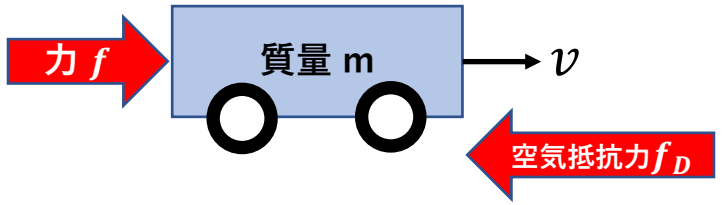
⇩ ラプラス変換

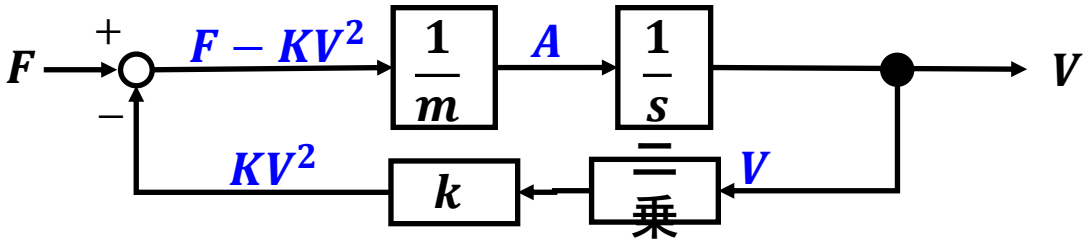
$$A(s) \times \frac{1}{s} = V(s)$$



（5） 空気抵抗の有る車を力fで加速した時の速度変化のブロック線図ー4

4. 空気抵抗力を力から引き算する。

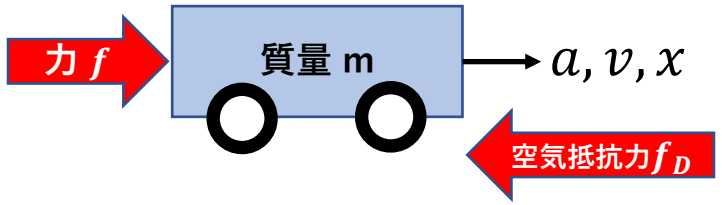




以上でブロック線図が完成

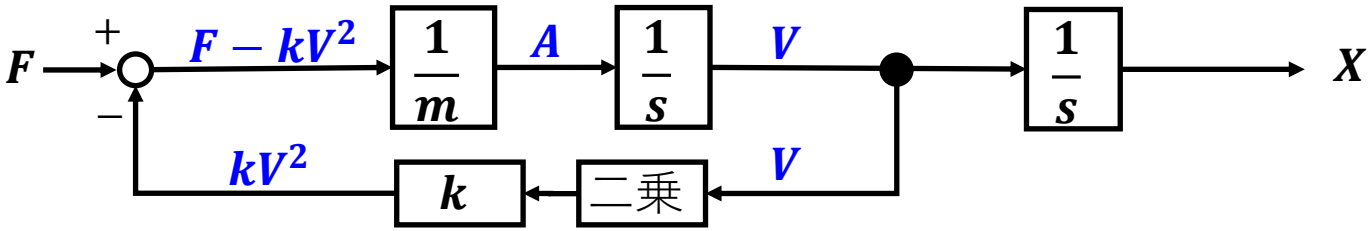
（6） 空気抵抗の有る車を力fで加速した時の位置変化のブロック線図

$$ma(t) = f(t) - f_D(t) = f(t) - Kv^2 \quad K = \frac{1}{2}\rho C_d S$$



$$a(t) = \frac{1}{m}(f(t) - Kv^2(t))$$

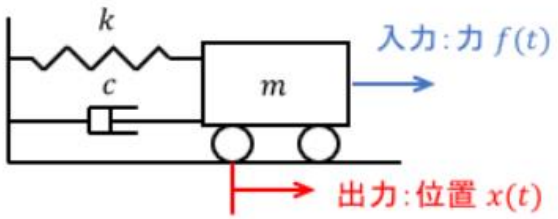
$$\left(\boldsymbol{F(s) - KV^2(s)}\right) \times \frac{\boldsymbol{1}}{\boldsymbol{m}} = \boldsymbol{A(s)} \quad \boldsymbol{A(s)} \times \frac{\boldsymbol{1}}{\boldsymbol{s}} = \boldsymbol{V(s)} \quad \boldsymbol{V(s)} \times \frac{\boldsymbol{1}}{\boldsymbol{s}} = \boldsymbol{X(s)}$$



（7） ばねダンパ系を力fで加速した時の位置変化のブロック線図

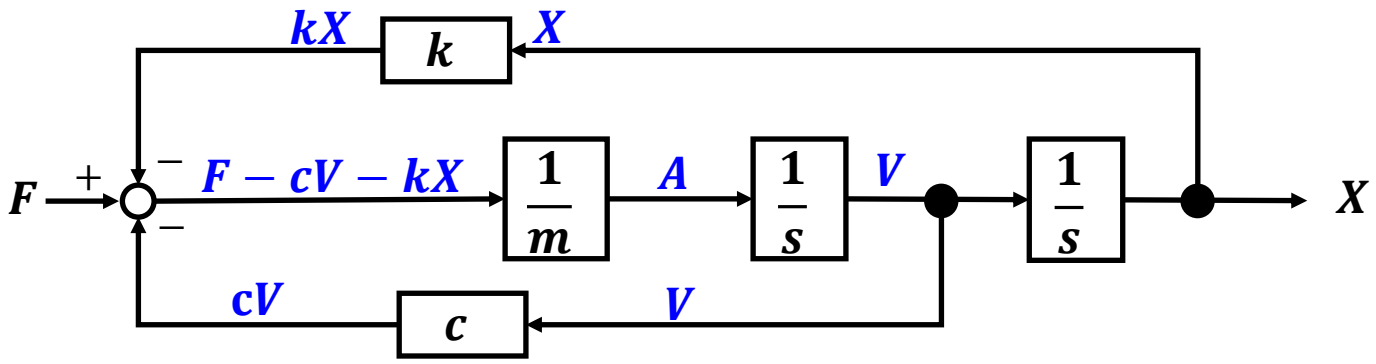
$$f(t) - cv(t) - kx(t) = ma(t)$$

$$\left(f(t) - cv(t) - kx(t)\right) \times \frac{1}{m} = a(t)$$



k：ばね定数
c： ダンパの減衰係数
m：車の質量
a：車の加速度
$f(t)$：車に働く力
$x(t)$：車の位置

$$\left(\boldsymbol{F(s) - cV(s) - kX(s)}\right) \times \frac{\boldsymbol{1}}{\boldsymbol{m}} = \boldsymbol{A(t)} \qquad \boldsymbol{A(s)} \times \frac{\boldsymbol{1}}{\boldsymbol{s}} = \boldsymbol{V(s)} \qquad \boldsymbol{V(s)} \times \frac{\boldsymbol{1}}{\boldsymbol{s}} = \boldsymbol{X(s)}$$



SIMLINK ４．Simlinkの起動

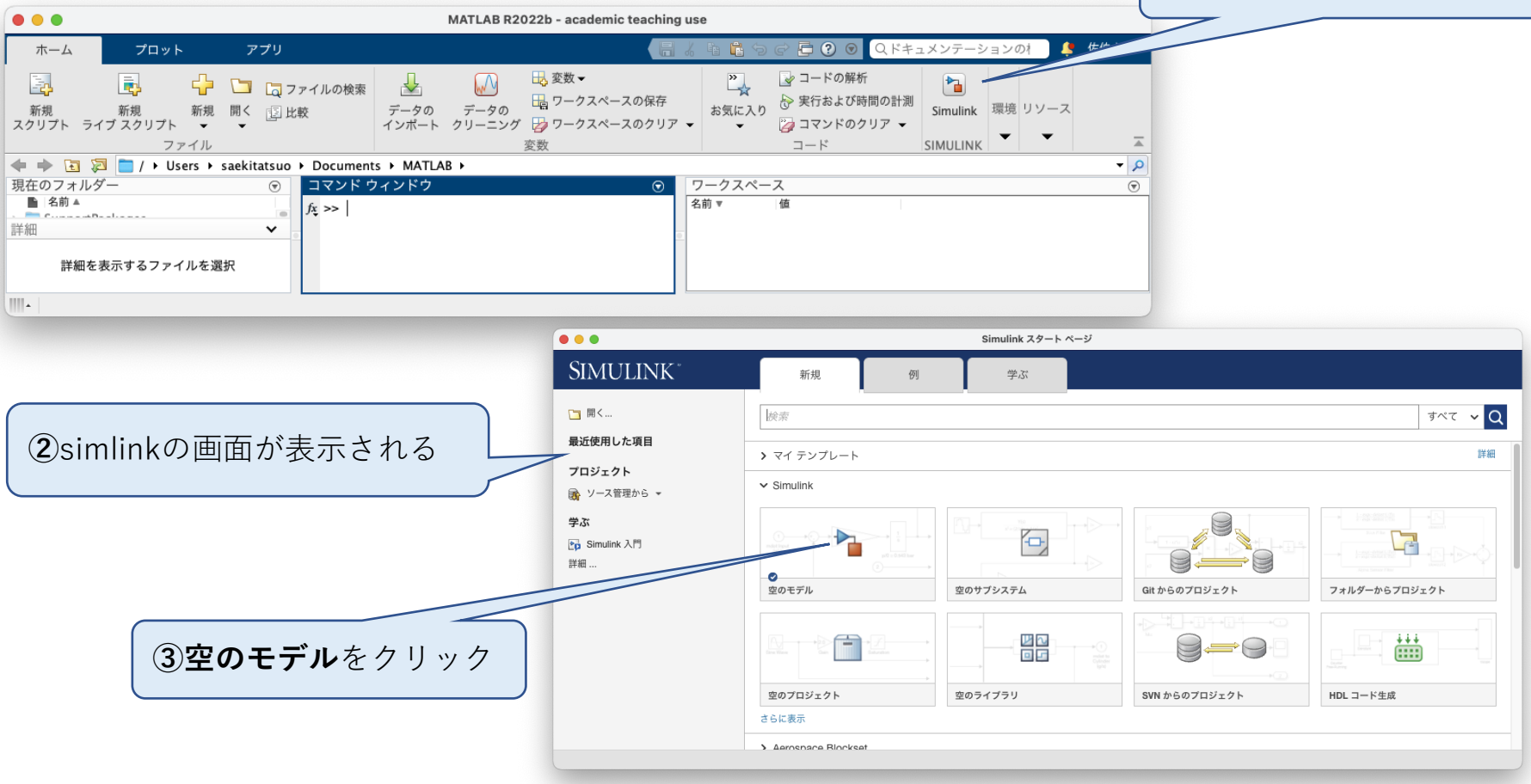
（１） 概要

MATLABは非常に強力な数値演算能力を活かして微分方程式を計算することも得意であり、この能力を応用して制御システムの設計・シミュレーションに広く使用されている。

simlinkでは、PCのGUIを利用してグラフィカルに制御システムの開発が可能で、このような開発環境をMBD(Model Based Design)と言う。

応用分野は車から飛行機、電機等ほぼ全ての産業機器の開発で使用されており、特に自動車産業ではJMAAB(Japan MBD Automotive Advisory Board)として標準化されて、現在開発・設計・試験等で広く利用されている。

