

早稲田大学実験レポート グラフの作成方法 ~MALAB編~

！注意！

これに従って作ったら体裁を必ず遵守している保証はありません。
必ず自分でテキストの指示を確認して作成しましょう。

はじめに

早稲田大学理工基礎実験のグラフの体裁には23項目もの指定があり、それら全てを満たした体裁を作るのは一手間かかる。

特に、12番目の項目「プロットは $\odot$, $\triangle$, $\square$ のように、中央に点をつけた記号を用い、その大きさは直径 3 [mm]以上とする。」が曲者で、対応するマーカーがないツールでは一工夫しないといけない。

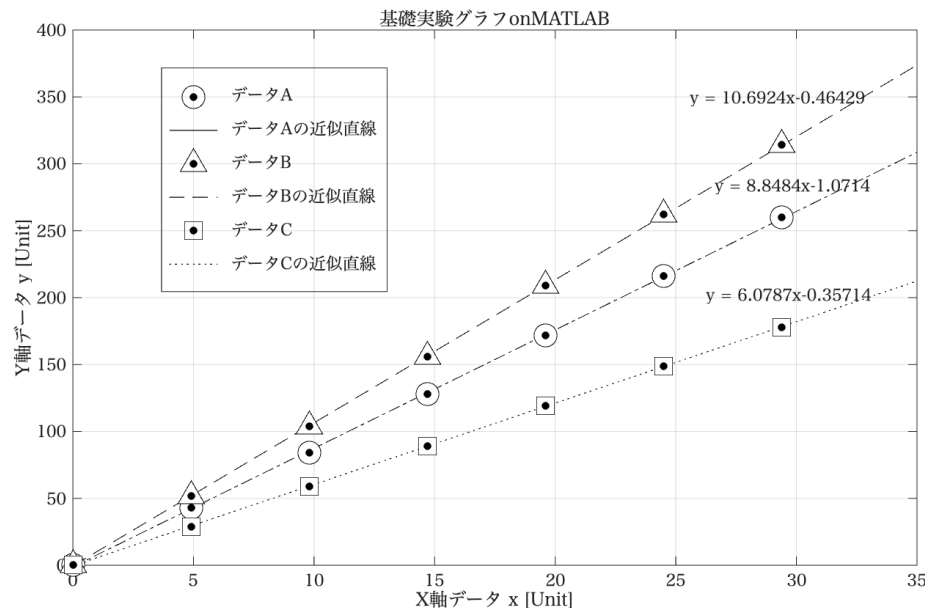
このスライドではMATLAB内でできる限り完結させつつ、体裁を守った作図の方法について説明していく。

実際に作る際にはMATLABの良い部分だけ生かし、面倒な部分は他で補うなどすると良いと思う。

今回作るグラフについて

今回はサンプルのデータを使って右のようなグラフの作成を行っていく。

エクセルでのデータを取り込み、MATLABのコード、コマンドウィンドウ、プロパティエディタなどを使ってグラフを作成し、PNGファイルで保存することを目指す。



作成環境について

一応共有しておく

MATLAB R2025a

OS: macOS Sequoia 15.6.1

資料の作成時期: 2025年9月

macOS



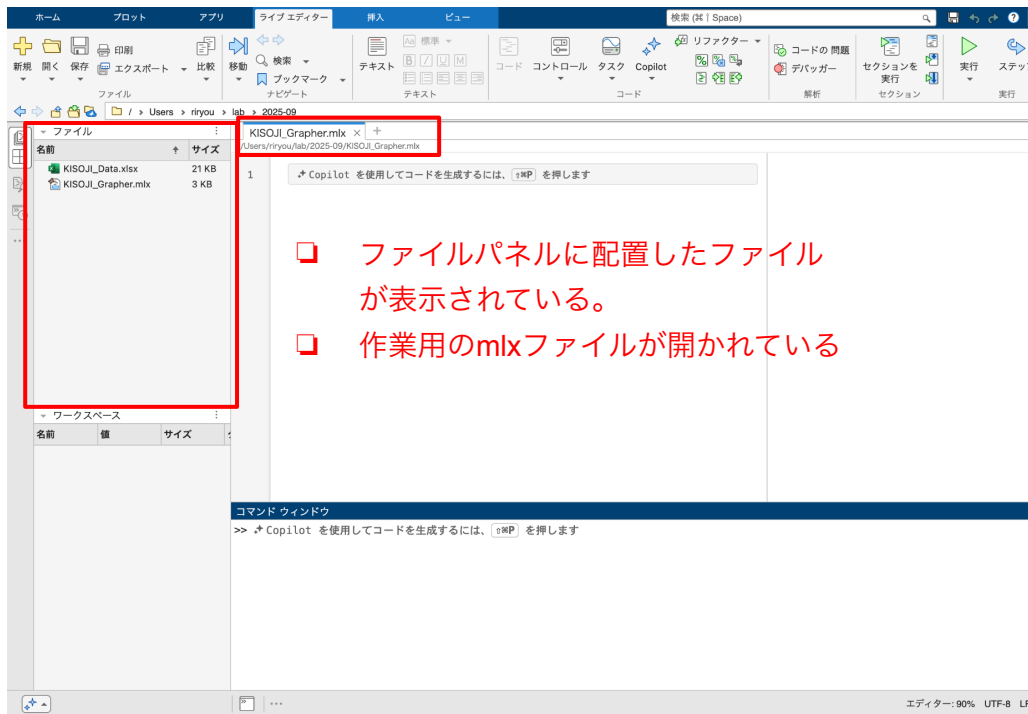
macOS Sequoia

バージョン 15.6.1 (24G90)



準備

1. 作業用フォルダを作り、その中にデータを記録したxlsxファイルとmatlabのライブスクリプトファイル(以降mlxファイル)を作成して配置する。
2. MATLABを起動し、該当のフォルダとmlxファイルが開いているようにする。



