はじめに

本セミナーの目的は、はじめてMATLABをご覧になる方に、実際にノートPCを用いたMATLABの実習を通して、MATLABの簡便な操作性、対話的プログラミング、豊富なグラフィックス機能などを体験していただくことです。

ここで取り上げることができるのは基本的な解析機能のごく一部に限られますが、本セミナーが、MATLABを皆様の業務に役立てていただく上でご参考になれば幸いです。

```
>> demo
```

なお、テキスト内では、コマンドとして実行していただくものは、右の欄または本文の四角い枠内に、プロンプト(>>)マーク付きで記述しています。コマンドを書いてあるとおりに入力し、Enterキーを押すと実行されます。

目次

※本テキストは、Windows環境にインストールされたMATLAB2012bをベースに記述されております。UNIXおよびLinux環境にインストールされたMATLABや、MATLABの他のバージョンとは若干インタフェースの表示や機能が異なりますのでご注意ください。

MATLABとは？

MATLABは、MATLABプロダクトファミリーのコアモジュールで、数値計算、データ解析、グラフィックス機能といった、各種解析に必要な統合開発環境を提供しています。また、プログラミング言語として、これらの機能を利用して効率的なプログラム開発を行うことも可能です。

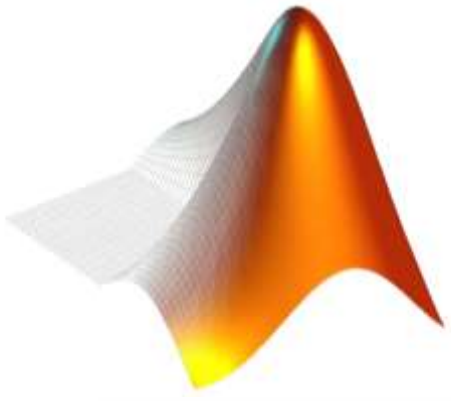
さらにこのMATLAB基本モジュールを主軸とし、最適化や統計処理といった解析から、各種ディジタル信号処理、制御系設計、金融工学、バイオインフォマティクスなど、各種分野に対して利便性の高い専門ツールが多数提供されています。

MATLABの特長としては、次の点が挙げられます。

1) 簡便なデータ操作と対話型プログラミング環境
変数の型宣言や配列の大きさを宣言する必要がなく、行列演算を直接記述することができるため、他言語と比べ非常に直感的にプログラミングを行うことができます。

2) 豊富なグラフィックス、GUI構築機能
多種多様なグラフィックス機能を提供しており、複雑な図も容易に描くことが可能です。また、GUIを構築する機能も備わっており、MATLAB環境上でオリジナルのアプリケーションを作成できます。

3) 外部インタフェース、アプリケーション開発機能
様々な外部インタフェースを提供しており、各種データベースやExcel等とシームレスに連携を図ることができます。さらに、MATLAB上で構築したプログラムを、MATLABがインストールされていないマシン上でも実行することができるスタンドアロン形式に変換するツールも提供しており、これを用いれば、開発したアプリケーションを容易に社内展開することができます。