（2） simlinkの立ち上げ

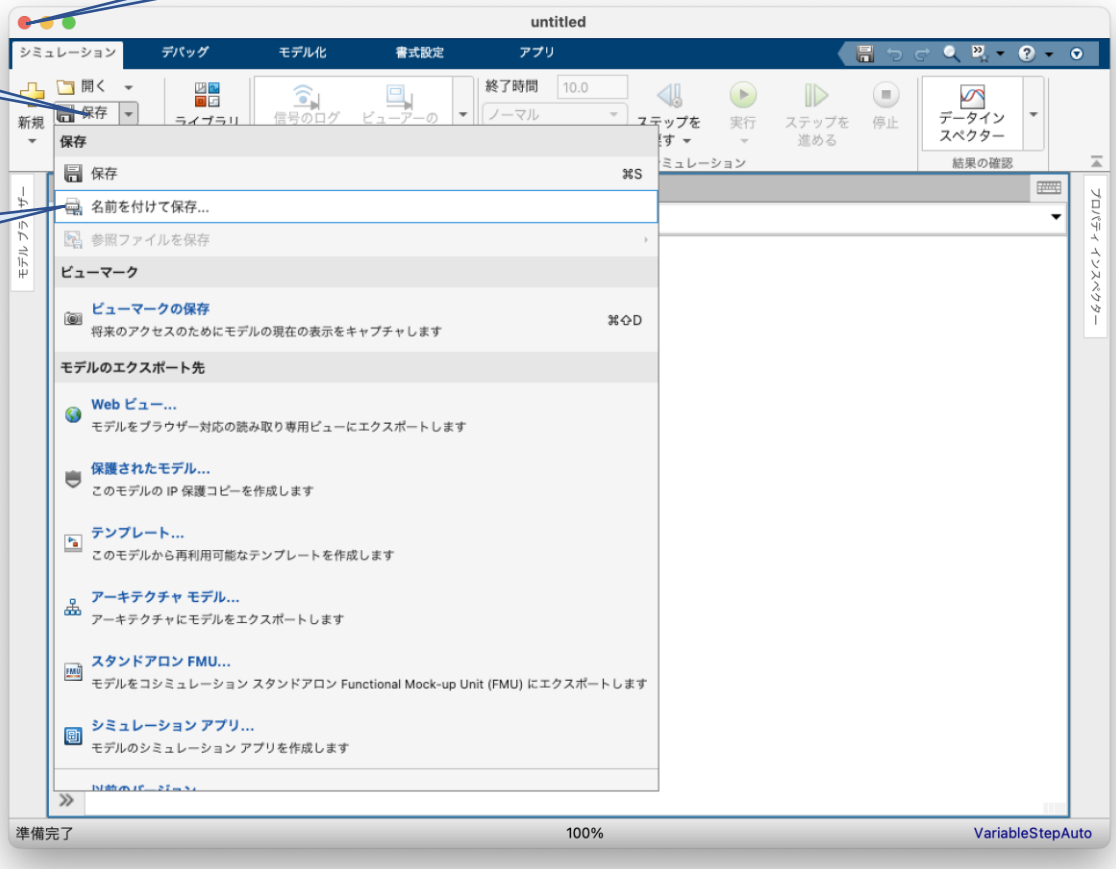


（3） ファイルの保存とsimlinkの終了

③●をクリックして終了

①**保存**をクリック

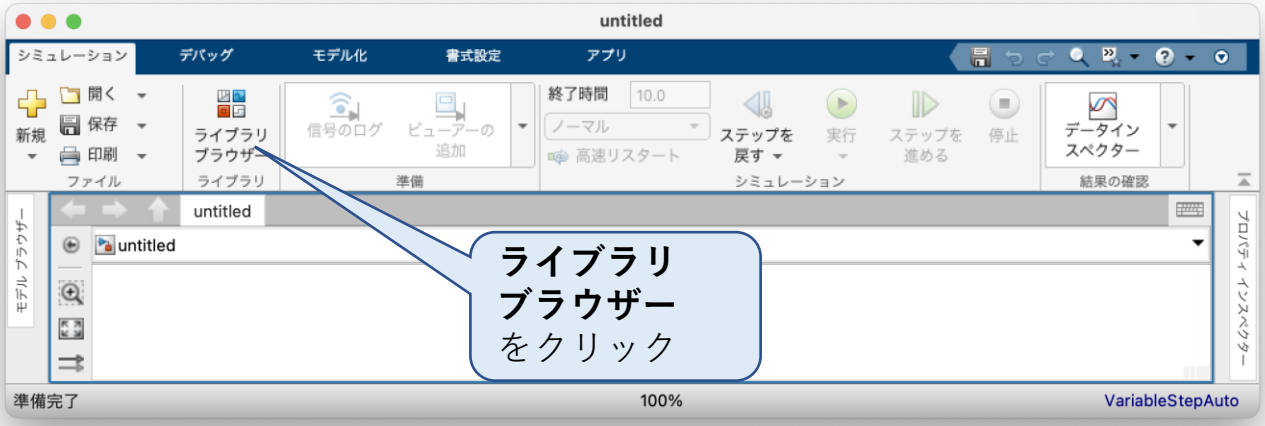
②**名前を付けて保存**をクリック



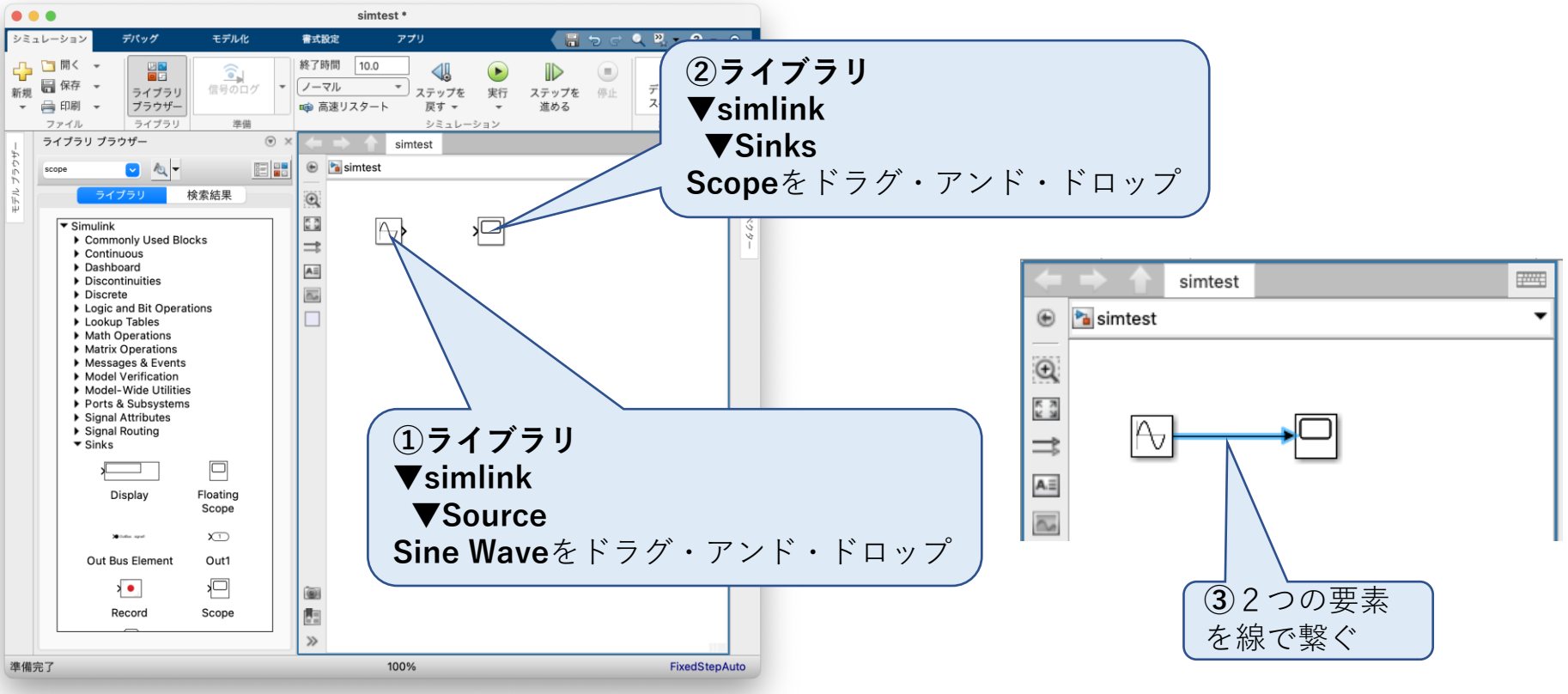
SIMLINK ５．波形の表示

（1） 概要

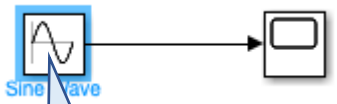
simlinkの信号発生機能(Sources)を使用してsin波形(Sine Wave)を発生し、信号などをモニターする機能(Sinks)を用いて波形を観測(Scope)する。
各要素は、ライブラリブラウザーから選択することができる。



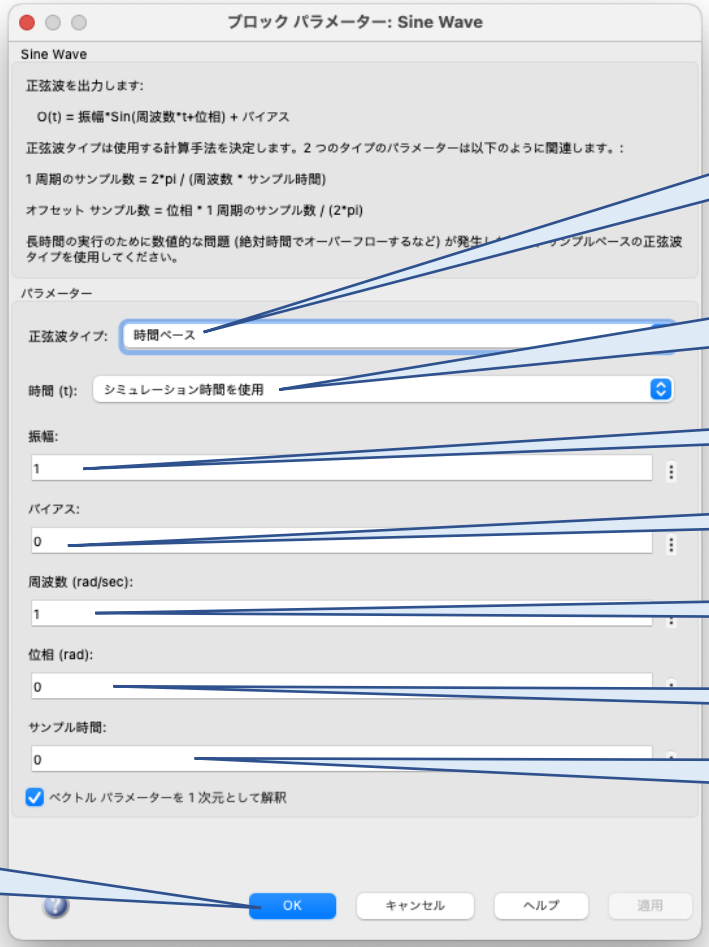
（2） モデルの作成
sin波形を発生する信号発生器をスコープで直接観測する。



（3） 発生信号の設定



①Sine Wave
をクリック



②正弦波タイプ：は、時間ベースを選択

③時間：は、シミュレーション時間を選択

④振幅：は、 1 を設定

⑤バイアス：は、 0 を設定

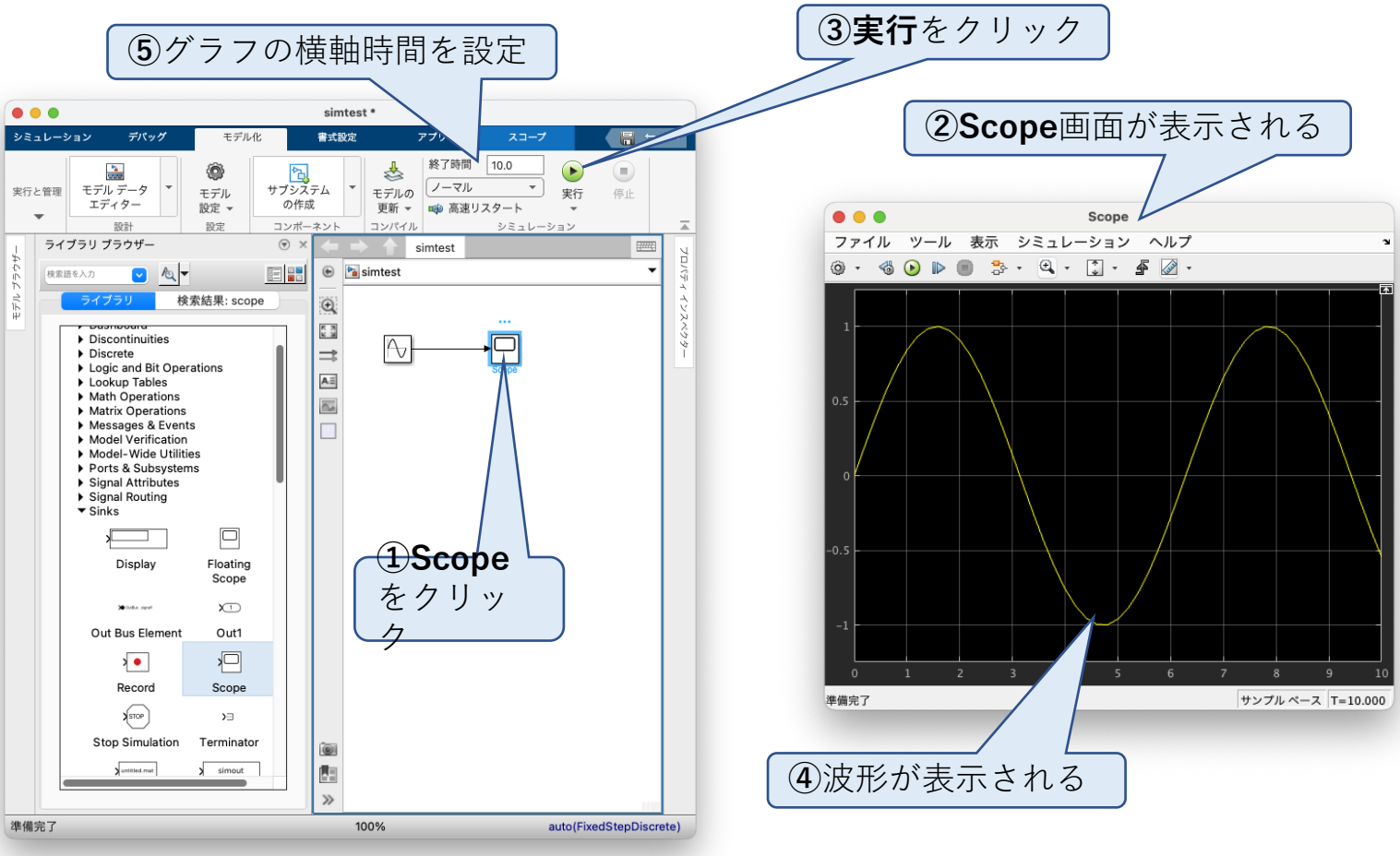
⑥周波数：は、 1 を設定

⑦位相：は、 0 を設定

⑥サンプル時間：は、 0 を設定

③設定終了後、
OKをクリック

（4） シミュレーションの実行



（5） 課題

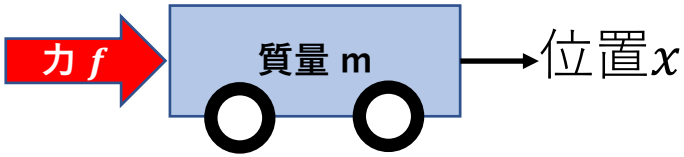
1．実効値100[Vrms]、周波数60[Hz]の波形を表示させよ。

2．初期値0、最終値1、ステップ時間0.1のステップ出力を表示させよ。

SIMLINK 6．抵抗の無い自動車の加速

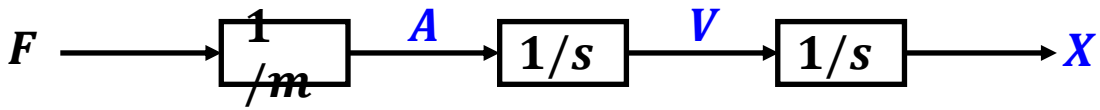
（1） 概要

質量M=2[ton]の自動車を、力F=1[kN]で押し続けた時の自動車の位置の変化をシミュレートする。

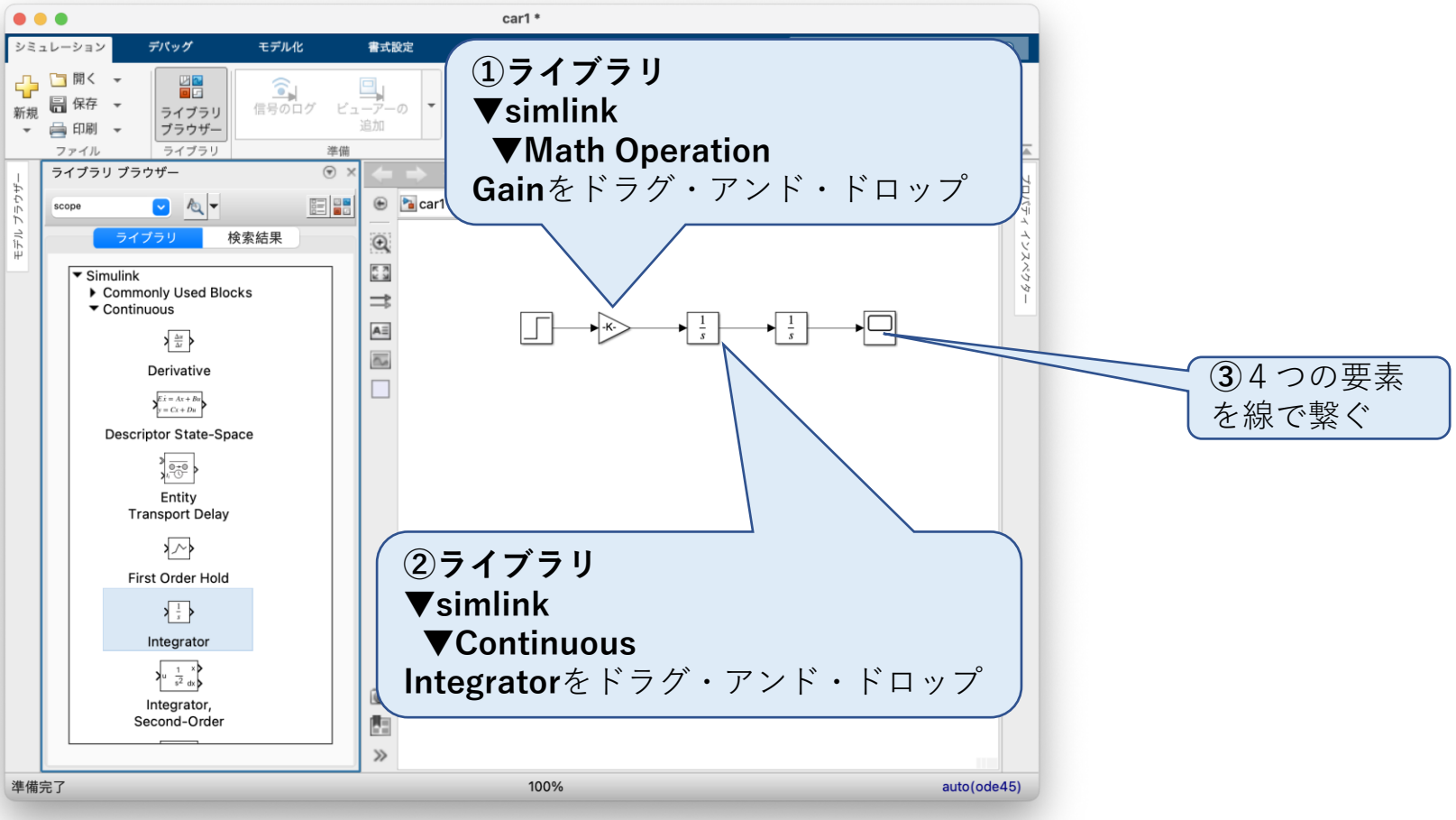


力と位置の伝達関数

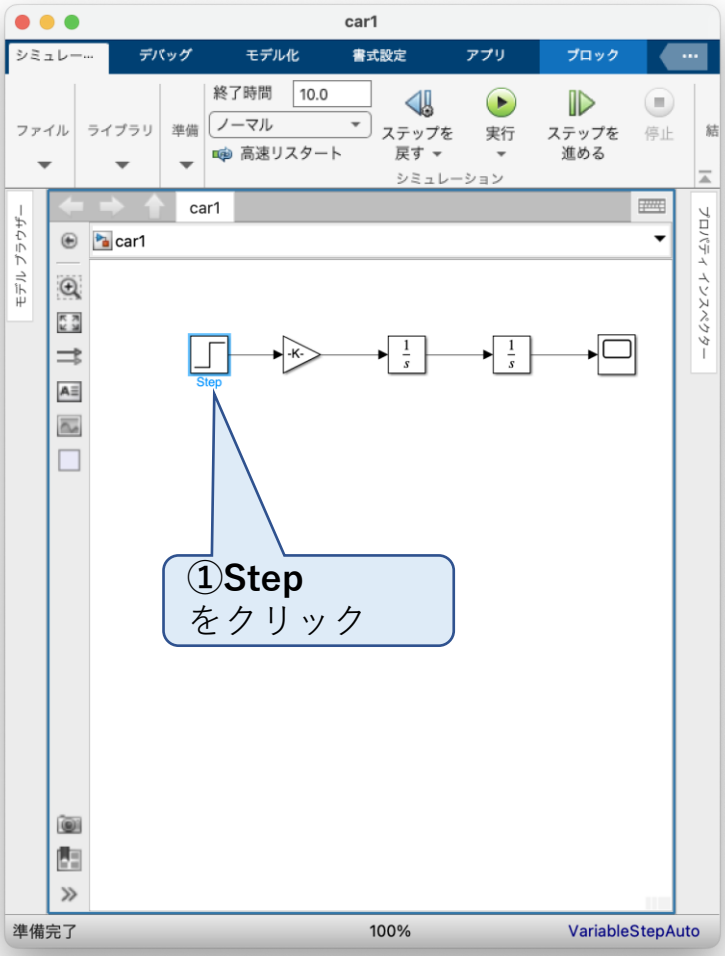
$$伝達関数：G(s)=\frac{X(s)}{F(s)}=\frac{1}{ms}=\frac{1}{m}\times\frac{1}{s}\times\frac{1}{s}$$