データの取り込み

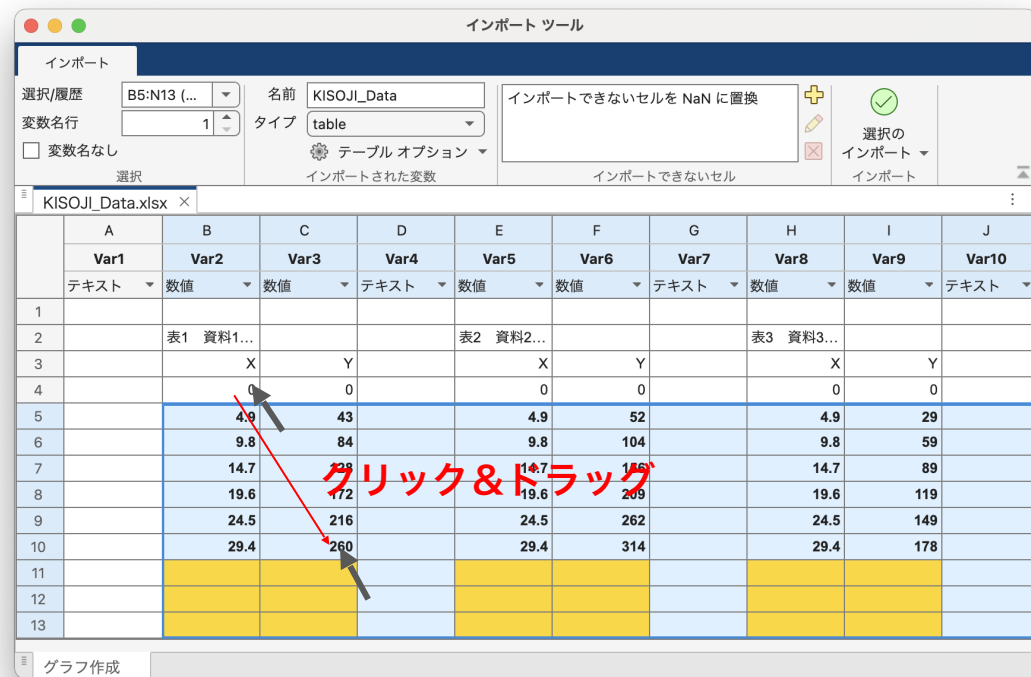
データのファイルをダブルクリックして、インポートツールをひらく。

The screenshot shows the MATLAB R2025a interface. The 'File' browser on the left has 'KISOJI_Data.xlsx' selected. The 'Import Tool' window is open, showing the 'Import' tab. The 'Select/History' section shows 'B5:N13 (...)' selected. The 'Variable Name' is 'KISOJI_Data', and the 'Type' is 'table'. The 'Import' button is highlighted. The 'Imported Variables' section shows 'KISOJI_Data.xlsx' imported as a table. The 'Imported Variables' section shows a table with 7 columns (Var1 to Var7) and 12 rows of data.

	A	B	C	D	E	F	G
	Var1	Var2	Var3	Var4	Var5	Var6	Var7
テキスト	数値	数値	テキスト	数値	数値	テキスト	
1							
2		表1 資料1...			表2 資料2...		
3		X	Y		X	Y	
4		0	0		0	0	
5		4.9	43		4.9	52	
6		9.8	84		9.8	104	
7		14.7	128		14.7	156	
8		19.6	172		19.6	209	
9		24.5	216		24.5	262	
10		29.4	260		29.4	314	
11							
12							

データの取り込み

1. 今回は、欲しいプロットについて個別にデータをインポートする。まず、表1のデータを取り込む。
2. エクセルの要領でカーソルをドラッグして表1のデータを選択する。（次のページへ）