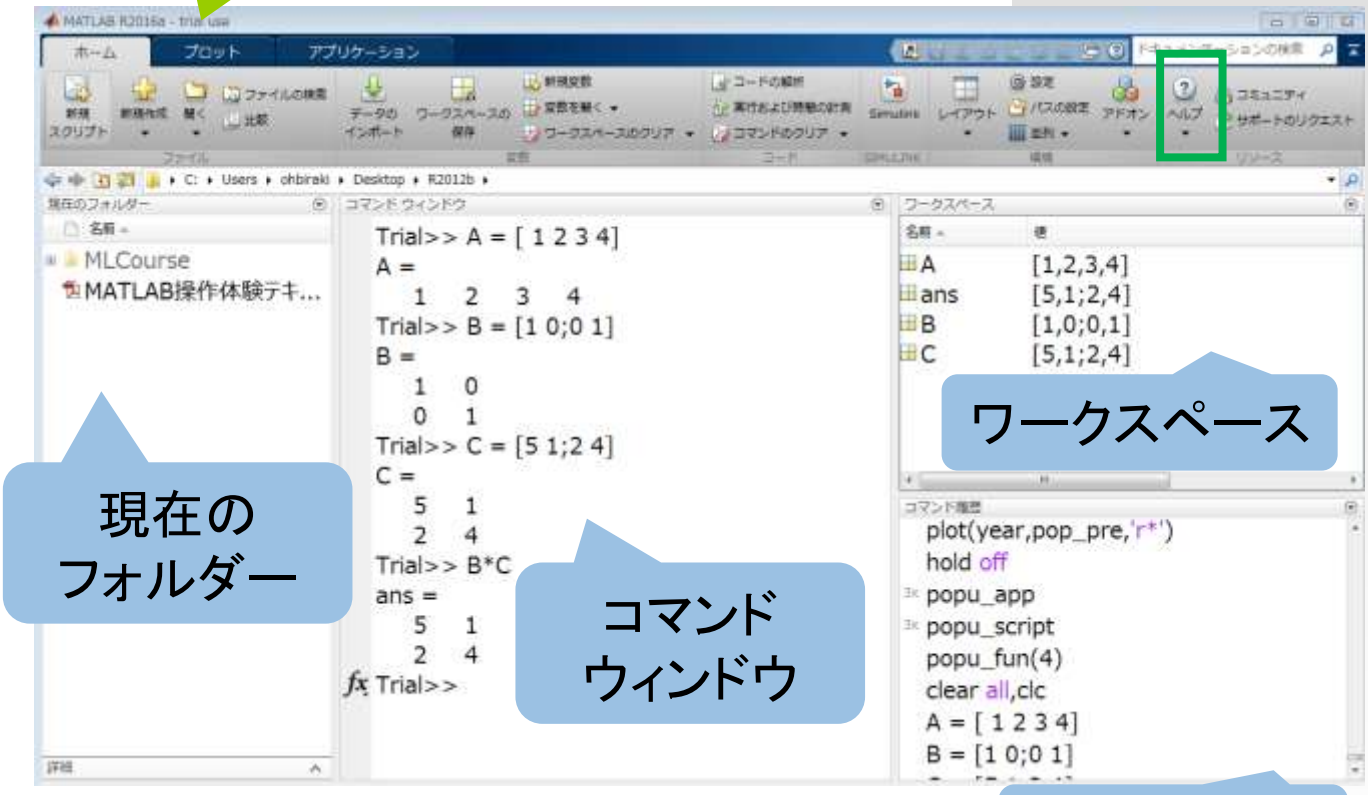


第1章　MATLABの基本操作

この章では、MATLABデスクトップ環境の紹介とともに、後半で行うプログラミングに必要な基本操作として、行列操作とそれらの演算方法などについて説明します。

1.1　デスクトップ環境

MATLABデスクトップ環境の概要を紹介します。



1.2 基本データ操作

ここでは、MATLAB操作のごく基本となる概念を、実行コマンドとともに紹介します。

MATLABでは、与えられたコマンドを以下のように解釈します。すなわち、イコール(=) 記号の右側に記述された計算結果を、イコール記号の左側で定義された変数に代入するという形式です。ここでは、鍵括弧([])記号を使って、2×2の行列を定義しています。行列内の要素の区切りにはスペースかカンマ(,)を、改行には、セミコロン(;)またはキャリッジリターンを使用します。

```
>> A = [1 2; 3 4]
```

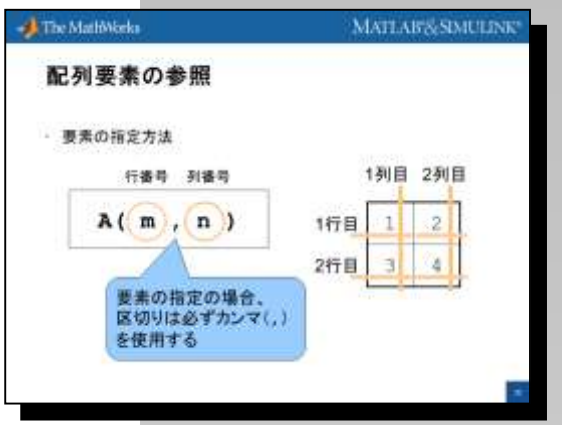
なお、この際、代入する変数について、事前に型や配列サイズを宣言しておく必要はありません。ステートメントの最後にセミコロンを付けると、実行結果が表示されなくなります。これは、データが大きい場合やプログラムの内部処理を記述する場合などに便利です。