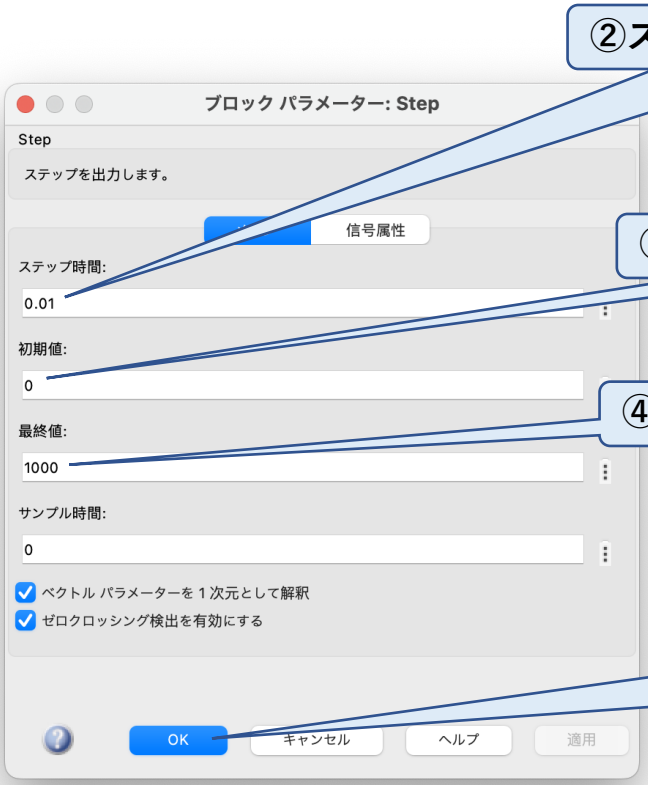


（2） モデルの作成


car1 *
シミュレーション
デバッグ
モデル化
書式設定
新規
開く
保存
印刷
ファイル
ライブラリブラウザー
ライブラリ
信号のログ
ビューアーの追加
準備
ライブラリ ブラウザー
scope
ライブラリ
検索結果
Simulink
Commonly Used Blocks
Continuous
Derivative
Descriptor State-Space
Entity
Transport Delay
First Order Hold
Integrator
Integrator,
Second-Order
car1
-K-
準備完了
100%
auto(ode45)
①ライブラリ
▼simlink
▼Math Operation
Gainをドラグ・アンド・ドロップ
②ライブラリ
▼simlink
▼Continuous
Integratorをドラグ・アンド・ドロップ
③４つの要素
を線で繋ぐ

（3） 入力値（力[N]）の設定





②**ステップ時間：**0を設定

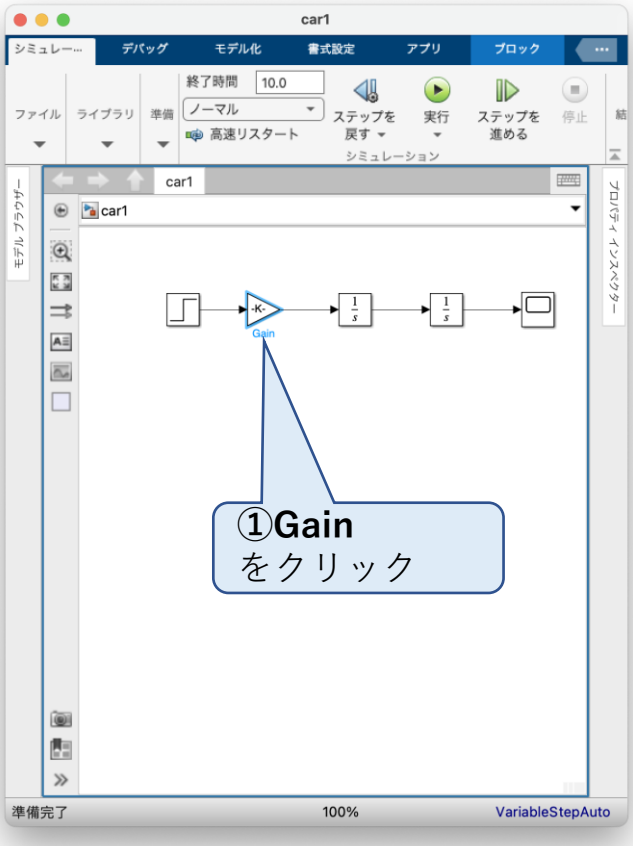
③**初期値：**0[N]を設定

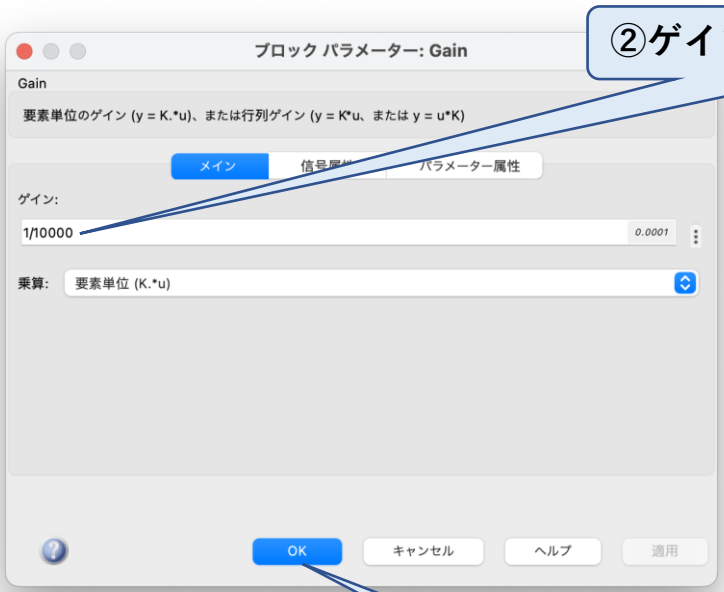
④**最終値：** 1 000[N]を設定

⑤設定終了後、
OKをクリック

（4） ゲインの設定

$$a = \frac{F}{M} = F[N]$$

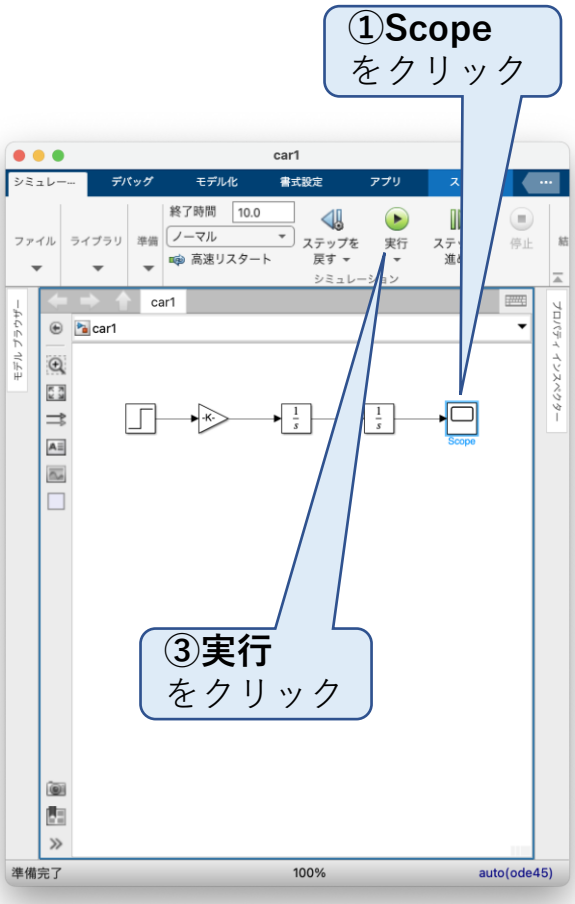


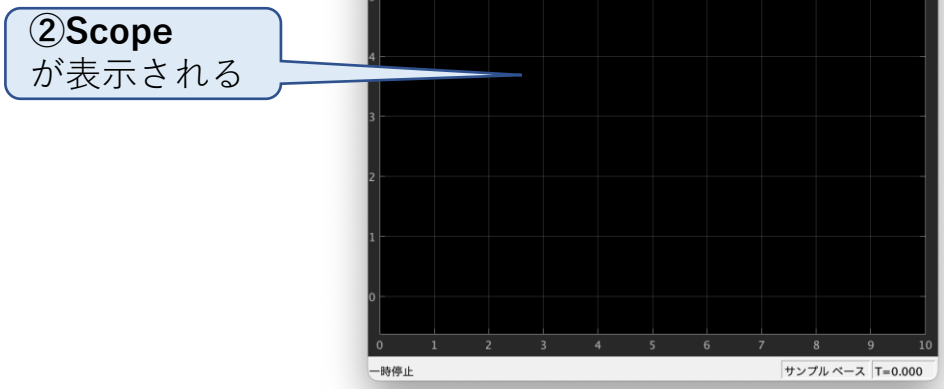


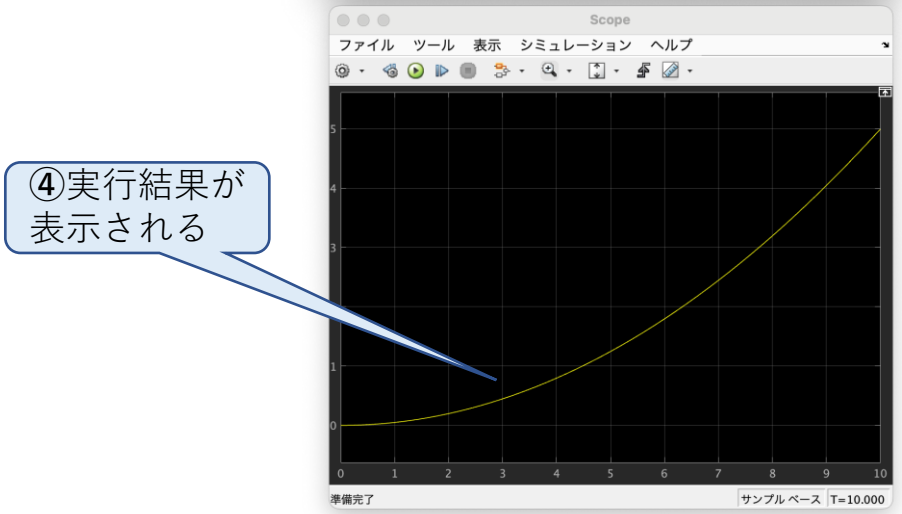
②**ゲイン**：1/2000を設定

③設定終了後、
OKをクリック

（5） 実行


①Scope
をクリック
car1
シミュレーション
デバッグ
モデル化
書式設定
アプリ
終了時間
10.0
ノーマル
高速リスタート
ステップを
戻す
実行
停止
car1
Scope
③実行
をクリック
準備完了
100%
auto(ode45)