6.上の三つの設定ができたなら、次のページへ

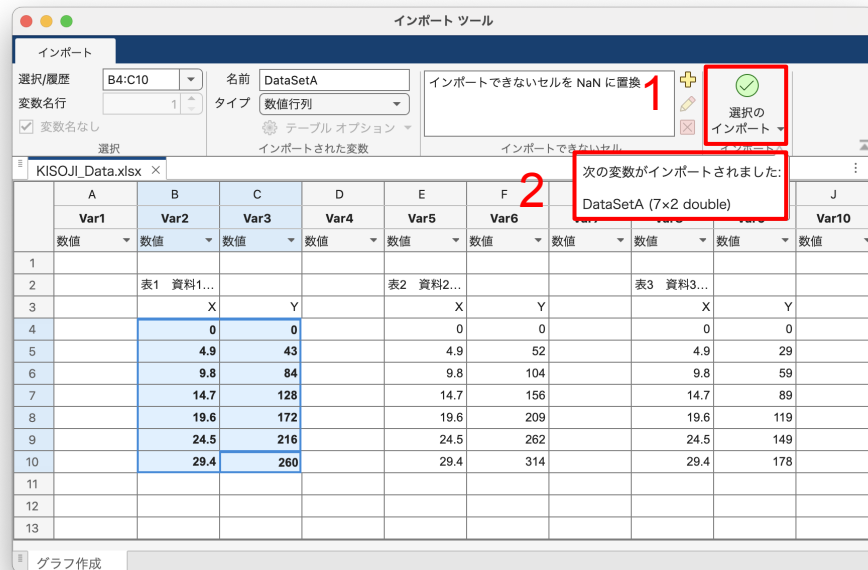


データの取り込み

1. 前のスライドでの3つの設定をしたら「選択のインポート」をクリックする。

数字が全角になっていたり、空欄があると上手くインポートできないことがあるので注意。

2. 上手くインポートできたら、メッセージがでる。



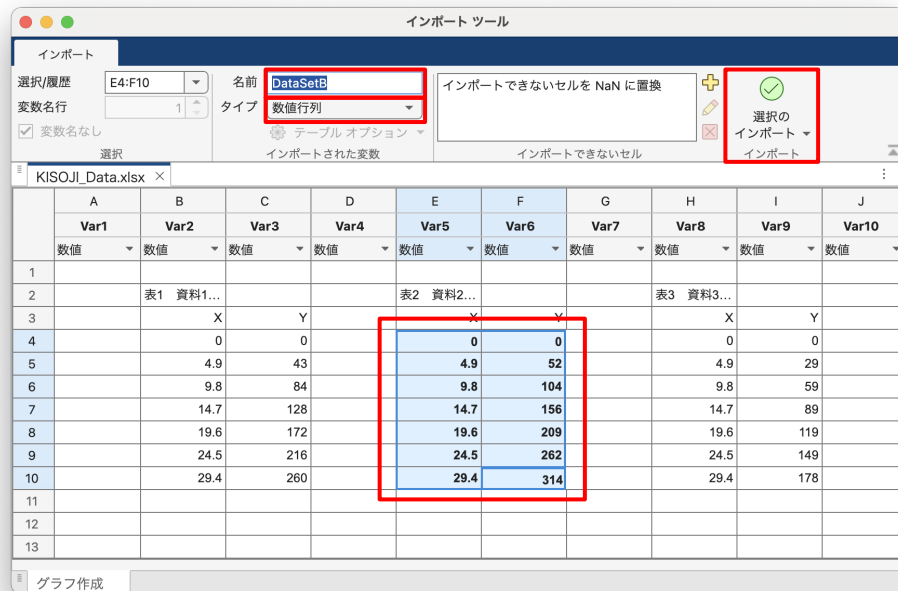
データの取り込み

同様にして表2、表3のデータもインポートする。

今回はデータを格納した変数の名前を

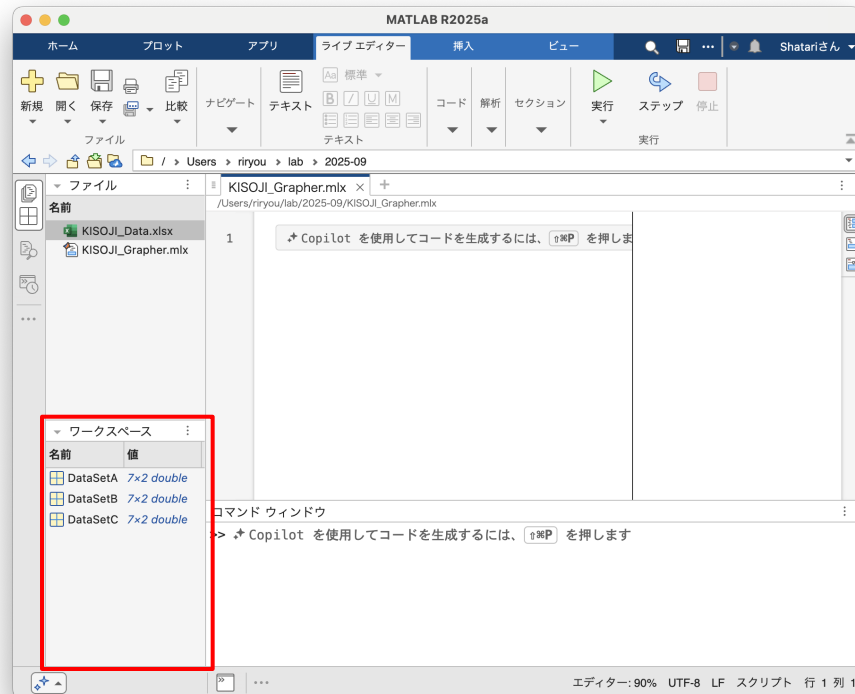
- DataSetA
- DataSetB
- DataSetC

とした。



データの取り込み

データをちゃんと取り込めていたら
ワークスペースのところに、ロードでき
たデータの変数名や型が表示される。
(ダブルクリックして編集できる)



一番簡単なプロット (plot)

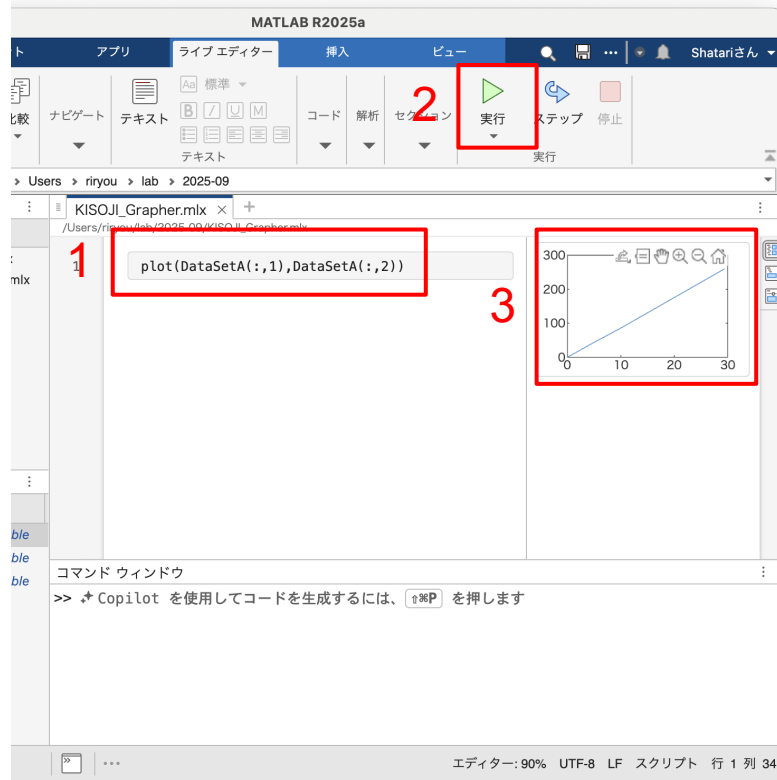
データをプロットする。

プロットする時には色や線の形式、表示する図など様々な事項を指定できるが、一番シンプルな方法では右のコードのようになる。

`plot(DataSetA(:,1),DataSetA(:,2))`
を入力。

上のリボンにある「実行」をクリック

ライブスクリプトの右側に出力が表示される



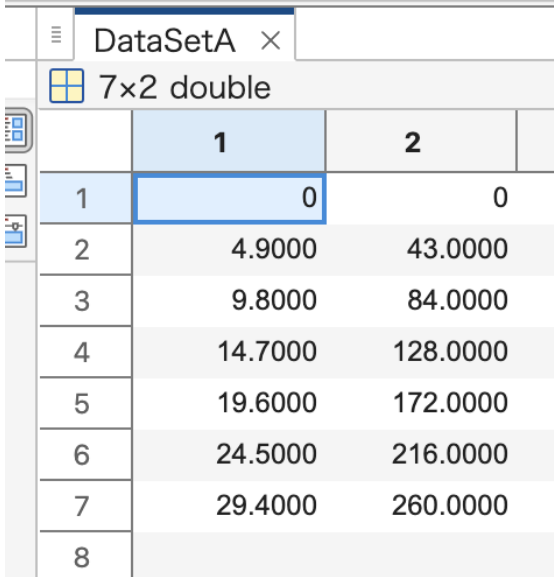
一番簡単なプロット（解説1）

plot関数は次のようにデータを受け取る

plot(x軸のデータ,y軸のデータ)

DataSetAのx軸のデータはDataSetA(:,1)で取り出せるので、これをplot関数の1個目の引数に渡し、y軸のデータDataSetA(:,2)を2つ目の引数に渡すことで、DataSetAのデータをプロットできる。

DataSetA(:,1),DataSetA(:,2)は右図のデータの1列目,2列目のデータに対応しており、これはMATLABの仕様によるものである。



DataSetA ×		
7x2 double		
	1	2
1	0	0
2	4.9000	43.0000
3	9.8000	84.0000
4	14.7000	128.0000
5	19.6000	172.0000
6	24.5000	216.0000
7	29.4000	260.0000
8		

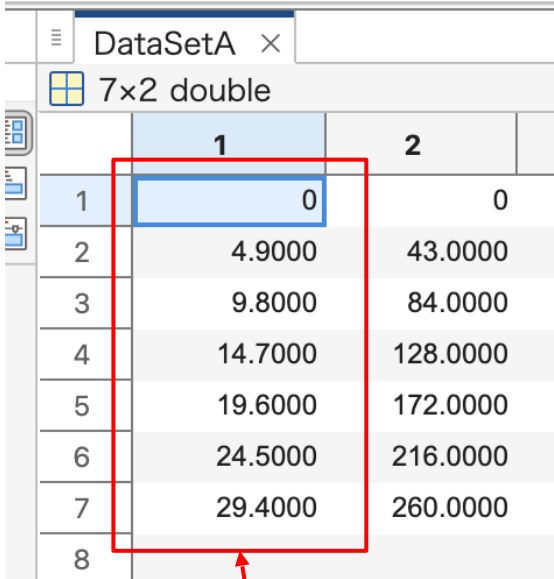
一番簡単なプロット（解説2）

今回のデータは行列形式で読み込まれている。行列形式のデータから n 行目 m 列目にあるデータを欲しい時は (n, m) を変数名の後に付ける。（インデックス）

例えば DataSetA の 2 行 1 列目のデータを取り出したい時は、
`DataSetA(2,1)` とする。

細かい説明を省くが、インデックスに「:」を指定すると、「全ての行（列）」を指定できる。

`DataSetA(:,1)` は「行列 DataSetA のうち、1 列目で、かつ全ての行を取り出したデータ」を得ることができる。



	1	2
1	0	0
2	4.9000	43.0000
3	9.8000	84.0000
4	14.7000	128.0000
5	19.6000	172.0000
6	24.5000	216.0000
7	29.4000	260.0000
8		