行列を定義する方法にはこの他に、行列を生成する関数を用いるものや、各種ファイルから読み込むもの、コロン(:)を利用する特別な方法などがあります。コロンを利用すると、等間隔のベクトルを簡単なコマンドで作成することができます。ここでは、コロン(:)を使って定義した時間データtを使って正弦波、余弦波をプロットする例をご紹介しています。

また、MATLABでは全ての数値データを倍精度浮動小数点型の配列として管理しますので、高精度の演算が可能です。

```
>> A = 1
>> B = 2
>> A * B

>> A = [1 2 3 4]
>> A = [1 2; 3 4]
>> B = [1 0; 0 1]
>> A * B


>> r = randn(3,4)
>> x = xlsread('dt.xls')

>> x = 1:5
>> x = 0:2:10

>> t = 0:pi/20:2*pi
>> y1 = sin(t);
>> plot(t,y1)
>> hold on

>> y2 = cos(t);
>> plot(t,y2)
>> hold off

>> close
>> plot(t,y1,'b',…
     t,y2,'r')
```

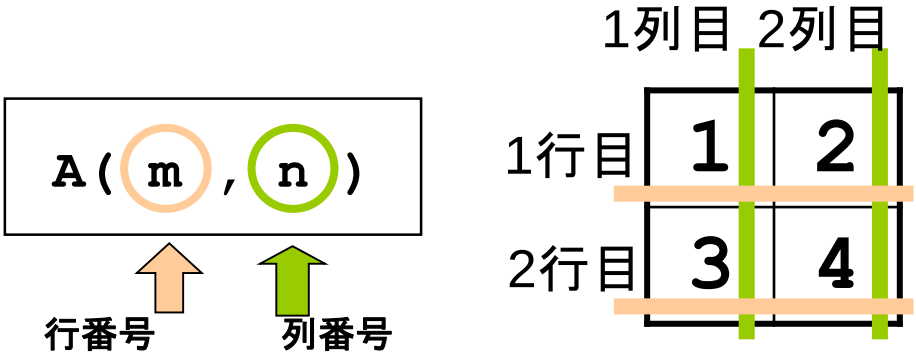
1.2 基本データ操作(続)



次に、配列内の要素の参照方法を説明します。ある変数の一部を取り出したり置き換えたりしたい場合に、どの部分を参照するか指定する必要があります。2次元の行列の場合、行番号と列番号を指定することで参照が可能です。丸括弧(())を使って、次のように参照します。

なお、番号の指定には、スカラ値だけでなくベクトル(複数要素の指定)やコロン(全部を指定)を使うこともできます。

```
>> A(2,1)
>> A(2,:)
```




```
>> A(1,1) = 10
>> A(:,2) = 5

>> C = rand(10,1)
>> idx = C > 0.5
>> D = C(idx)
```

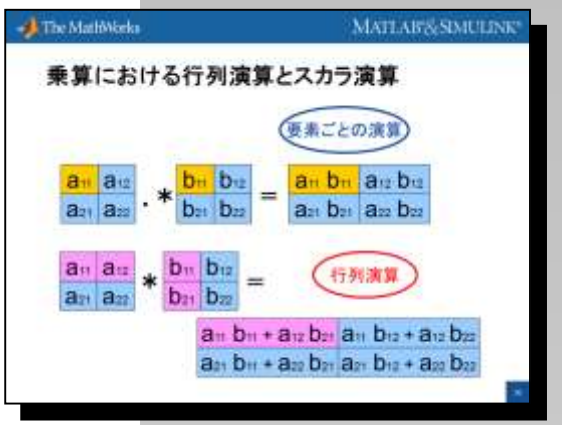
表1.1 代表的な行列生成関数と演算子

zeros	ゼロ行列	rand	一様乱数
ones	全ての要素が1の行列	randn	正規乱数
eye	単位行列	linspace	線形等間隔ベクトル
diag	対角行列	logspace	対数等間隔ベクトル
magic	魔方陣行列	:	等間隔ベクトル

表1.2 代表的な配列操作関数と演算子

size	配列の大きさを取得	reshape	行列のサイズを変更
length	ベクトルの長さを取得	fliplr	行列を左右反転
'	共役転置	flipud	行列を上下反転
.'	転置	rot90	行列を90°回転
diag	対角行列の作成と抽出	triu	上三角行列の抽出