②Scope
が表示される
Scope
ファイル
ツール
表示
シミュレーション
ヘルプ
一時停止
サンプル ベース
T=0.000


④実行結果が
表示される
Scope
ファイル
ツール
表示
シミュレーション
ヘルプ
準備完了
サンプル ベース
T=10.000

（6） 課題

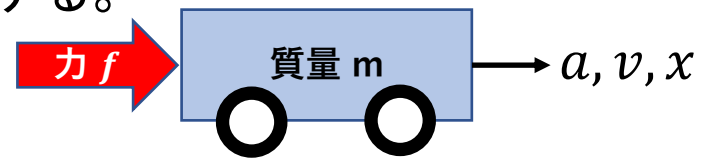
車の加速度、速度、位置の変化を同時に3つ表示させよ。

SIMLINK 7．抵抗の有る自動車の加速

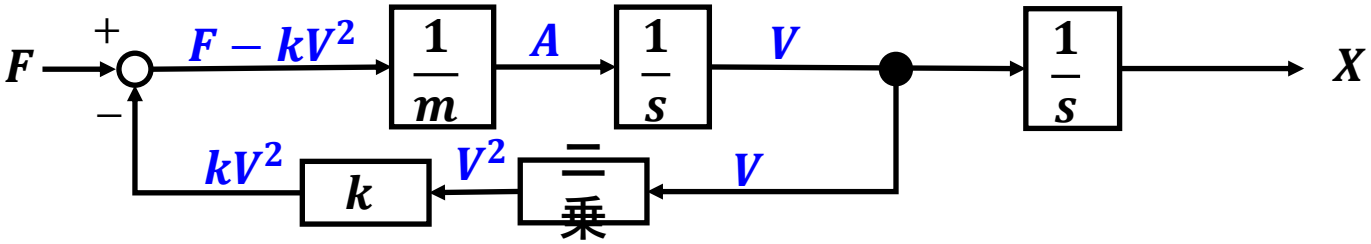
（1） 概要

質量M=2[ton]の自動車を、力F=1[kN]で押し続けた時の自動車の動きをシミュレートする。

尚、空気抵抗力$f = kv^2[N]$、$k = 40$とする。



ブロック線図を以下に示す。



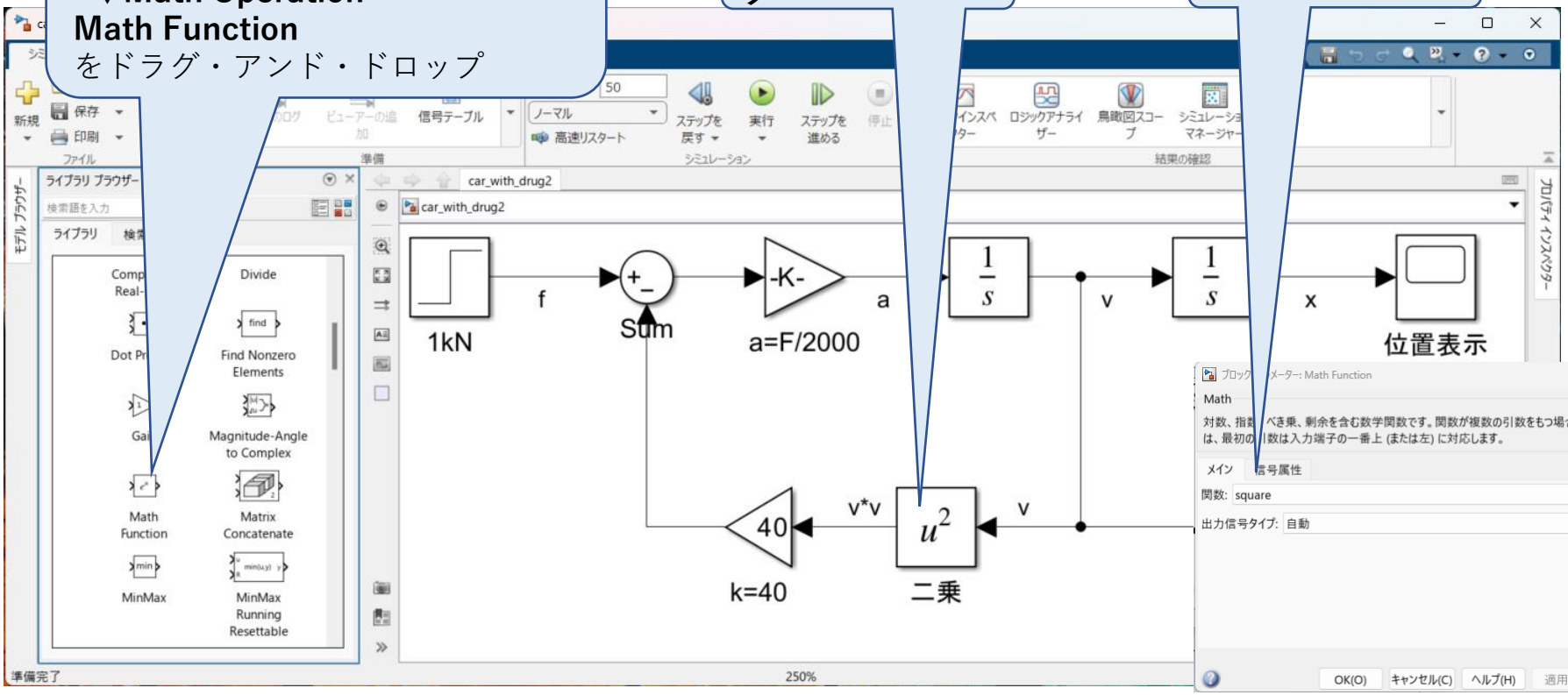
（2） モデルの作成

簡単に二乗

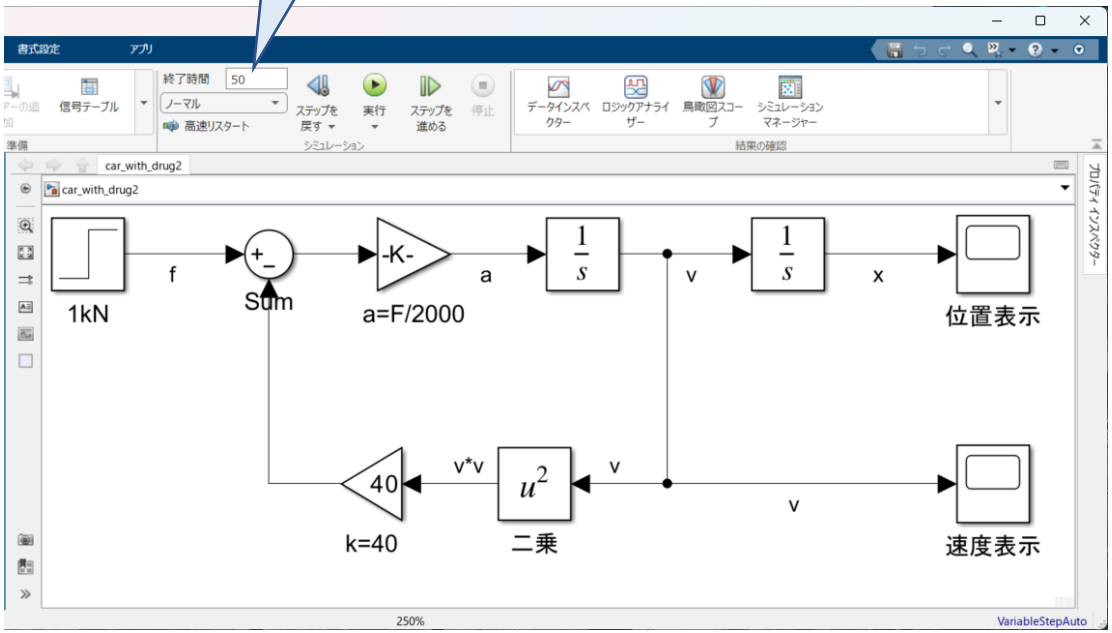
①ライブラリ
▼simlink
▼Math Operation
Math Function
をドラグ・アンド・ドロップ

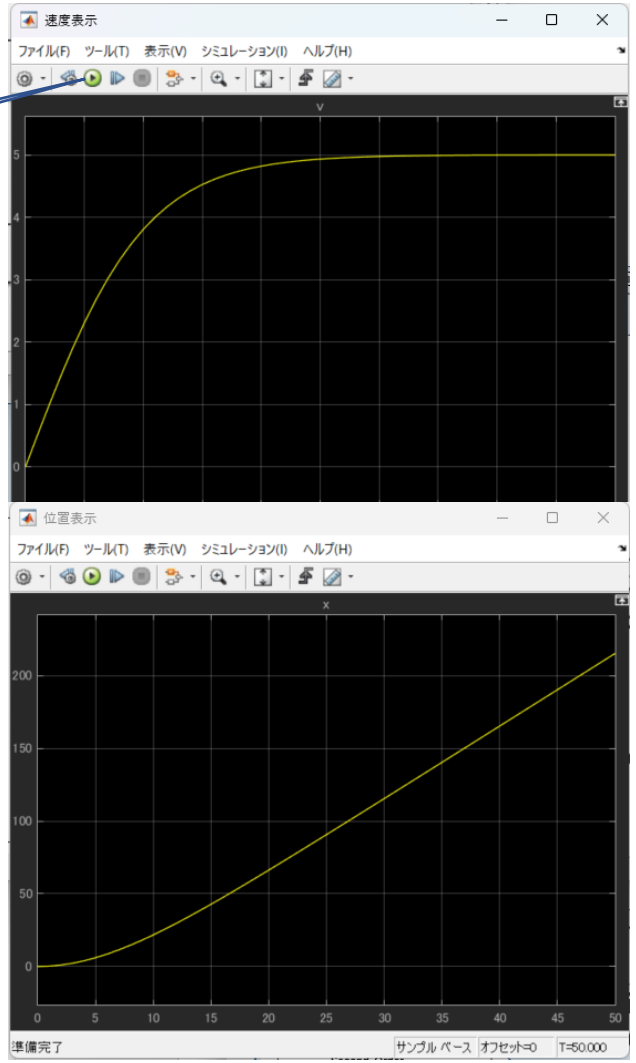
②ダブルクリック

③squareを選択



（3） 実行

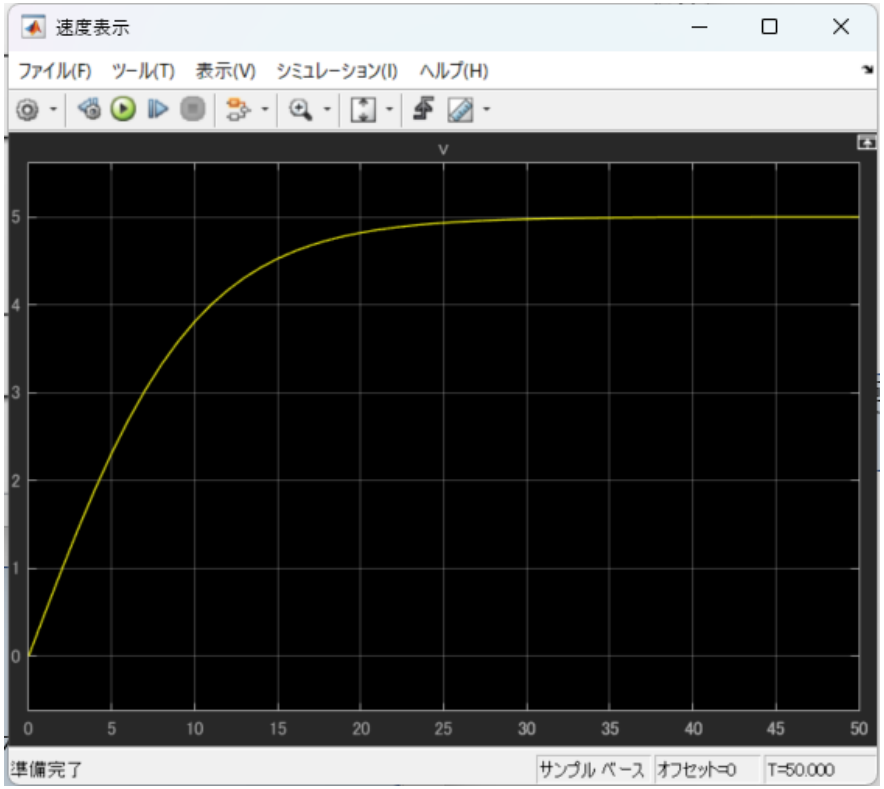

①終了時間を設定
②実行
書式設定
アプリ
終了時間
50
ノーマル
高速リスタート
信号テーブル
ステップを戻す
実行
ステップを進める
停止
データインスペクター
ロジックアナライザー
鳥瞰図スコープ
シミュレーションマネージャー
シミュレーション
結果の確認
car_with_drug2
1kN
f
Sum
-K-
a=F/2000
a
1/s
v
1/s
x
位置表示
40
k=40
v*v
u^2
二乗
v
v
速度表示
250%
VariableStepAuto


速度表示
ファイル(F) ツール(T) 表示(V) シミュレーション(I) ヘルプ(H)
v
0
1
2
3
4
5
位置表示
ファイル(F) ツール(T) 表示(V) シミュレーション(I) ヘルプ(H)
x
0
50
100
150
200
0
5
10
15
20
25
30
35
40
45
50
準備完了
サンプル ベース
オフセット=0
T=50.000

（4） 実行結果の考察

速度の変化を見ると、始めは大きく加速するが徐々に減少し、最終的には5[m/s]で一定となっている。

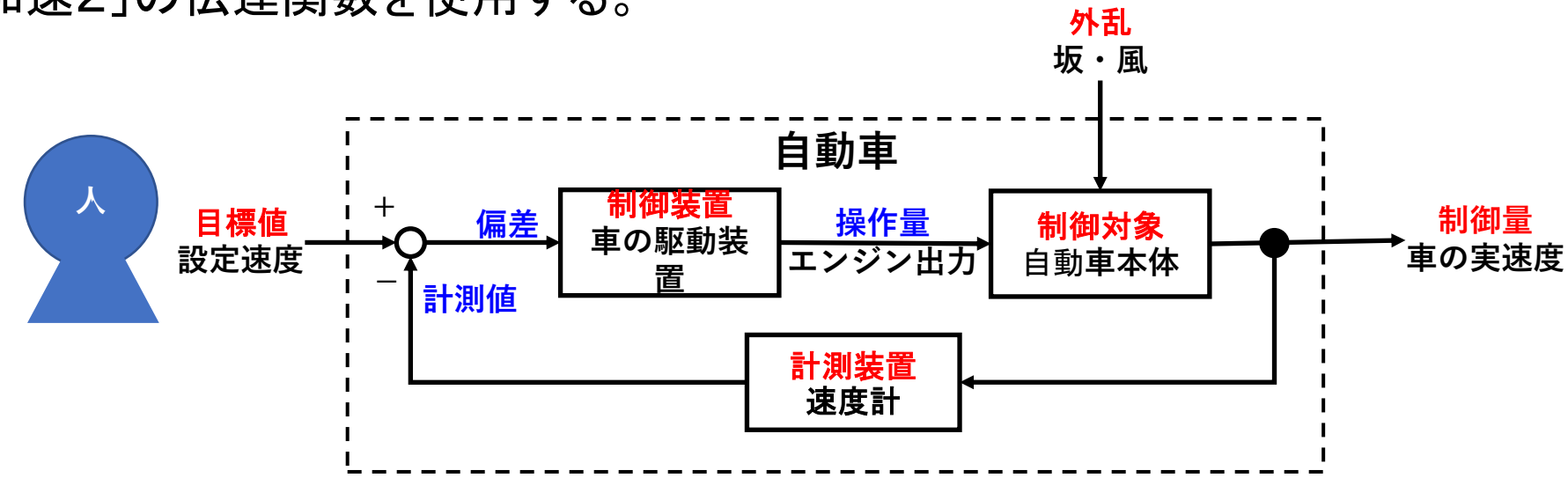
車の推進力が1[kN]であり、速度が5[m/s]の時の空気抵抗力は
$f = kv^2 = 40 \times 5^2 = 1000[N]$となり、
これ以上加速できないことを示している。



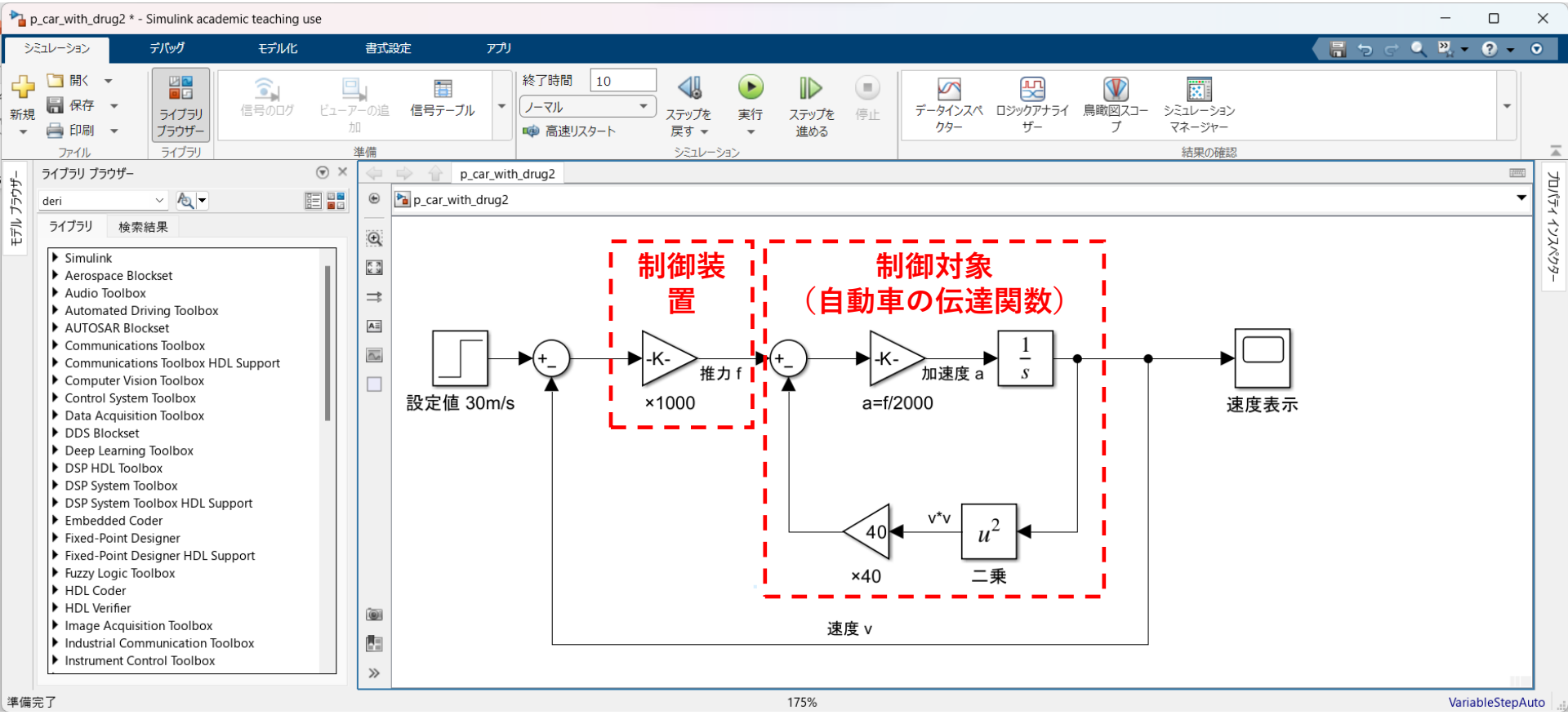
SIMLINK ８．比例制御（P制御）

（1） 概要

　空気抵抗のある車の速度を、比例制御を使ったフィードバックで制御する。ここで設定値を30[m/s]とし、制御対象は「SIMLINK ８．抵抗のある自動車の加速２」の伝達関数を使用する。