X軸のデータと対応する

`DataSetA( : , 1 )`

全
て
の
行

1
列
目

グラフの重ね書き (hold)

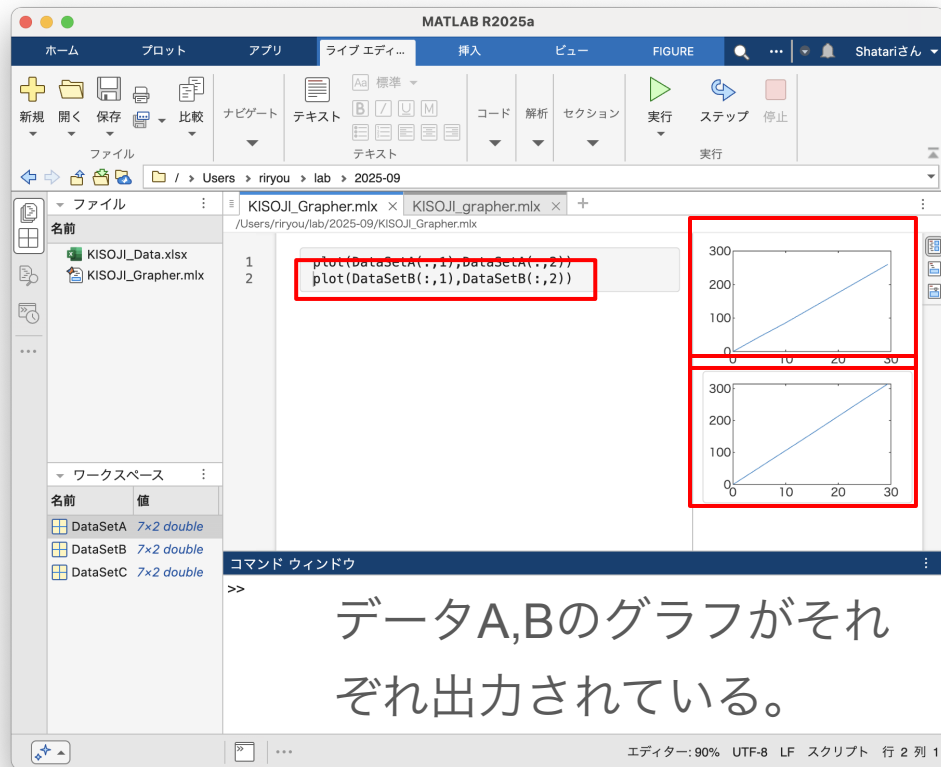
では先ほどのコードに加えて、次の行に DataSetB をプロットするコードを書くとうなるか。

この場合グラフは各プロットにそれぞれ新たに作成され、2つの別々のグラフが出力される。

複数のデータを一枚のグラフに同時にプロットするには、あらかじめ重ね書きを許可する必要がある

hold on

これを一つ目のプロットの後に加える

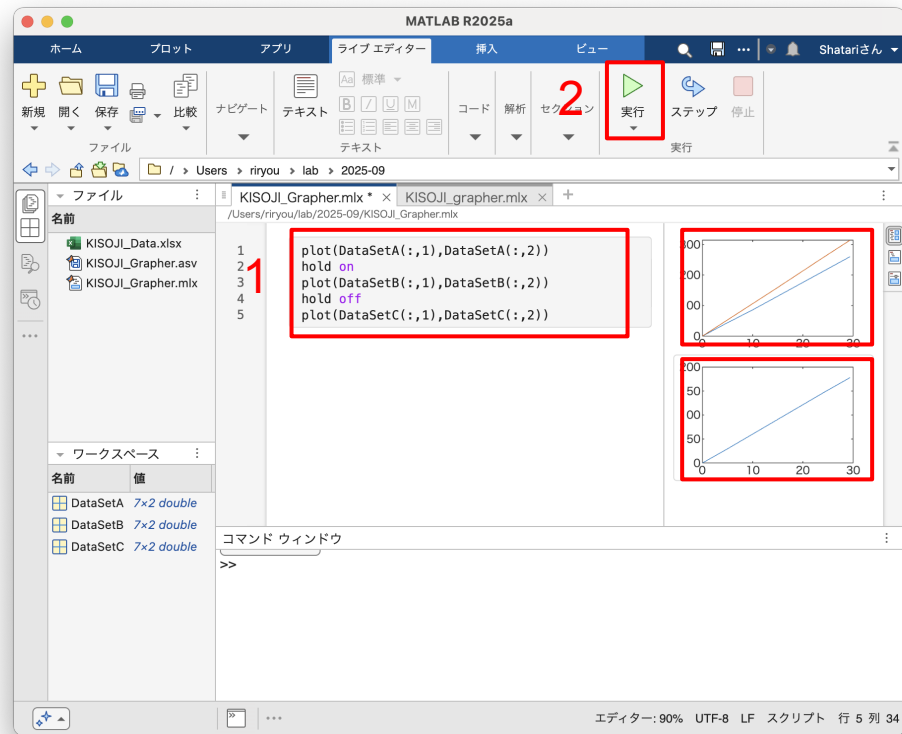


グラフの重ね書き (hold)

一度目のplotの後にhold onを書くことで、以降hold offをするまでplotした結果がhold onしたplotに重ね書きされる。

右のコードではA,B,Cをプロットするコードの間にhold on,hold offを挟んだものである。

A,Bは同じグラフにプロットされた一方、Cは別のグラフにプロットされている。

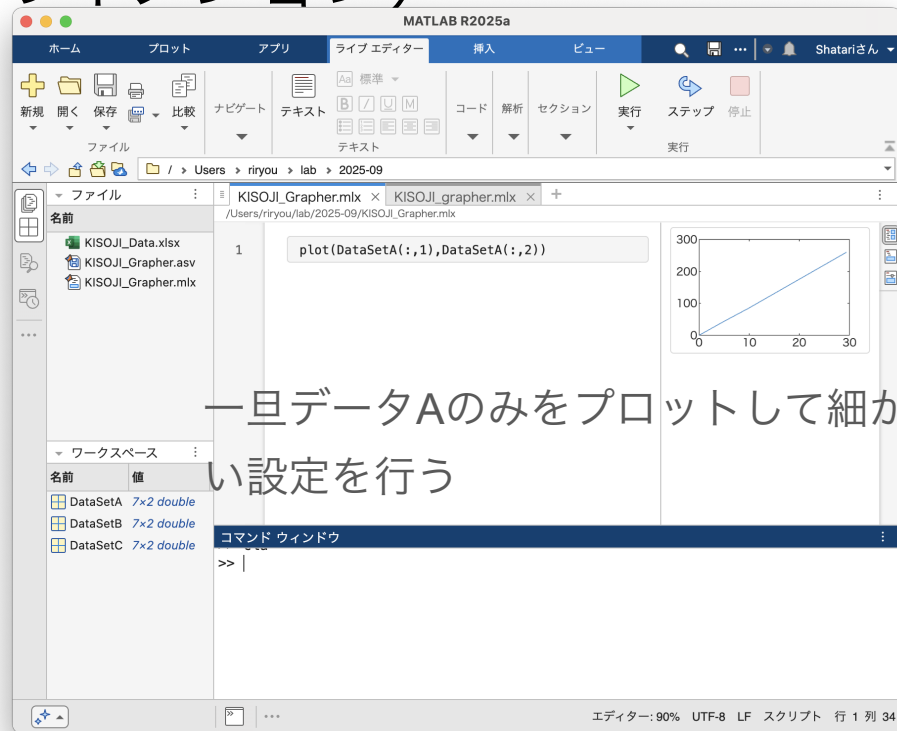


プロットのスタイル (plotのオプション)