1.3 行列演算



MATLABでの数値演算は要素ごとではなく、行列演算として直接行列を扱えるところに最大の特徴があります。
行列の四則演算を通してこれを確認してみましょう。

要素ごとの乗算

a11	a12
a21	a22

.*

b11	b12
b21	b22

=

a11 b11	a12 b12
a21 b21	a22 b22

```
>> A+B
>> A.*B
>> A*B
>> A./B
```

行列積の計算

a11	a12
a21	a22

*

b11	b12
b21	b22

=

a11 b11 + a12 b21	a11 b12 + a12 b22
a21 b11 + a22 b21	a21 b12 + a22 b22

[例題 1] 例えば、下記の連立方程式を解くとします。

$$\begin{cases} x+5y=7 \\ 2x+4y=8 \end{cases}$$

連立方程式を行列方程式に変更して
未知数を求めます。

$$\begin{bmatrix} 1 & 5 \\ 2 & 4 \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 7 \\ 8 \end{bmatrix}$$

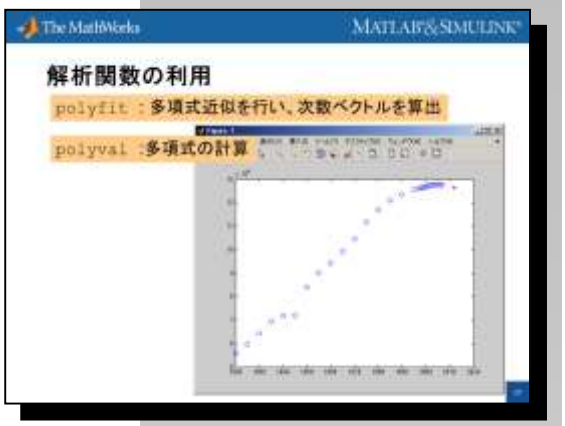
A　x　b

$$A * x = b$$

$$x = A^{-1} * b$$

```
>> A = [1 5; 2 4];
>> b = [7;8];
>> x = inv(A)*b
```

1.4 解析関数の利用



ここでは、過去の人口の移り変わりのデータを元に2020年の人口の予測を行いたいと思います。今回は、Excelファイルにから読み込んだ年数とそのときの人口のデータから多項式方程式の係数ベクトルを求め、係数ベクトルから未来の人口値を予測します。

ここでは、Excelファイルpopulation.xlsから年数及び人口データを読み込み、そのデータから多項式近似を行います。では、まずは全体のデータから処理を行うデータ(2列目のデータ)を取り出し、プロット表示を行います。

```
>> [D,hd] = xlsread('population.xlsx');
>> t = D(:,1);
>> pop = D(:,2);
>> year = 2020;
>> plot(t,pop,'o')
```

読み込みを行うファイルの1列目に年データ、2列目に人口のデータの記述があります。

次に、特異なデータを取り除くために、「論理インデックス」を使います。

```
>> idx = t<1940 | t>1945;
>> t1 = t(idx);
>> pop1 = pop(idx);
```

多項式近似を行うために、年数データ(t1)と取り出した人口データ(pop1)から、polyfit関数を用いて多項式の係数ベクトルを求めます。今回は、フィッティングする方程式の次数を3次とします。

```
>> [p,S,mu] = polyfit(t1,pop1,3)
```

今回、フィッティングさせる3次式

$$p_1x^3 + p_2x^2 + p_3x + p_4$$

$$\mathrm{p} = [\,p_1\ \ p_2\ \ p_3\ \ p_4\,]$$

上記で求めた係数ベクトルを使って2020年の人口予測を行います。

```
>> pop_pre = polyval(p,year,S,mu)
```

上記を実行しますと、フィッティング値から予測される2020年の人口が求まります。

では、予測したデータ pop_pre を最初にプロットしたグラフに追加して表示をしてみましょう。グラフィックスに追加で表示を行う場合は、`hold` コマンドを利用します。

```
>> hold on
>> plot(year,pop_pre,'r*')
>> hold off
```