（2） モデルの作成


p_car_with_drug2 * - Simulink academic teaching use
シミュレーション
デバッグ
モデル化
書式設定
アプリ
新規
開く
保存
印刷
ファイル
ライブラリブラウザー
ライブラリ
信号のログ
ビューアーの追加
信号テーブル
準備
終了時間
10
ノーマル
高速リスタート
ステップを戻す
実行
ステップを進める
停止
シミュレーション
データインスペクター
ロジックアナライザー
鳥瞰図スコープ
シミュレーションマネージャー
結果の確認
ライブラリ ブラウザー
deri
ライブラリ
検索結果
Simulink
Aerospace Blockset
Audio Toolbox
Automated Driving Toolbox
AUTOSAR Blockset
Communications Toolbox
Communications Toolbox HDL Support
Computer Vision Toolbox
Control System Toolbox
Data Acquisition Toolbox
DDS Blockset
Deep Learning Toolbox
DSP HDL Toolbox
DSP System Toolbox
DSP System Toolbox HDL Support
Embedded Coder
Fixed-Point Designer
Fixed-Point Designer HDL Support
Fuzzy Logic Toolbox
HDL Coder
HDL Verifier
Image Acquisition Toolbox
Industrial Communication Toolbox
Instrument Control Toolbox
p_car_with_drug2
制御装置
制御対象
（自動車の伝達関数）
設定値 30m/s
-K-
推力 f
×1000
-K-
加速度 a
a=f/2000
速度表示
v*v
40
×40
二乗
速度 v
準備完了
175%
VariableStepAuto

（3） 実行結果と考察―1

　設定速度が30[m/s]であるものの、最終的には約17.6[m/s]となっている。

　この時の空気抵抗力は

$f_d = kv^2 = 40 \times 17.6^2$

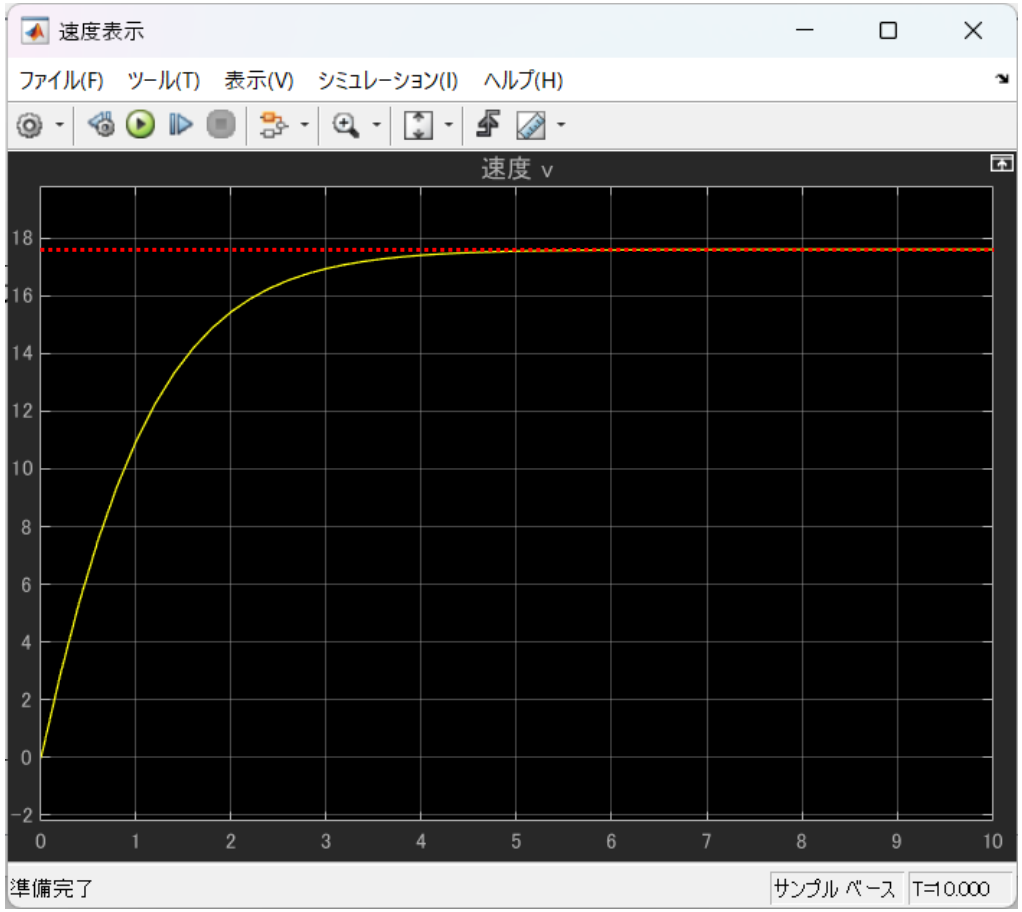
$= 12,390[N]$となる。

　比例制御では推力は偏差に比例するため、この速度での推力は

$f = (30 - 17.6) \times 1000$

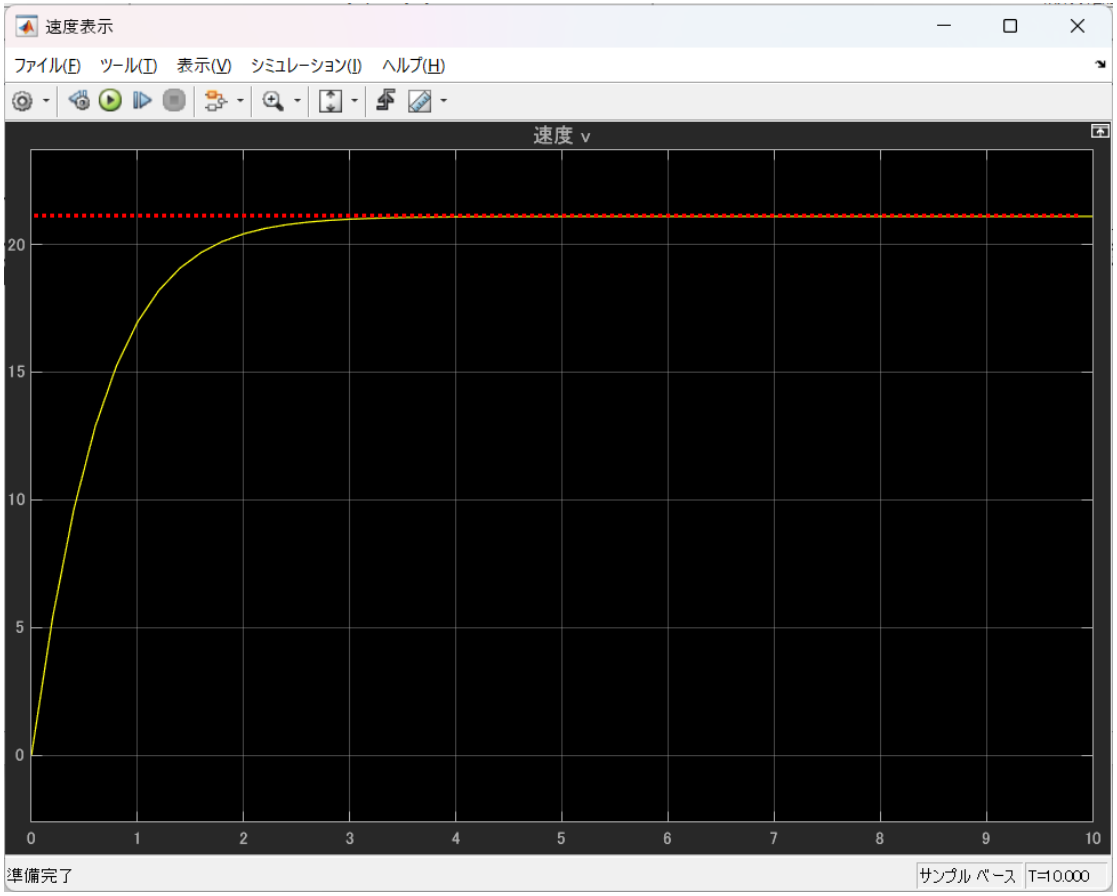
$= 12,400[N] \cong 12,390$

であり、推力と抵抗力がつりあっていてこれ以上の速度が上昇しないことを示している。



（3） 実行結果と考察ー2

制御装置の係数を1000から2000に変更すると、最終速度は21.1[m/s]まで上昇するが、設定速度である30[m/s]までは到達しない点は同様となる。



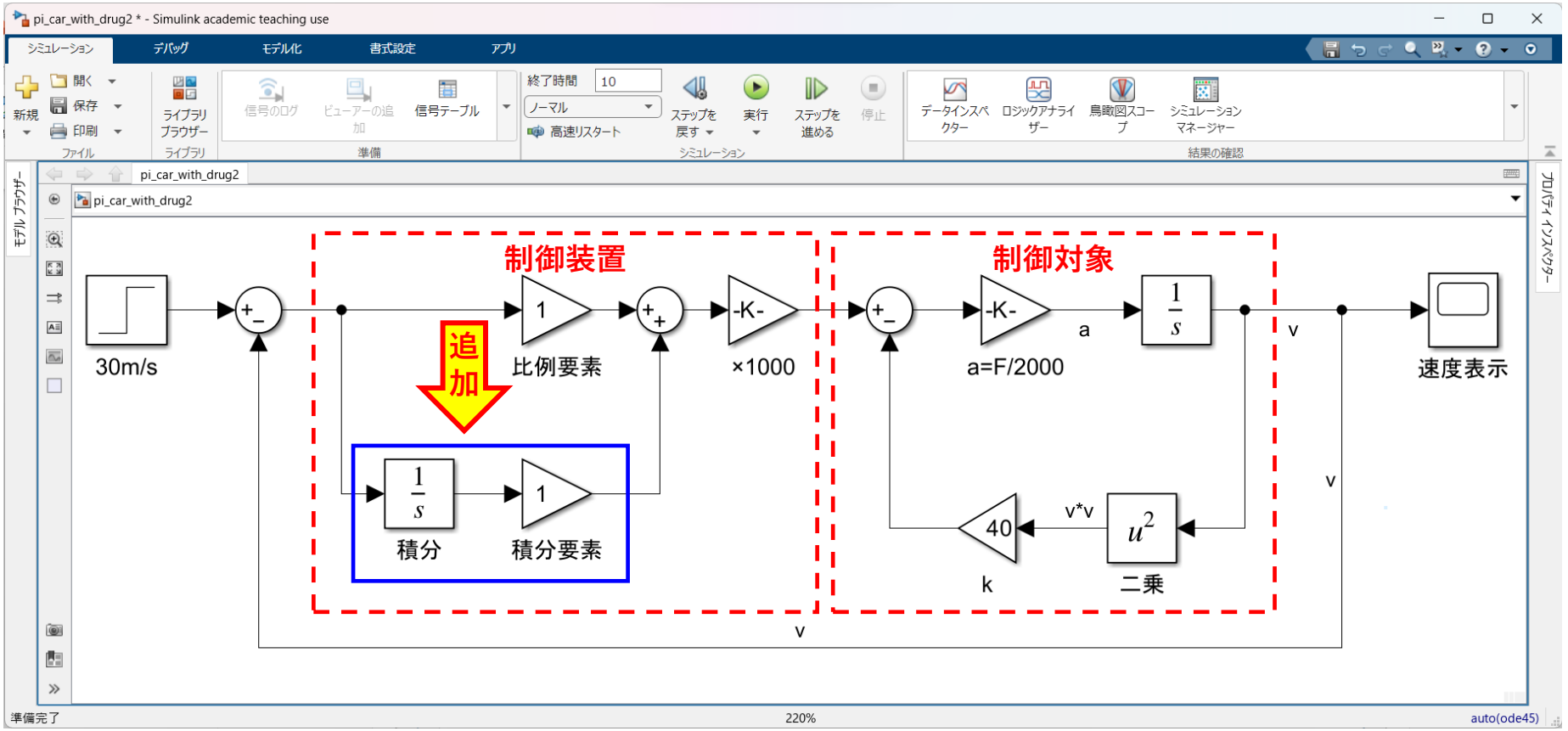
SIMLINK ９．比例積分制御（PI制御）

（1） 概要

「SIMLINK ９．比例制御（P制御）」の制御システムでは、設定速度30[km/s]に自動車の速度は到達しない。この問題を解決するために比例要素に平行に積分要素が追加される。

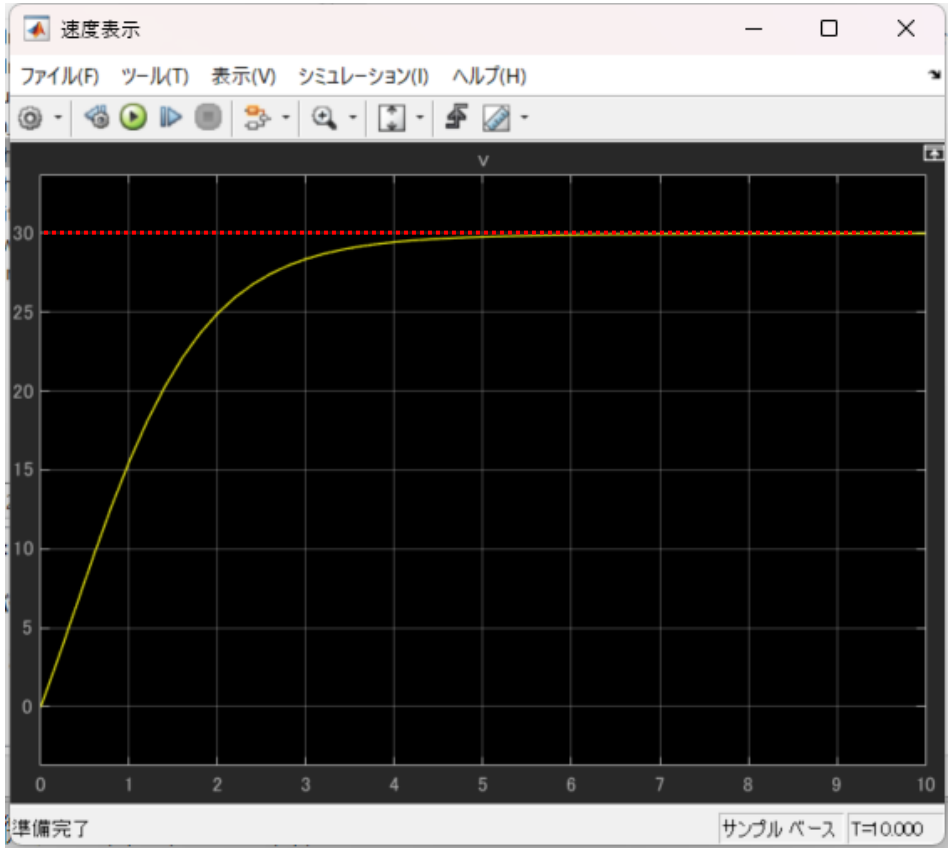
偏差がある状態で偏差を時間的に積分すると、その値は時間とともに増大するため、やがて操作量が多きくなり設定速度に到達できる。

（2） モデルの作成


pi_car_with_drug2 * - Simulink academic teaching use
シミュレーション
デバッグ
モデル化
書式設定
アプリ
終了時間
10
ノーマル
高速リスタート
ステップを戻す
実行
ステップを進める
停止
データインスペクター
ロジックアナライザー
鳥瞰図スコープ
シミュレーションマネージャー
pi_car_with_drug2
制御装置
制御対象
30m/s
追加
比例要素
×1000
積分
積分要素
a=F/2000
a
v
速度表示
40
k
v*v
二乗
準備完了
220%
auto(ode45)