指定のフォーマットでは得られたデータの
プロットは $\odot$, $\triangle$, $\square$
に指定されているので、このスタイルの
マーカーを配置していく。

簡単のために一度、DataSetAのみをプ
ロットする。

またMATLABには同じ挙動をする様々な
オプションの指定方法が用意されてい
る。今回説明するのはほんの一例であ
る。



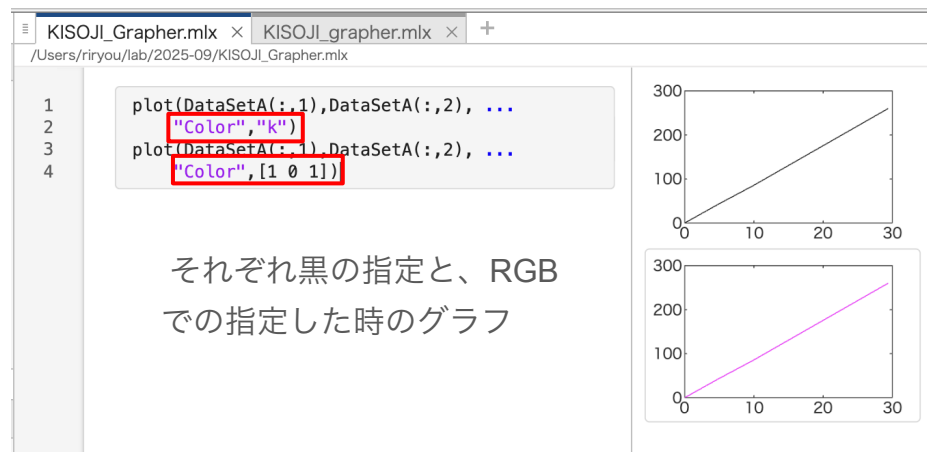
プロットの色 (colorプロパティ)

plot(x軸のデータ,y軸のデータ)に加えてさらに、種々のオプションを指定できる。プロットを黒にするには

```
plot(x,y, "Color", "k")
```

のように"k"を指定する (1,2行目)

"r"とすれば赤になり、さらにRGBやカラーコードなども柔軟に設定できる。詳しくはMathworksのドキュメントを見てほしい。



プロットのマーカー (Markerプロパティ)

次にMarkerプロパティの設定をする。先ほどと同様"Marker"とした後、次の引数に"o"を渡す。

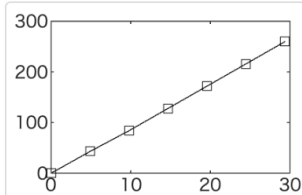
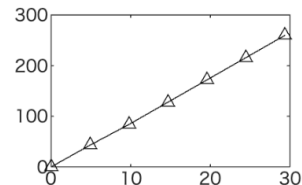
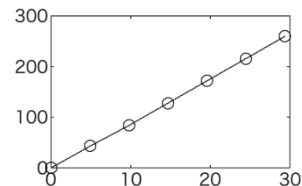
"^"ならマーカーが三角に、"square"なら四角に、さらに"."ならドットになる。

他にも色々あるのでドキュメントをみるべし。

KISOJI_Grapher.mlx * ×
/Users/riryoyou/lab/2025-09/KISOJI_Grapher.mlx

```
1 plot(DataSetA(:,1),DataSetA(:,2), ...  
2     "Color","k", ...  
3     "Marker","o")  
4 plot(DataSetA(:,1),DataSetA(:,2), ...  
5     "Color","k", ...  
6     "Marker","^")  
7 plot(DataSetA(:,1),DataSetA(:,2), ...  
8     "Color","k", ...  
9     "Marker","square")
```

上から、丸、三角、四角をマーカーに指定したもの

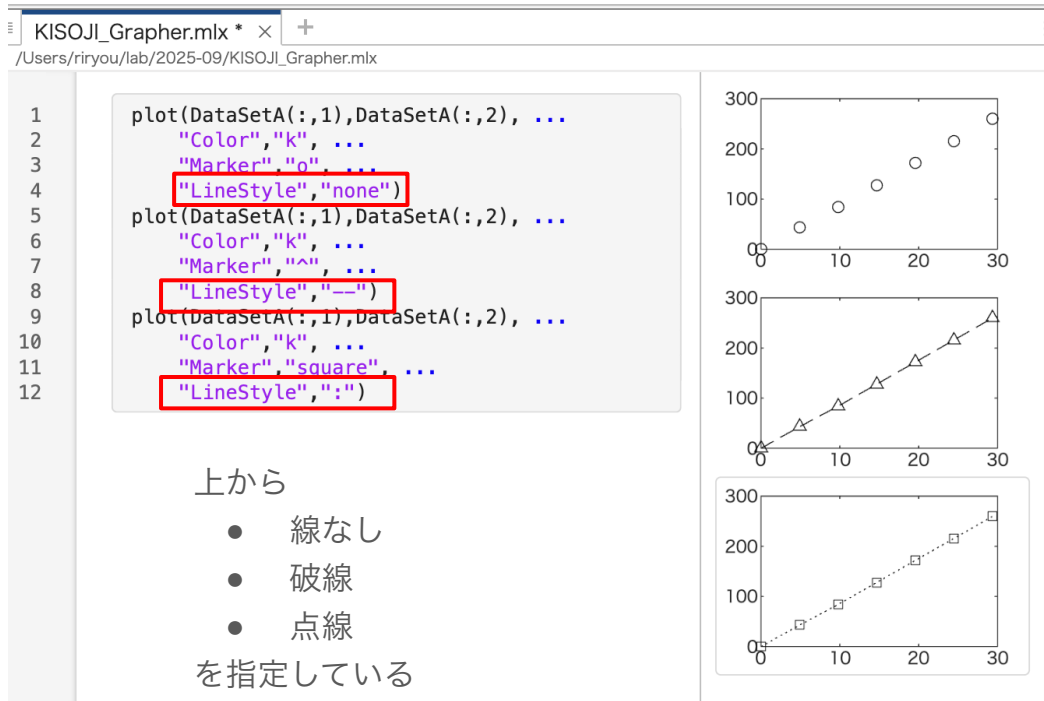


プロットの線の体裁 (LineStyleプロパティ)

グラフに表示されている線は近似直線ではなく、ただ近い点同士を結んだものである。これを消すには

“LineStyle”プロパティに“none”を指定する。

また線を点線にするには“:”、破線にするには“--”など、これも色々あるので公式ドキュメントをみると良い



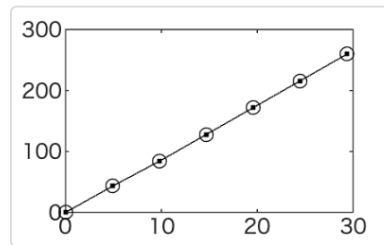
中央に点をつける方法 (Markerプロパティ)