hold onとすると有効なFigureがホールド状態になります。ホールド状態を解除する場合はhold offとしてください。

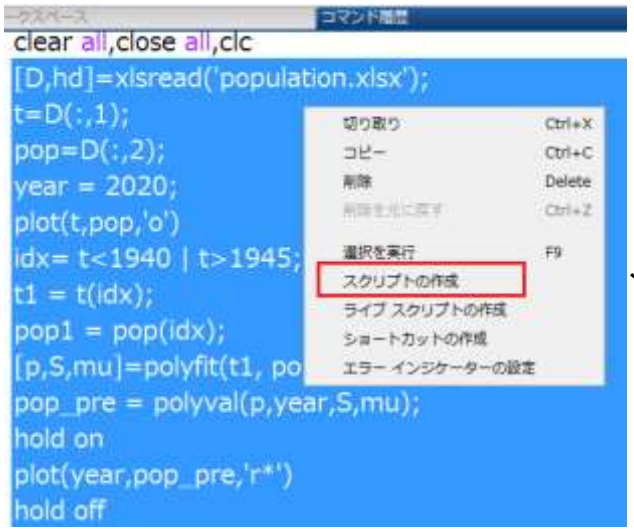
今回は、3次方程式にフィッティングする処理を行ってきましたが、この次数を変更すると、フィッティングデータから予測される2020年の人口の値が変わってきます。

第2章　プログラミング機能

この章では、第1章の例題を用いて、MATLABの簡便なプログラミング環境についてご紹介します。

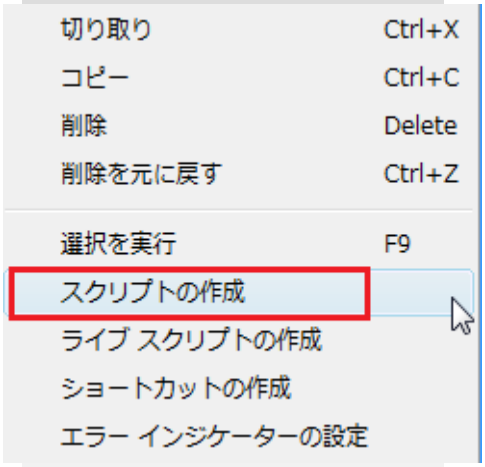
2.1　スクリプトMファイルの作成

“1.4 解析関数の利用”で行った操作を、プログラムにまとめて保存してみましょう。MATLABに付属している専用のエディタに、直接プログラムを記述することも可能ですが、今回は、コマンド履歴を利用して作成しましょう。保存したいコマンドをまとめて選択し、右クリックして表示されるポップアップメニューから、”スクリプトの作成”を選択します。



注意！

コマンド履歴からコマンドを選択する場合、Ctrlキーを押しながらクリックすると複数のコマンドを選択できます



メニュー部分の拡大

エディタが自動的に立ち上がりますので、名前をつけて保存します。popu_scriptと入力しますと、現在のフォルダーにプログラムファイル(通称Mファイル)であるpopu_script.mが作成されます。

コマンドウィンドウ上でMファイル名をタイプすれば、このプログラムを実行することができます。実行する前に、clearコマンドによりメモリ上の変数を消去しておくと、より分かりやすいでしょう。余分な行が含まれていたり、エラーが起きたりする場合はこの時点で修正を行ってください。

```
>> clear
>> popu_script
```

2.2 関数Mファイルの作成

2.1で作成したスクリプトMファイルは、コマンドを保存しただけのバッチファイル型のプログラムでした。これを、入力と出力を持った関数形式(関数Mファイル)にしたい場合、プログラムの1行目にfunction宣言行を挿入します。

```
1  function pop_pre = popu_fun(order)
2  %% 日本の人口推移
3  %  エクセルからデータ読み込み
4  [D,hd]=xlsread('population.xlsx');
5  t=D(:,1);
6  pop=D(:,2);
7  year = 2020; % 予測年
8  % 特異データの除外
9  idx= t<1940 | t>1945;
10 t1 = t(idx);
11 pop1 = pop(idx);
12 % プロット（移動しました！）
13 plot(t1,pop1,'o')
14 % 多項式近似
15 [p,S,mu]= polyfit(t1, pop1, order);
16 % 指定した年の人口予想
17 pop_pre = polyval(p,year,S,mu);
18 % 予測値の追加プロット
19 hold on
20 plot(year, pop_pre, 'r*')
21 hold off
22 popu_graph
```

ファイル名：**popu_fun.m**

フィッティングする次数を入力引数として指定

出力引数

追加記述

popu_graph.mは、グラフィックスを表示するための年数データの作成、軸の設定、フィッティングした2020年の人口データの表示位置の設定、タイトル、x軸・y軸ラベルの表示を行っています。尚、最初に表示を行ったプロットデータに追加記述するためにholdコマンドを使っています。