（3） 実行結果と考察ー1

　制御要素に積分要素を追加して、比例・積分各要素に
P=1.0
I=1.0
を設定した場合、
自動車の速度が設定値と同じ30[m/s]に達していることがわかる。

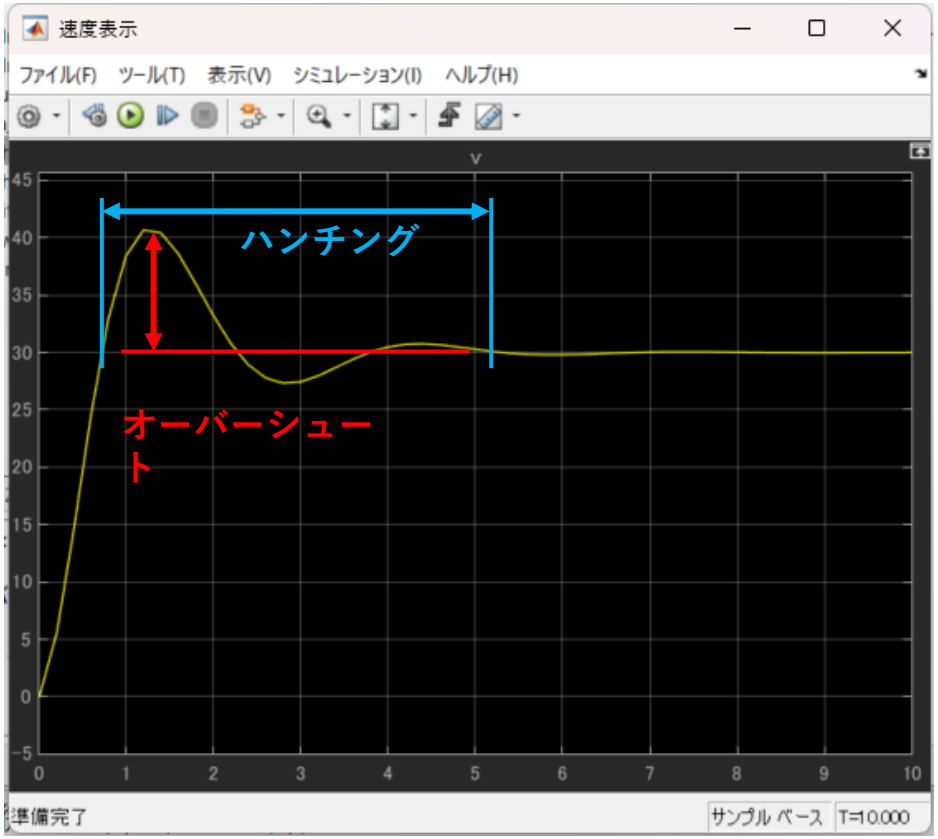


（4） 実行結果と考察―2

　前頁と同じ制御システムで、 比例・積分各要素にP=1.0, I=10.0 を設定すると、速度の立ち上がりが早く、最終的には30[m/s]になるものの、始めに40[m/s]まで一時的に上がり、その後下がりすぎている。

　この上がり過ぎ量をオーバーシュート、上がり下がりの振れをハンチングと言う。

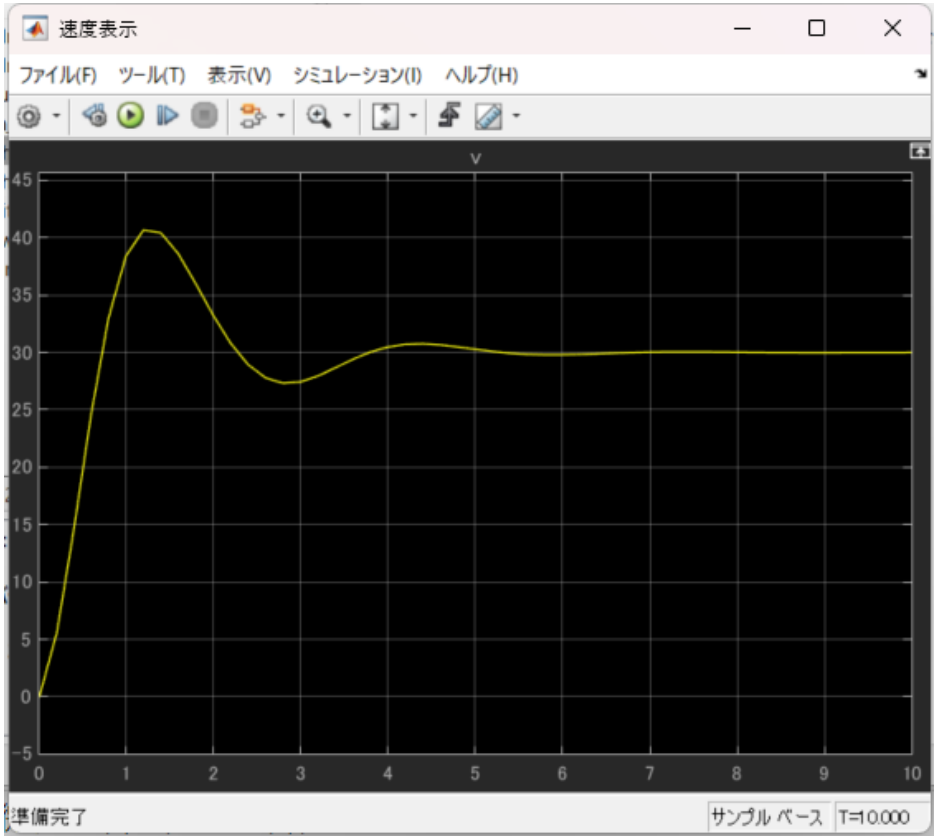
　積分要素を大きくするとこのような問題があるので注意が必要。



SIMLINK 10. 過渡特性

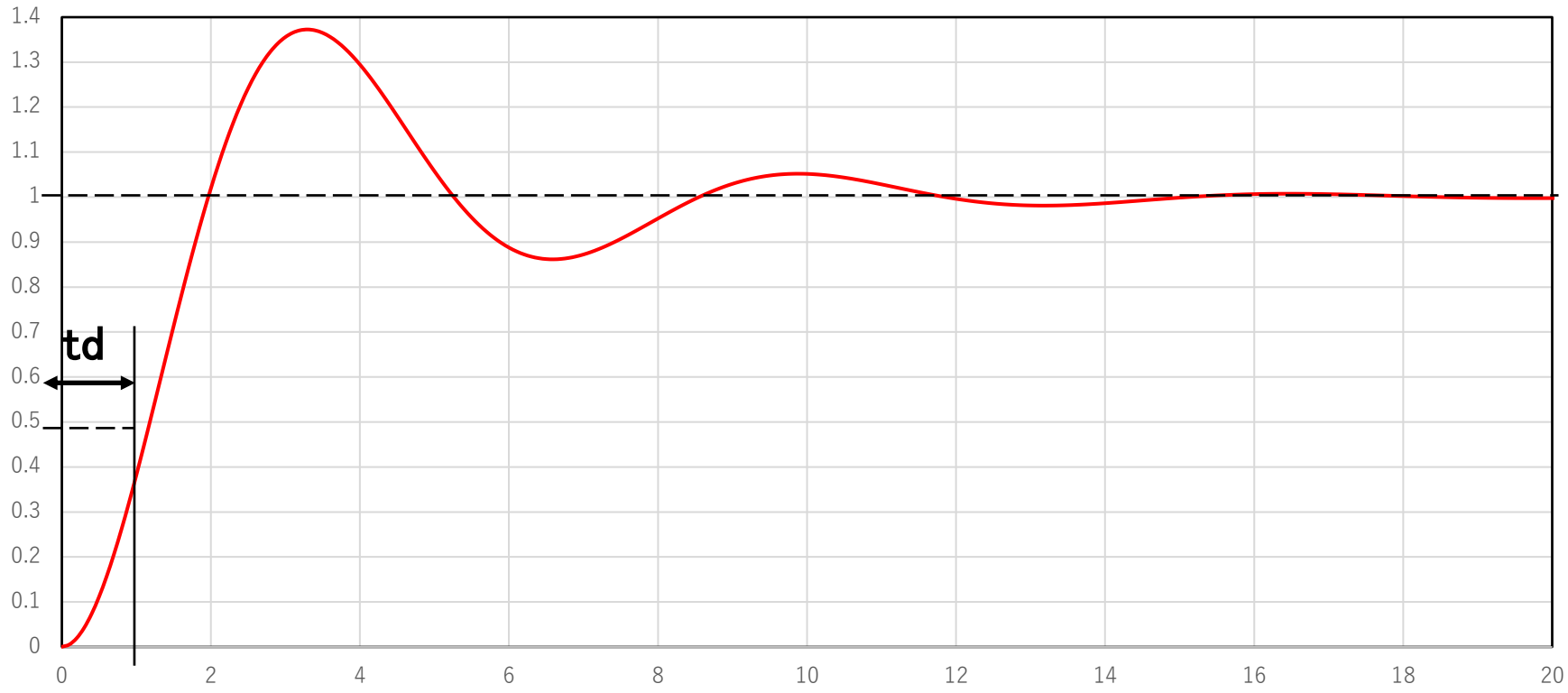
（1） 概要

　PI制御で積分要素を大きくすると右図のようにオーバーシュートやハンチングが発生することになる。
　又比例要素や積分要素が小さいと、設定値に到達するまでの時間が長くなる特性がある。
　ここでは、制御システムの良し悪しを評価する基準につい説明する。