一旦ここからは一つのデータについてのみグラフをプロットし、作図する

マーカーで丸や三角の中心に点をつけたフォーマットのものはMATLABにはない。そこで、同じデータを2回描画し、一方はマーカーを図方は点にして重ね描く。

hold onにして同じデータのプロットをkとなるマーカーで重ね描く

```
1 plot(DataSetA(:,1),DataSetA(:,2), ...  
2      "color","k", ...  
3      "Marker","o", ...異なるマーカー  
4      "LineStyle","_")  
5 hold on  
6 plot(DataSetA(:,1),DataSetA(:,2), ...  
7      "Color","k", ...  
8      "Marker",".", ...  
9      "LineStyle","none")  
10 hold off
```



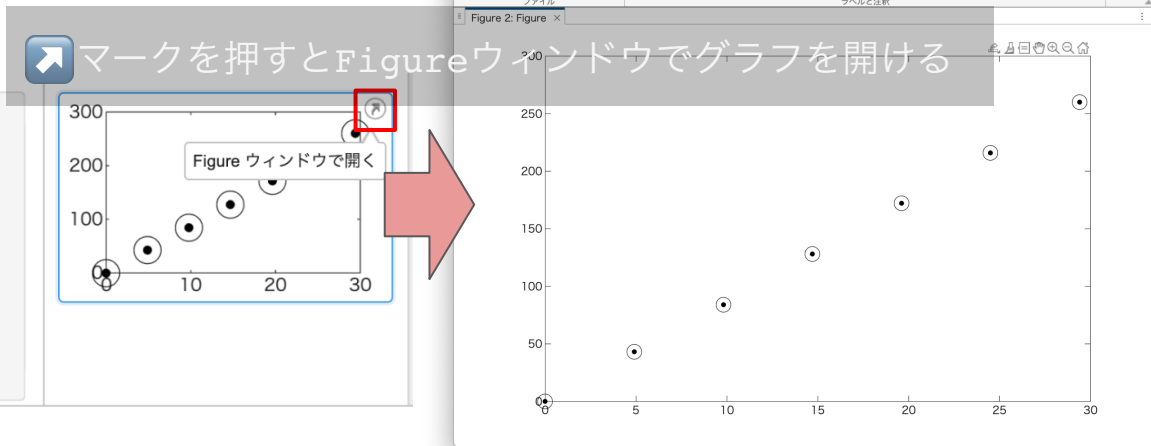
マーカーのサイズ MarkerSizeプロパティ

“MarkerSize”に数字を指定してやることでマーカーのサイズを指定できる。

この数値はダブルクォートなしで渡すので注意

右側の出力画面で見ている時と、Figureウィンドウで見た時の大きさが異なるので適宜確認して調整する。

```
19 plot(DataSetA(:,1),DataSetA(:,2), ...  
20      "Color","k", ...  
21      "Marker","o", ...  
22      "MarkerSize", 15, ...  
23      "LineStyle","none")  
24 hold on  
25 plot(DataSetA(:,1),DataSetA(:,2), ...  
26      "Color","k", ...  
27      "Marker","o", ...  
28      "MarkerSize", 15, ...  
29      "LineStyle","none")  
30 hold off
```



マーカの塗りつぶし (MarkerFaceColorプロパティ)

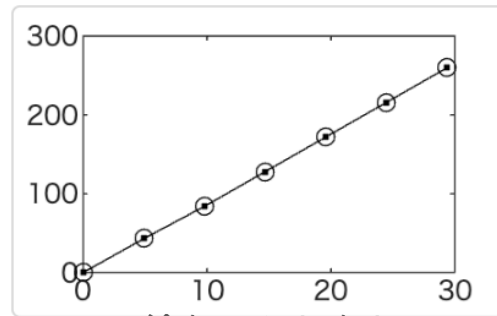
“o”の方はこのままだとマーカの内部が透明になっており、裏側の線が透過して見えてしまう。

体裁では貫通しないように指定があるので内部の塗りつぶしの色に白を指定する。

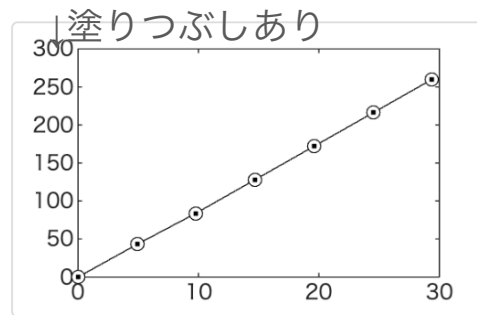
“MarkerFaceColor”に[1 1 1]を指定する。

ダブルクォートの有無に注意

```
1 plot(DataSetA(:,1),DataSetA(:,2), ...  
2      "Color","k", ...  
3      "Marker","o", ...  
4      "MarkerFaceColor",[1 1 1], ...|  
5      "LineStyle","-")  
6 hold on  
7 plot(DataSetA(:,1),DataSetA(:,2), ...  
8      "Color","k", ...  
9      "Marker",".", ...  
10     "LineStyle","none")  
11 hold off
```



↑塗りつぶしなし



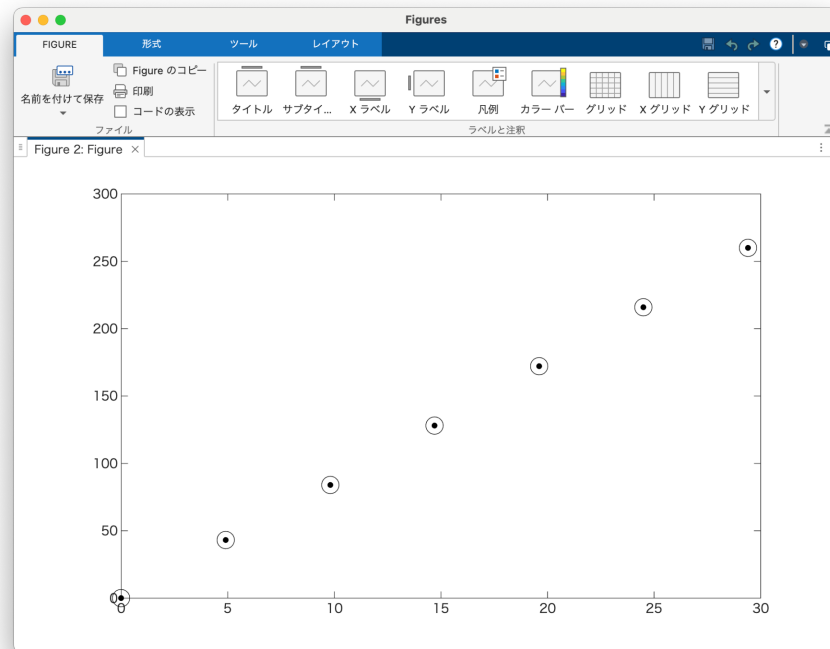
↑塗りつぶしあり

現状のグラフ

点同士を線で結ばない

図形+中心に点を置いたマーカ
でプロット

マーカサイズを適切なサイズに
設定



近似直線の作成

近似曲線の計算は`polyfit`, `polyval`を用いて簡単に計算できる。

`polyfit`は与えられたデータを元に多項式近似を行い、得られた多項式の係数列を返す。また`polyval`は多項式の係数列と`x`の値を元に、対応する`y`の値を計算する。