先ほど作成したスクリプトMファイル（popu_script.m）の記述を少し変えてファンクションMファイルを作成してみましょう。変更を行う箇所は以下です。

① カーブフィットする次数（order）の変更：polyfit(t1,pop1,order)
② ”popu_graph”をプログラムの最後に追記する。

”popu_graph”は、予めこちらで用意したグラフィックスの表示を行うためのスクリプトMファイルです。関数Mファイル作成のポイントは、次の3点です。

① function宣言行では入力変数と出力変数の関係を記述します
② フィッティングを行う次数を入力引数として与えます
③ 出力変数となる引数pop_pre（予測値）を求めます

>> edit popu_graph
を実行すると指定したファイル名のエディタが表示され、ファイルの中を確認することができます。

以下、フィッティングを行う次数を指定して、作成した関数Mファイルpopu_funを実行してみましょう。フィッティングを行った2020年の予測値を出力変数として出力します。4次の多項式にフィッティングする場合、下記のように実行します。

```
>> clear all
>> pop2020 = popu_fun(4)
```

このように、変数をデスクトップのワークスペースとやり取りするような関数Mファイルとすることで、他のデータに適用したり、階層化して複雑なプログラムを構築したりすることが容易になります。

第3章　アプリケーション開発

この章では、プログラムを実行する為のアプリの開発と、そのスタンドアロン化を行います。

3.1　アプリの構築

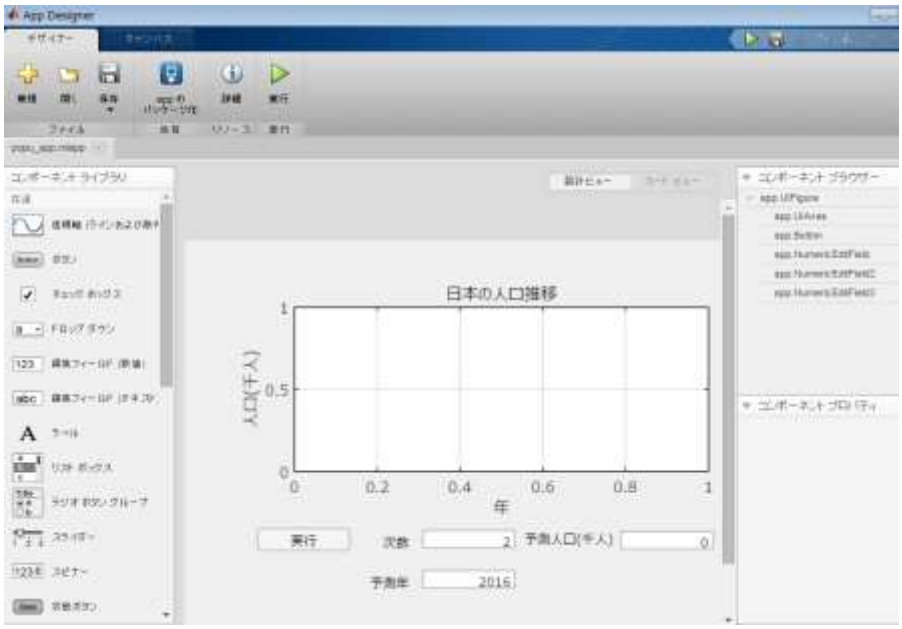
MATLABには、アプリを構築するための環境であるツールが備わっています。appdesignerを利用し、人口データと次数を選択し、グラフィックスを表示、フィッティングデータからの予測したい年の予測値と実データとの誤差を求めるアプリを作成してみましょう。

```
>> appdesigner popu_app
```

まず、appdesignerコマンドよりアプリ設計ツールを立ち上げます。今回は、作成されているファイル名：popu_appを使います。すでに作成されているアプリを起動する際は右記に記してあるコマンドを実行します。立ち上がったpopu_appは、下記のように、座標軸、ボタン、編集フィールド(数値)が配置されており、各種プロパティも設定されています