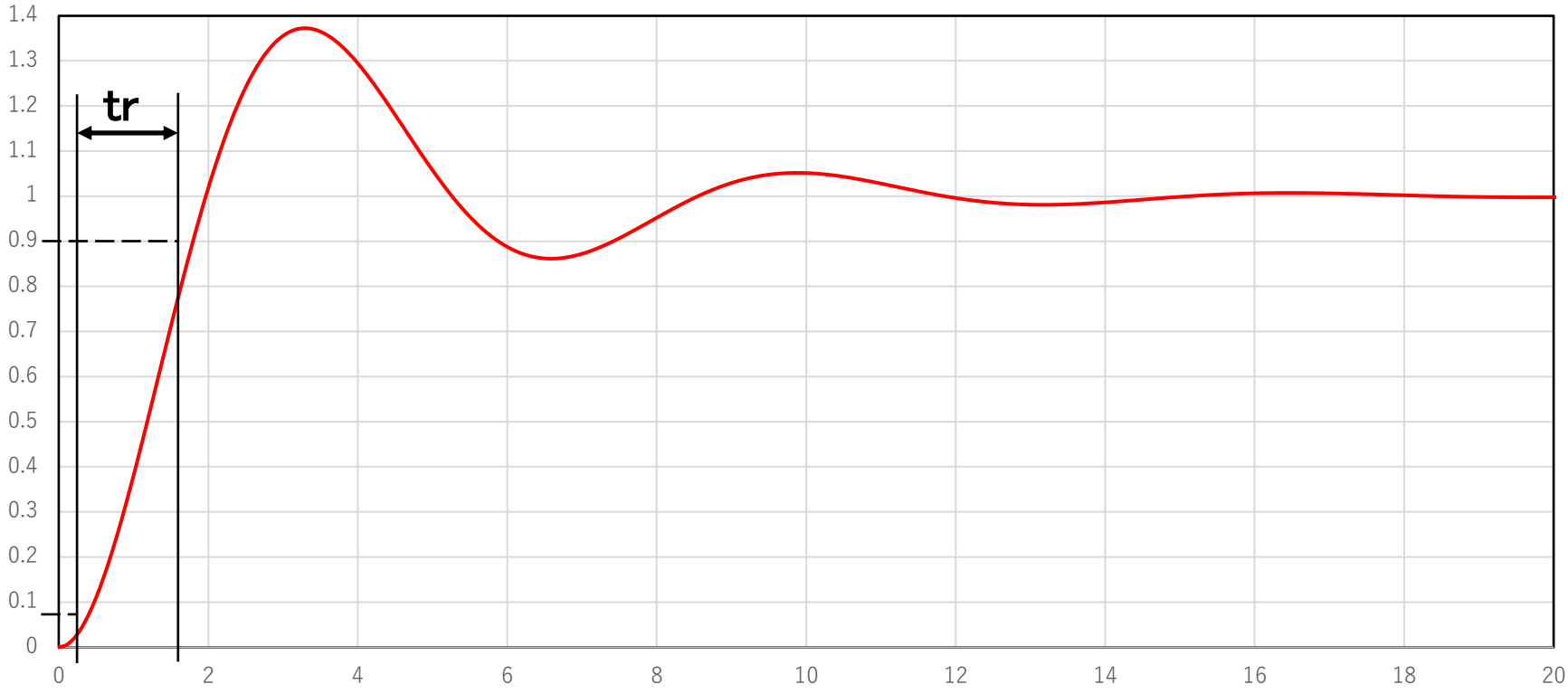
（2） 遅れ時間（td）

応答が定常状態の50%の所に達するまでの時間を示す。



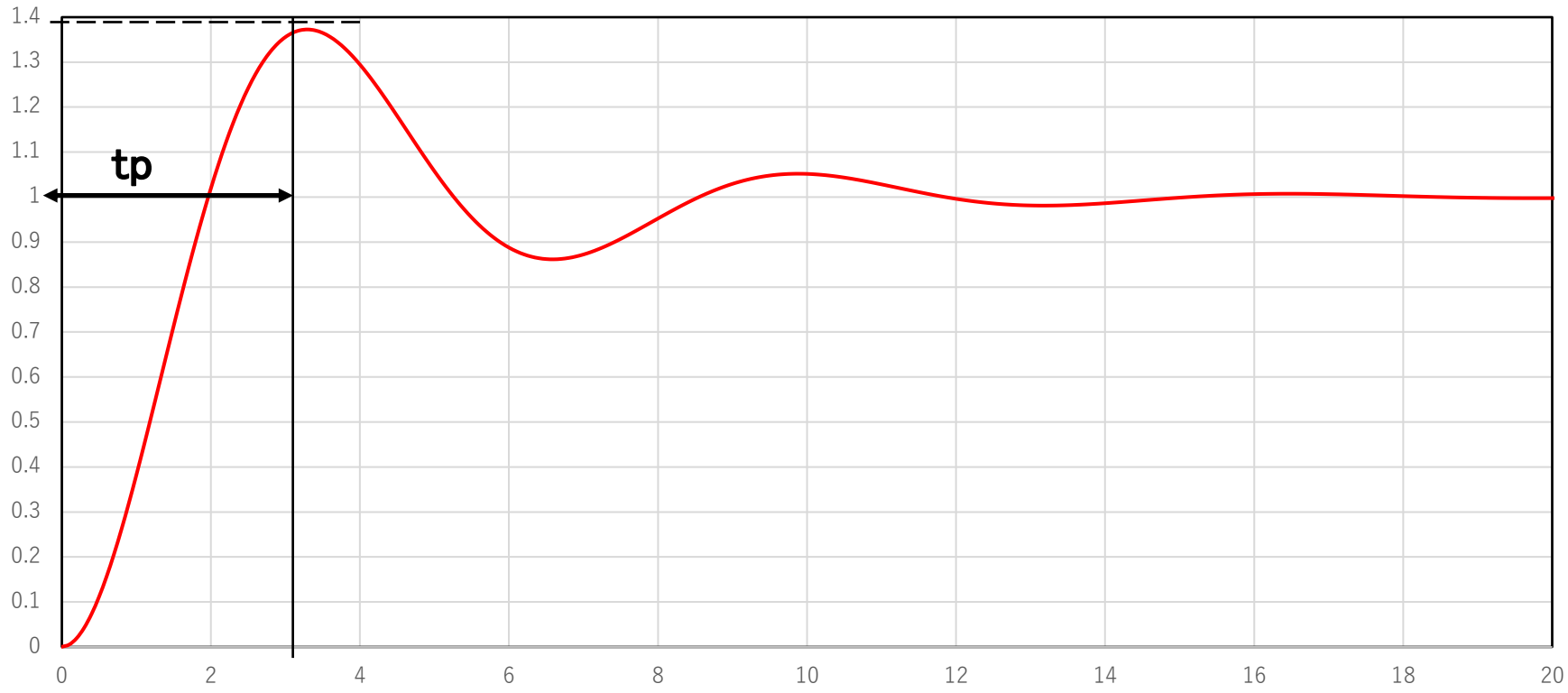
（3） 立ち上がり時間（tr）

応答が最終値の10%から90%までに達する時間を示す。減衰振動的な応答のシステムでは、普通、0%～100%や5%～95%が使われることもある。



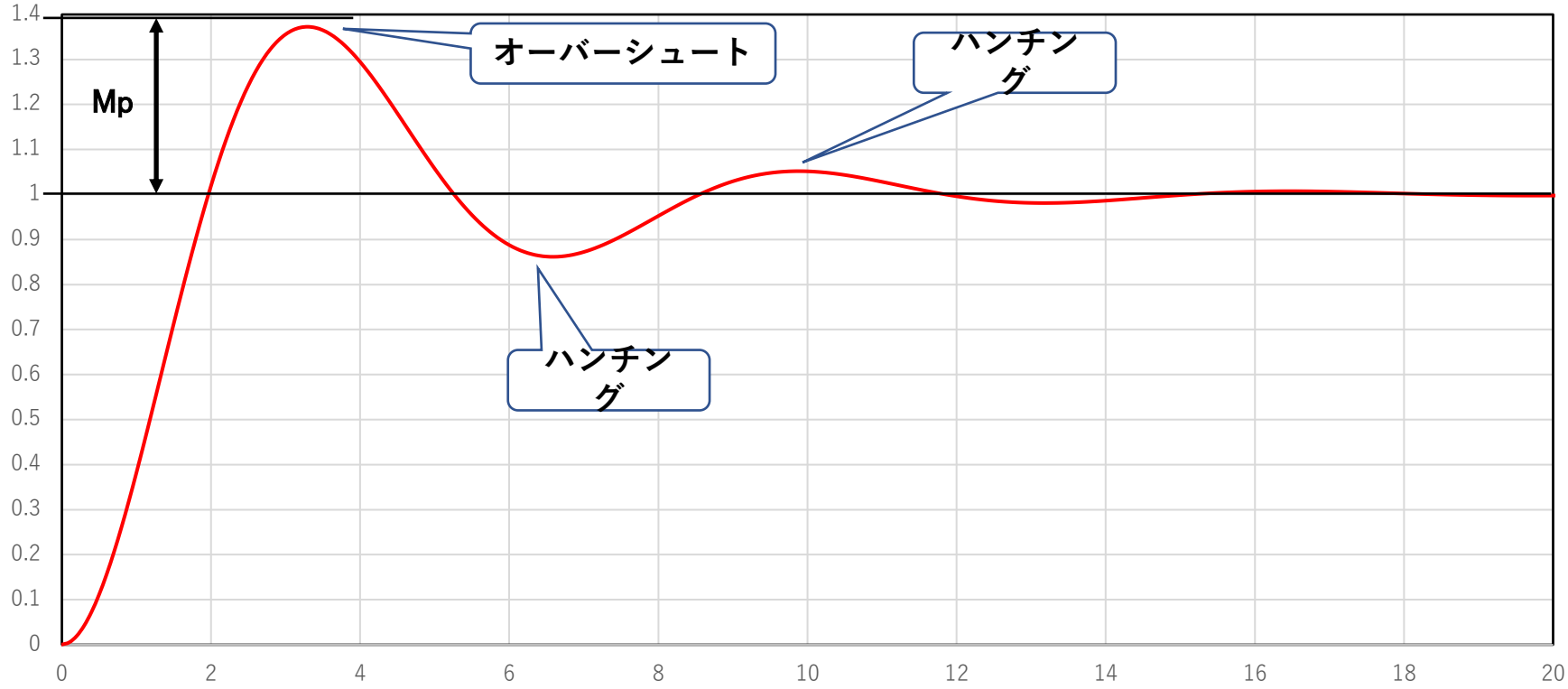
（4） ピーク時間(tp)

行き過ぎの第1ピークまでの時間を指す。



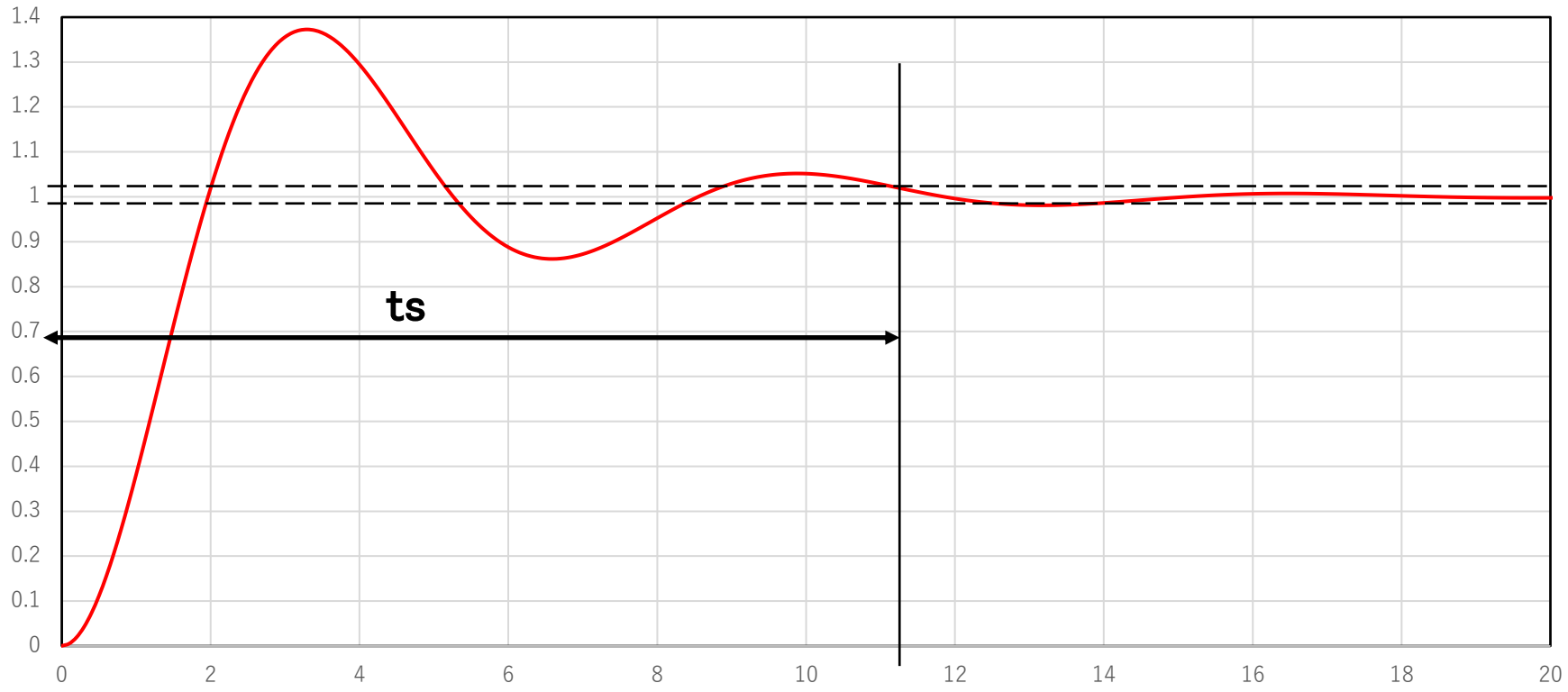
（5） 最大行き過ぎ量（オーバーシュート）（Mp）

応答の定常状態に対する最大ピークの値を指す。パーセント（%）で現されることもある。 最大行き過ぎ量はシステムの相対安定性と直接比例しいる。



（6） 整定時間（ts）

応答が定常状態の2%または5%の所に達するまでの時間を指す。



SIMLINK １１．比例積分微分制御（PID制御）

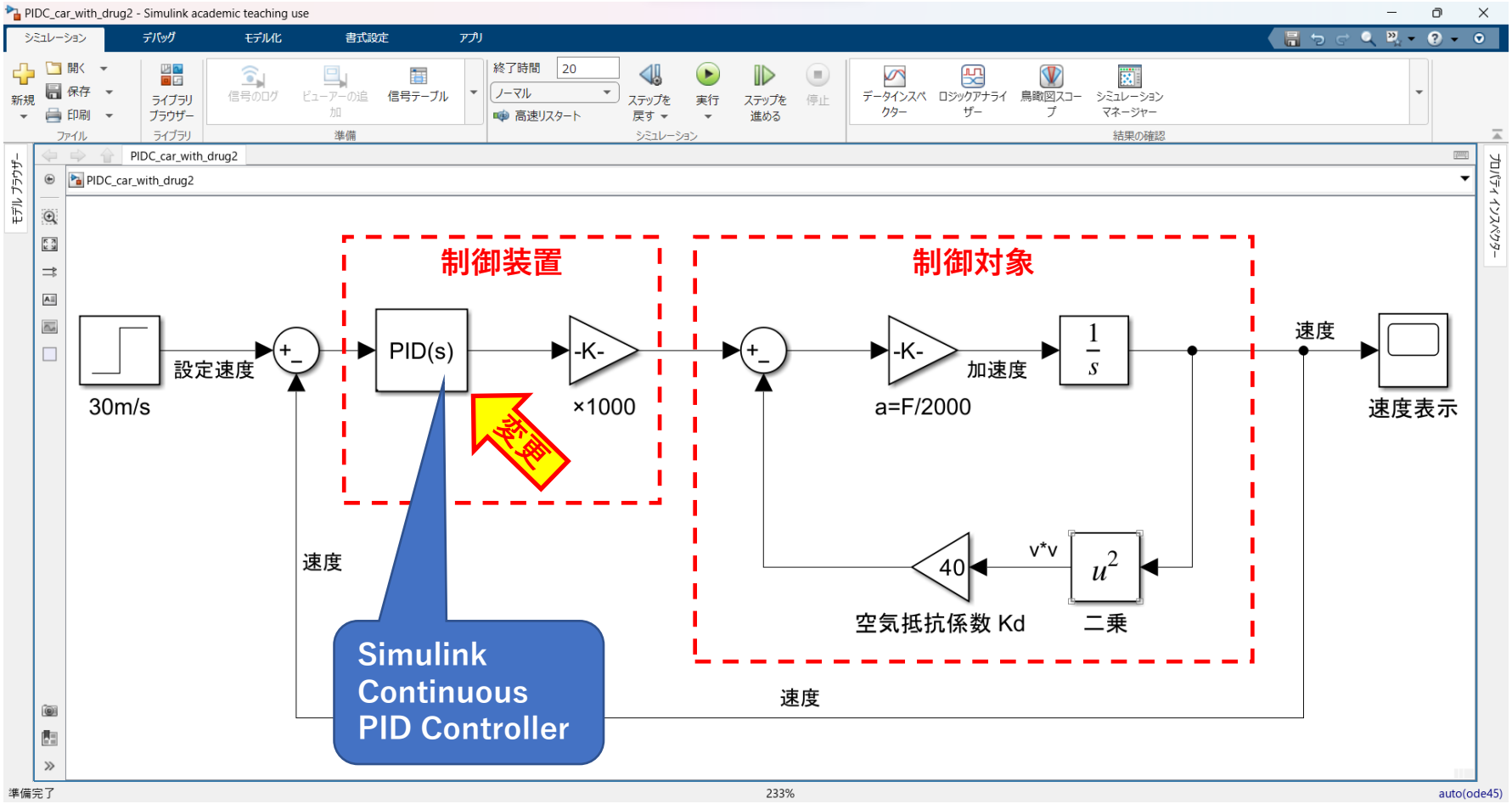
（1） 概要

「SIMLINK １１．比例積分制御（PI制御）」の制御システムでは積分要素を大きくすると立ち上がりが早くなるが、オーバーシュートとハンチングが発生している。

この対策として、制御に微分要素を追加することによりオーバーシュートとハンチングを低減でき、制御特性を向上することができる。

PID制御は下のブロックを使用する。

（2） モデルの作成


PIDC_car_with_drug2 - Simulink academic teaching use
シミュレーション
デバッグ
モデル化
書式設定
アプリ
新規
開く
保存
印刷
ファイル
ライブラリブラウザー
ライブラリ
信号のログ
ビューアーの追加
信号テーブル
準備
終了時間
20
ノーマル
高速リスタート
ステップを戻す
実行
ステップを進める
停止
シミュレーション
データインスペクター
ロジックアナライザー
鳥瞰図スコープ
シミュレーションマネージャー
結果の確認
モデル ブラウザー
プロパティ インスペクター
PIDC_car_with_drug2
制御装置
制御対象
30m/s
設定速度
PID(s)
-K-
×1000
変更
-K-
a=F/2000
加速度
速度
速度表示
速度
40
空気抵抗係数 Kd
v*v
二乗
速度
Simulink
Continuous
PID Controller
準備完了
233%
auto(ode45)