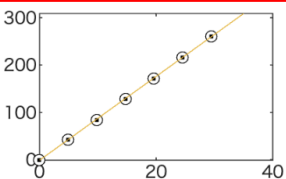
```
pol_f = polyfit(DataSetA(:,1),DataSetA(:,2),1)
x = [0 35]
y_p = polyval(pol_f,x)
```

```
plot(DataSetA(:,1),DataSetA(:,2), ...
     "Color","k", ...
     "Marker","o", ...
     "MarkerFaceColor",[1 1 1], ...
     "LineStyle","none")
hold on
plot(DataSetA(:,1),DataSetA(:,2), ...
     "Color","k", ...
     "Marker",".", ...
     "LineStyle","none")
plot(x,y_p)
hold off
```

```
pol_f = 1x2
      8.8484  -1.0714
```

```
x = 1x2
      0      35
```

```
y_p = 1x2
     -1.0714  308.6224
```



左のように近似曲線を書き加えたい

polyfit関数

<https://jp.mathworks.com/help/matlab/ref/polyfit.html>

`polyfit(x軸のデータ, y軸のデータ, 近似する次数n)`

とすることでx, yのデータを元にn次近似したn+1個の数字を得られる。

今回は一次近似なのでn=1として、2個の係数を得る。

入力と出力は以下の図のようになる。

1	<code>pol_f = polyfit(DataSetA(:,1),DataSetA(:,2),1)</code>	<code>pol_f = 1x2 8.8484 -1.0714</code>
---	---	---

↑x軸のデータ ↑y軸のデータ ↑1次近似

近似直線は

$y = 8.8484 \cdot x - 1.0714$
と算出された

↑DataSetAのデータを渡して得られた係数

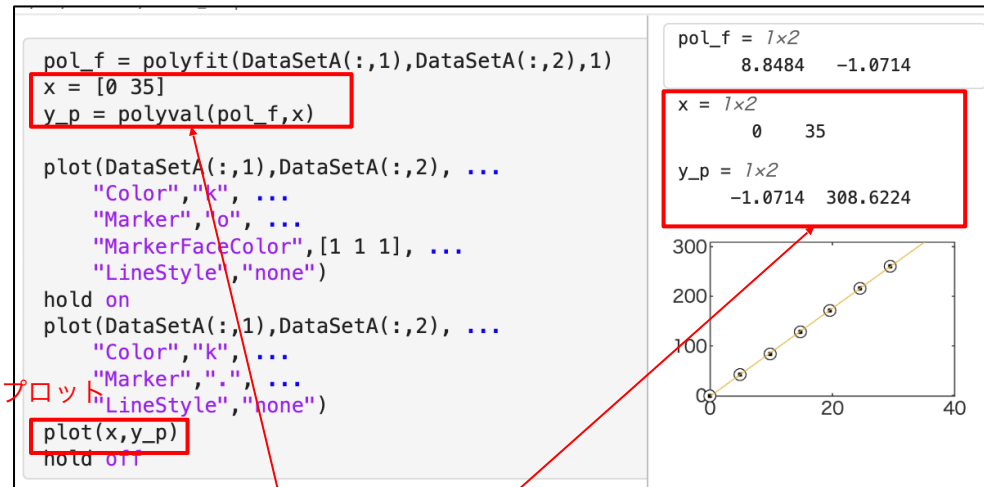
列[8.8484 -1.0714]を変数pol_fに格納

polyval関数

`polyval(係数列,x)`

前のスライドで係数列[8.8484 -1.0714]を得た、これをグラフにプロットするには
 $y=8.8484x-1.0714$ を満たす点をもつデータを作り、これをプロットすることで近似直線を描画する。

例えば右の図ではxに[0 35]を与え、対応するy座標[-1.0714 308.6224]を得ている。この2点(0,-1.07),(35,308.62)をプロットし、“LineStyle”を指定して線で結ぶことで近似直線を描画できる。



近似直線描画用データの計算

<https://jp.mathworks.com/help/matlab/ref/polyval.html>

近似直線の描画

前のスライドで得たデータをplotすることで近似直線を描画する。

近似直線のデータは右のコードにおいては x_p, y_p に格納してある。

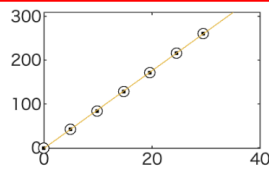
`plot(x_p, y_p)`とすることで描画できるが、マーカーを貫通してしまっている
のでこれを修正する。

```
1 pol_f = polyfit(DataSetA(:,1),DataSetA(:,2),1)
2 x_p = [0 35]
3 y_p = polyval(pol_f,x_p)
4
5 plot(DataSetA(:,1),DataSetA(:,2), ...
6      "Color","k", ...
7      "Marker","o", ...
8      "MarkerFaceColor",[1 1 1], ...
9      "LineStyle","none")
10 hold on
11 plot(DataSetA(:,1),DataSetA(:,2), ...
12      "Color","k", ...
13      "Marker",".", ...
14      "LineStyle","none")
15 plot(x_p,y_p)
16 hold off
```

プロット

```
pol_T = 1x2
      8.8484 -1.0714
```

```
x_p = 1x2
      0      35
y_p = 1x2
     -1.0714 308.6224
```



近似直線の描画2

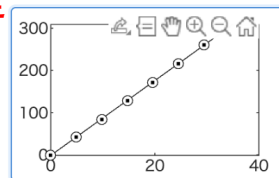
plotの重ね描くと、後から書いたものがグラフの手前側に上書きされる。このため近似直線を一番最初にプロットするようにすることで、マーカーの貫通を防ぐことができる。

また"LineStyle"と"Color"も指定しておくこと。

```
1 pol_f = polyfit(DataSetA(:,1),DataSetA(:,2),1)
2 x_p = [0 35]
3 y_p = polyval(pol_f,x_p)
4
5 plot(x_p,y_p, ...
6      "Color","k", ...
7      "LineStyle","-")
8
9 hold on
10 plot(DataSetA(:,1),DataSetA(:,2), ...
11      "Color","k", ...
12      "Marker","o", ...
13      "MarkerFaceColor",[1 1 1], ...
14      "LineStyle","none")
15 plot(DataSetA(:,1),DataSetA(:,2), ...
16      "Color","k", ...
17      "Marker",".", ...
18      "LineStyle","none")
19 hold off
```

最初に描画
色とスタイルを指定

```
pol_f = 1x2
      8.8484 -1.0714
x_p = 1x2
      0      35
y_p = 1x2
     -1.0714 308.6224
```