新規のアプリを作成する時

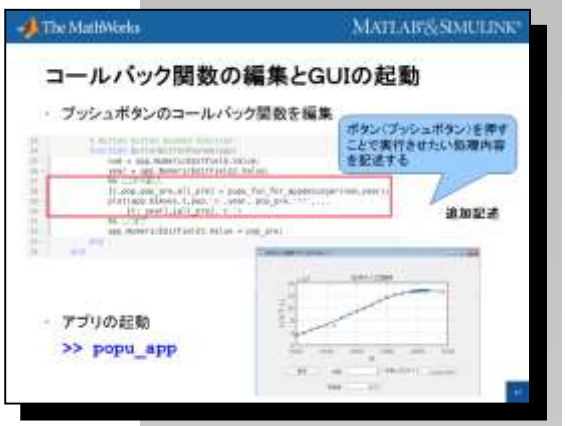
```
>> appdesigner
```



プロパティの設定を行うには、配置した各オブジェクトをクリックすると、右下にある、コンポーネントプロパティから、設定をすることが可能です。設定が一通り終わったら、一度保存を行います。保存を行います保存は左上の保存ボタンをクリックすると、[.mlapp]の拡張子のファイルが生成されます。

各オブジェクト上で右クリック → コンポーネントプロパティを選択

3.1 アプリの構築(続)



次に、実行ボタンを押すと人口データのフィッティングシミュレーションが行われるようにするため、ボタンのコールバック関数を編集します。コールバック関数の編集は、コードビューからプログラムを編集します。

ボタン上(実行)で右クリック → コールバック → ButtonButtonPushedコールバックに移動 を選択

すると、コード画面が立ち上がります。このコードは、アプリの動作内容を記述したプログラムです。自動生成されたコードの中にあるボタンのコールバック関数内に、自動的にカーソルが置かれますので、そのまま実行したいコマンドを追加し、保存します。

今回、アプリの[実行]ボタンを押したときに実行させたい処理は次の処理です。

① 編集フィールド(数値)に記述される次数と予測年を取得

② すでに作成してある関数Mファイルpopu_fun_for_appdesinger.mに値を渡して、予測したい年の予測値などの情報を取得

③ 取得した予測値を下の編集フィールド(数値)とグラフに表示

プログラムに追加する処理は、次のようになります。(28行目から30行目)

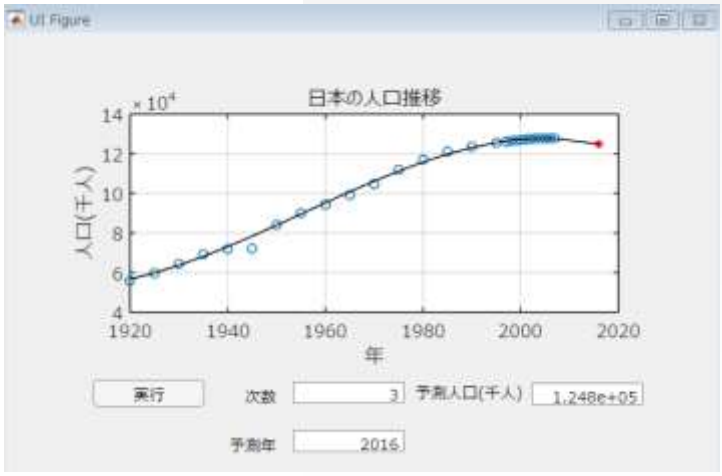
```
[t,pop,pop_pre,all_pre] = popu_fun_for_appdesinger(num,year);
 plot(app.UIAxes,t,pop,'o',year, pop_pre,'r*',...
      [t; year],[all_pre],'k-')
```

これで、作業は終了です。保存が終わりましたら、コマンドウィンドウ上からpopu_appを起動してみましょう。

```
>> popu_app
```

人口のフィッティングを行う次数を指定して[実行]ボタンを押しますと、グラフィックスの表示と予測年の予測値(フィッティングデータ)が表示されます。次数による予測年の推移を御覧ください。

本セミナーでは、簡単なアプリを作成しましたが、アプリ設計ツールを使用して、他の様々なデータを入力できるようにしたり、グラフを多数表示したりと様々な応用が可能となります。



3.2 スタンドアロンアプリケーションの作成

それでは、3.1で作成したアプリを、MATLABがインストールされていない環境でも動くようなスタンドアロン形式に変換してみましょう。MATLAB Compilerと、サポートされるC/C++コンパイラが必要となりますが、コンパイル操作は、Command Windowからmccコマンドを実行するだけ*1で行うことができます。この場合、コ ンパイルコマンドは下記となります。

```
>> mcc -m popu_app
```