（3） 実行結果と考察―1

空気抵抗の無い車の速度制御を考える

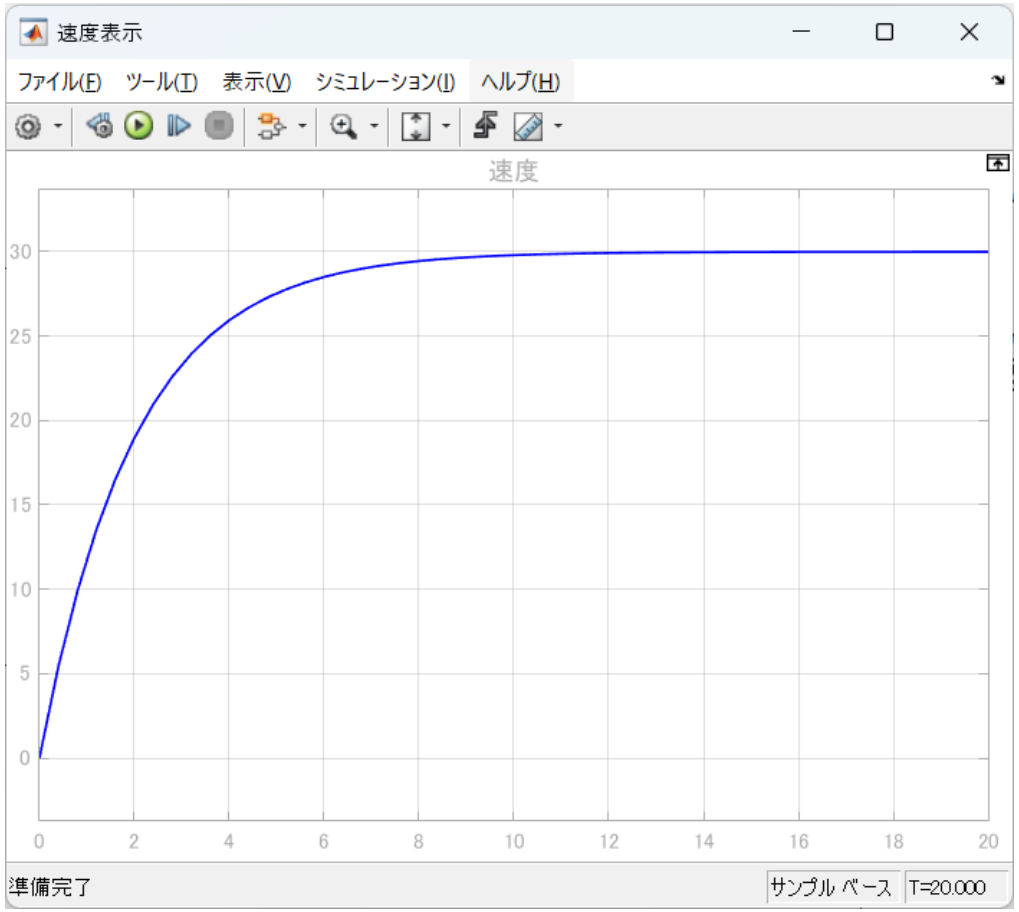
P=1.0

I=0.0

D=0.0

空気抵抗係数 Kd = 0

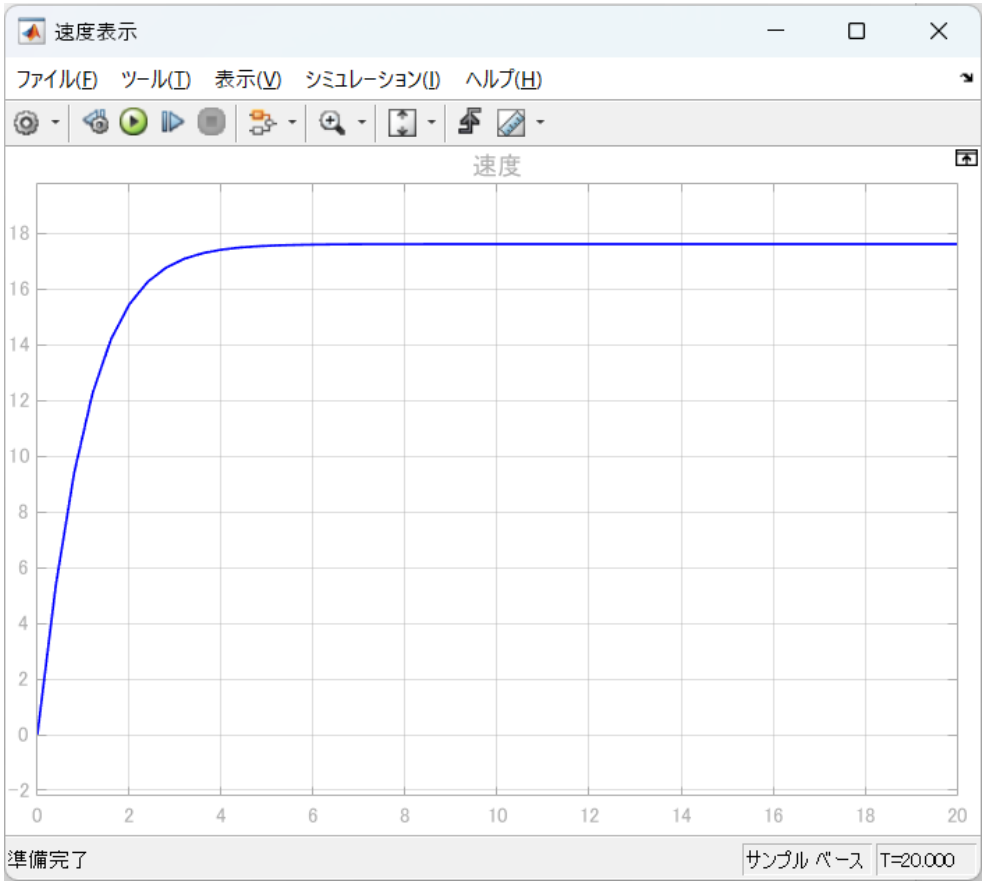
速度は１２秒程度で設定速度30[m/s]に到達している。



（4） 実行結果と考察―2

　空気抵抗を加えた状態を考える。
P=1.0
I=0.0
D=0.0
空気抵抗係数 Kd = 40

　空気抵抗が有るため、最高速度が約17.5[m/s]となっており、設定値30[m/s]に到達しない。



（5） 実行結果と考察ー3

　比例要素を大きくしてみる。

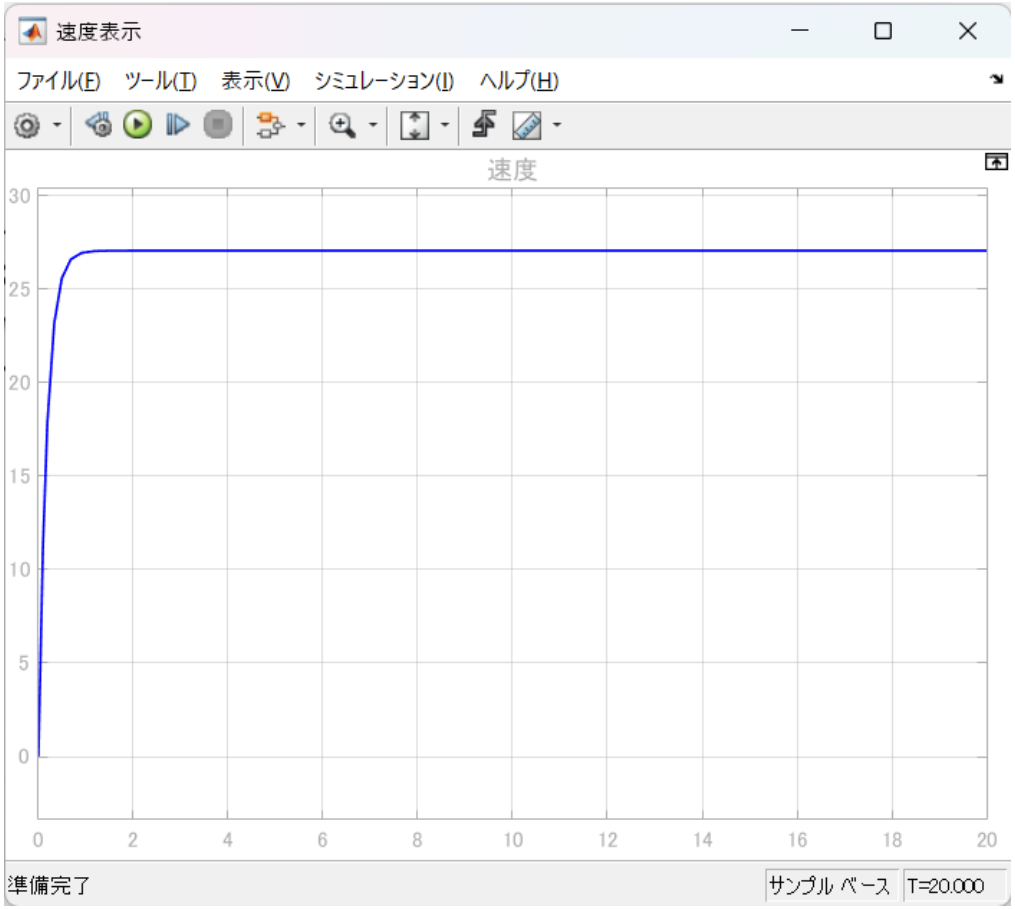
P=10.0

I=0.0

D=0.0

空気抵抗係数 Kd = 40

　最高速度が約27[m/s]となり、前頁より大きくなったが、空気抵抗のため、依然設定値30[m/s]に到達しない。



（6） 実行結果と考察―4

積分要素を加えてみる。

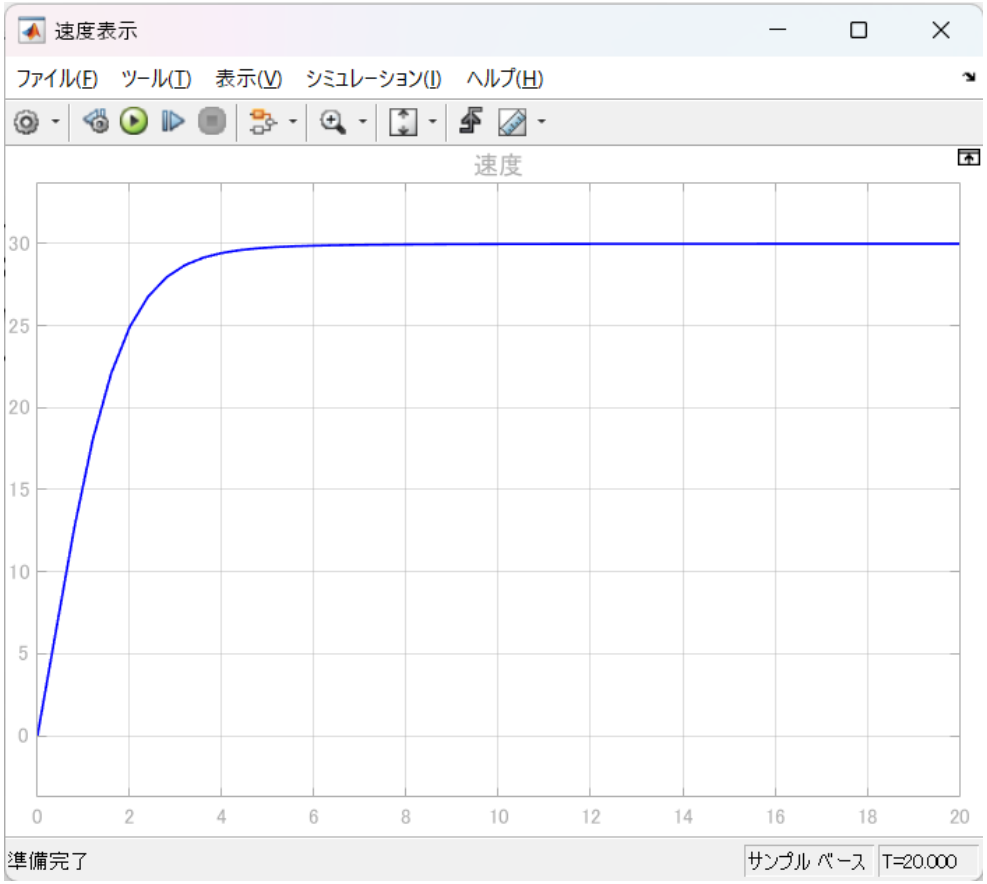
P=1.0

I=1.0

D=0.0

空気抵抗係数 Kd = 40

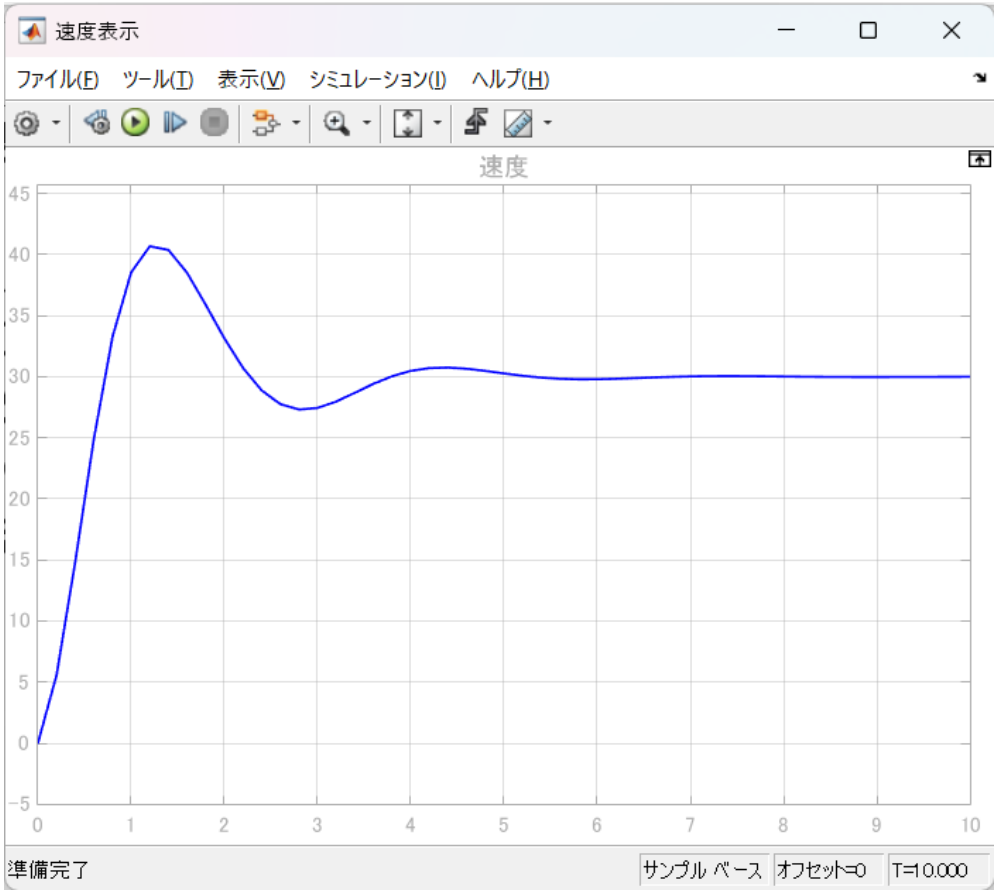
積分要素の効果のため、設定値30[m/s]まで速度が上昇している。



（7） 実行結果と考察―5

　さらに積分要素を大きくしてみる。
P=1.0
I=10.0
D=0.0
空気抵抗係数 Kd = 40

　積分要素が大きすぎるとオーバーシュートとハンチングが発生する。



（8） 実行結果と考察―6

　微分要素を設定する。

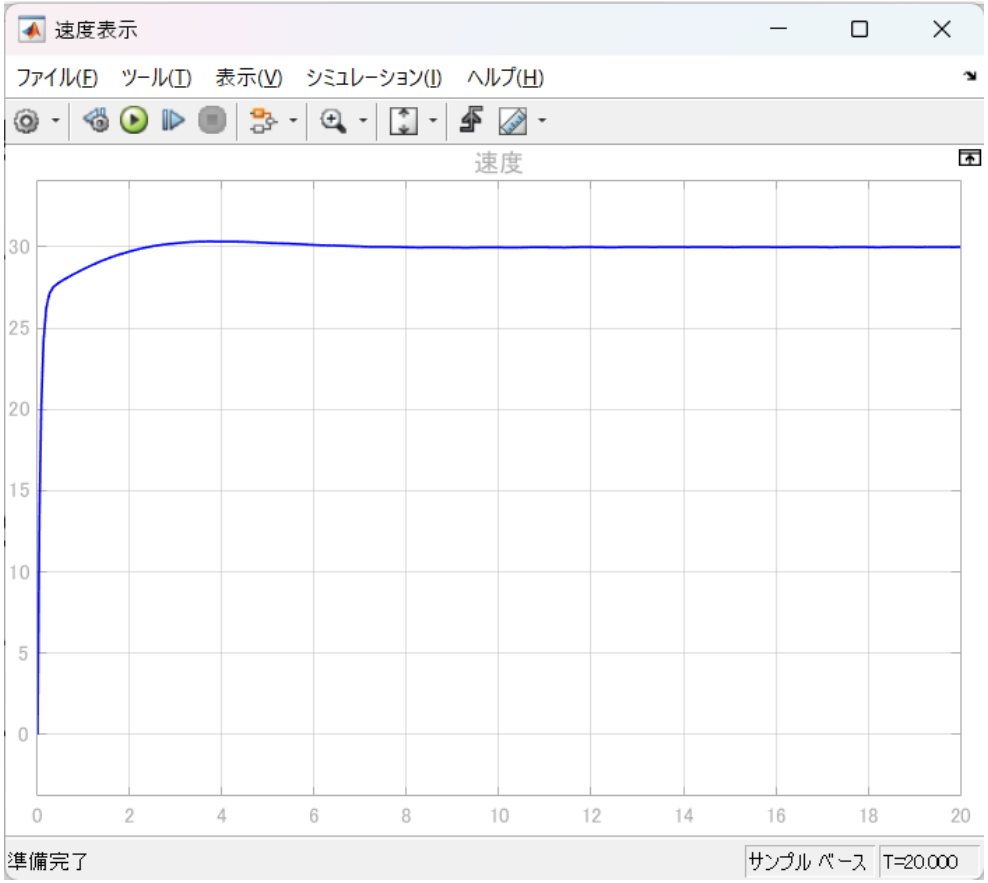
P=1.0

I=10.0

D=10.0

空気抵抗係数 Kd = 40

　微分要素を設定することによりオーバーシュートを低減することができる。



（9） まとめ

　空気抵抗が有る環境での速度制御は、積分要素が必須となる。
　又積分要素が多き過ぎる場合はハンチングが発生する場合があり、微分要素を加えることによりハンチングを低減することができる。