コンパイルが無事終了しましたら、popu_app.exeを実行してみましょう。MATLAB上で簡単に確認するには、シェルエスケープ(!)機能を使います。右記コマンドを実行してください。シェルエスケープにより、MATLABを介さずにexeファイルを実行することができます。

```
>> !popu_app
```

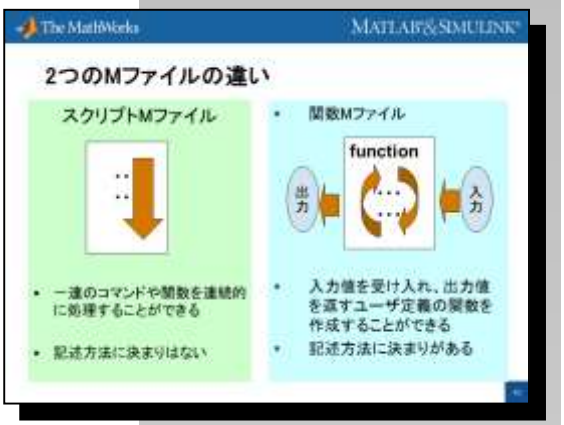
[補足] 配布について

MATLAB Compilerによって作成された実行形式のファイルは、特に制限なく配布することが可能です。エンドユーザには、以下のファイルを配布します。

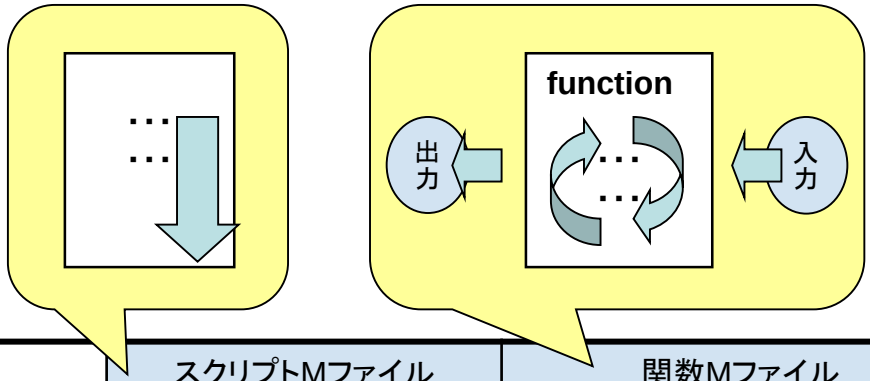
・作成した実行形式のファイル(.exeファイル)
・コンパイル時に作成されたCTFファイル
・ $MATLAB/toolbox/compiler/deploy/win64/MCRinstaller.exe
（$MATLABはMATLABのインストールディレクトリ）

＊1　初めてMATLAB Compilerを使われる場合は、設定コマンド (>> mbuild -setup)を行う必要がありますが、ここでは既に実行済みです。

参考:2つのMファイルの違い



2つのMファイルの機能的な違いを明確にして、効率のよいプログラミングを行うためのポイントを確認します。



	スクリプトMファイル	関数Mファイル
機能	一連のコマンド・関数を連続的に処理できる。	入力値を受け入れ、出力値を返すユーザ定義の関数を作成できる。
構造	特別な記述は必要なし。実行したいステートメントを順に記述するだけ。	入出力の関係を記述したfunctionで始まる関数宣言の行が必要。
変数管理	ファイル内で定義された変数は、ベースWorkspace変数として定義。	ファイル内で定義された変数は、そのファイル内でのみ有効なローカルWorkspace *1 変数として定義。

特に関数Mファイルは、2.2 関数Mファイルの作成 で作成した関数popu_funのような出力引数を持たない関数や、入力引数を持たない関数を作成することもできます。

なお、基本的な関数宣言は次のものです。

```
function [出力引数] = 関数名(入力引数)
```

例えば、2入力、3出力の関数myfunを作成したい場合は、

```
function [y1,y2,y3] = myfun(x1,x2)
```

といった形で記述します。

また、MATLABでは、複数のMファイルを自由に組み合わせたプログラムを作成することも可能です。スクリプトからスクリプト/ファンクションMファイルを呼び出して実行したり、ファンクションからファンクションを呼び出して実行したりすることもできます。

*1 ファンクションMファイルでは、関数を実行するごとにそれぞれの関数用のメモリ領域(ローカルWorkspace)が作成され、その中で変数が定義されます。