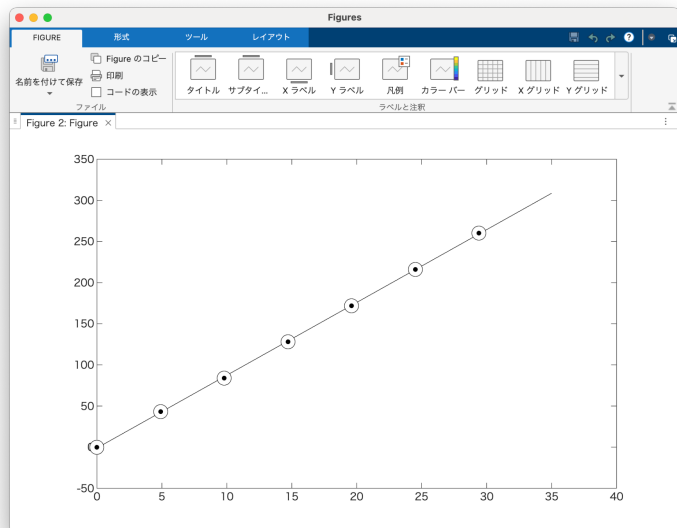


現状のグラフ

マーカーの体裁・サイズ

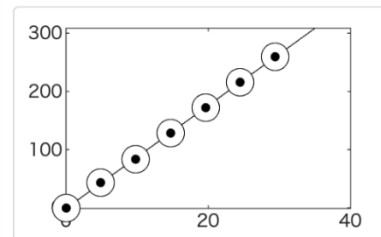
近似直線の計算、描画

近似直線がマーカーを貫通していない。



```
1 %近似直線の算出
2 pol_f = polyfit(DataSetA(:,1),DataSetA(:,2),1)%係数の算出
3 x_p = [0 35]%近似直線の表示範囲の設定
4 y_p = polyval(pol_f,x_p)%近似直線描画用データの作成
5 %近似直線のプロット
6 plot(x_p,y_p, ...
7      "Color","k", ...
8      "LineStyle","-")
9 hold on % 重ね描きをonにする
10 % 図形マーカーでデータをプロット
11 plot(DataSetA(:,1),DataSetA(:,2), ...
12      "Color","k", ...%色の設定
13      "Marker","o", ...%マーカーの種類
14      "MarkerSize",15, ...%マーカーの大きさ
15      "MarkerFaceColor",[1 1 1], ...%マーカー内部の塗りつぶし
16      "LineStyle","none")%線を非表示にする
17 % ドットマーカーでデータをプロット
18 plot(DataSetA(:,1),DataSetA(:,2), ...
19      "Color","k", ...
20      "Marker",".", ...
21      "MarkerSize",15, ...
22      "LineStyle","none")
23 hold off % 重ね描きをoffにする
```

```
pol_f = 1x2
      8.8484 -1.0714
x_p = 1x2
      0 35
y_p = 1x2
     -1.0714 308.6224
```



↑現状のコード+コメント

←Figureウィンドウで確認したグラフ

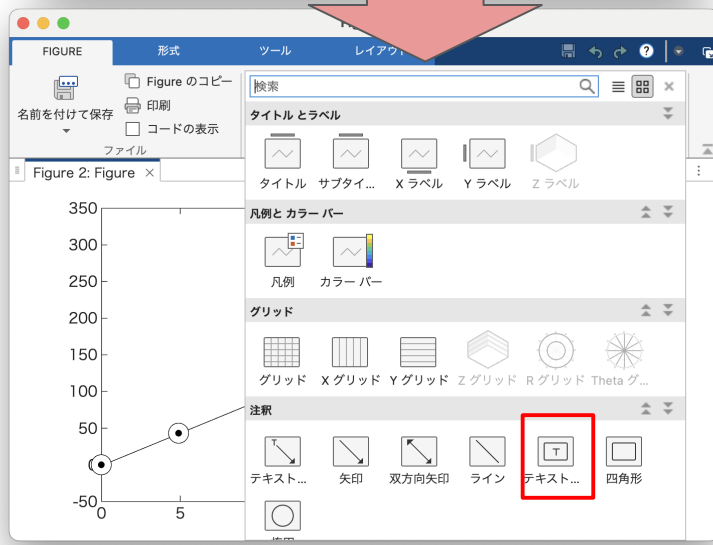
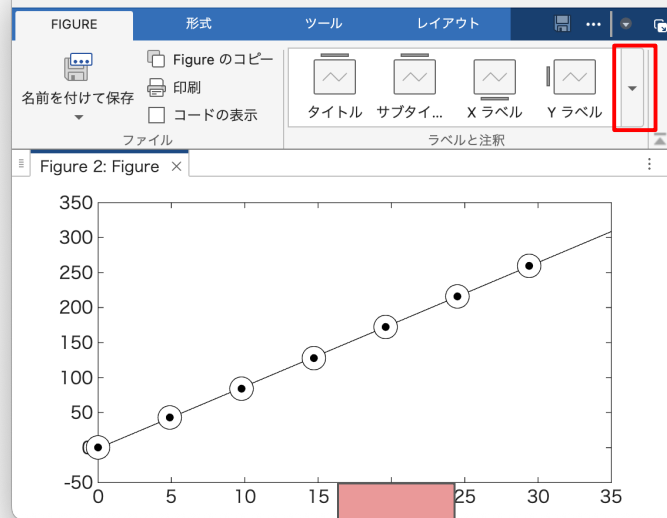
近似式の表示

近似直線は既に算出し、係数がわかっていると思う

figureウィンドウでテキストを配置することで近似式を表示できる。

figureウィンドウを表示して

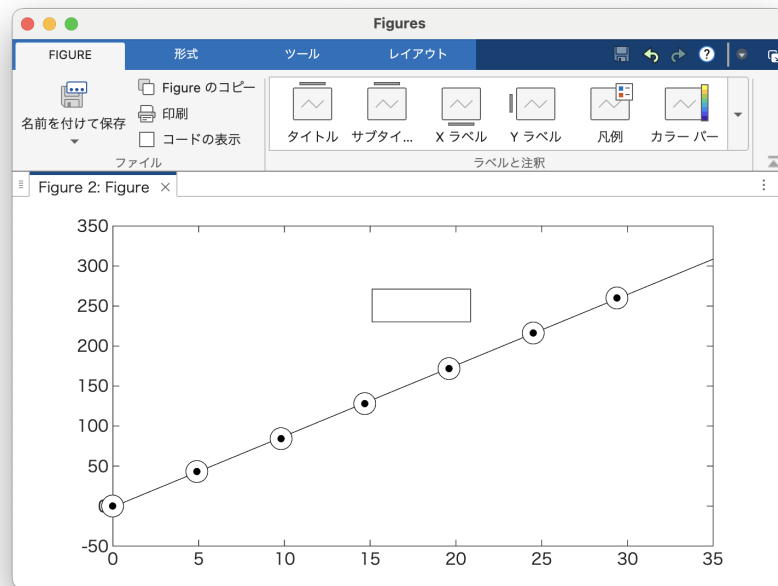
「ラベルと注釈」のプルダウンを開き、「注釈」の「テキスト」をクリックする。



近似式の表示

「テキスト」をクリックした後、
グラフの枠内にマウスカーソルを
持ってくるとカーソルが「+」記
号になるのでその状態で、適当な
場所をクリック。

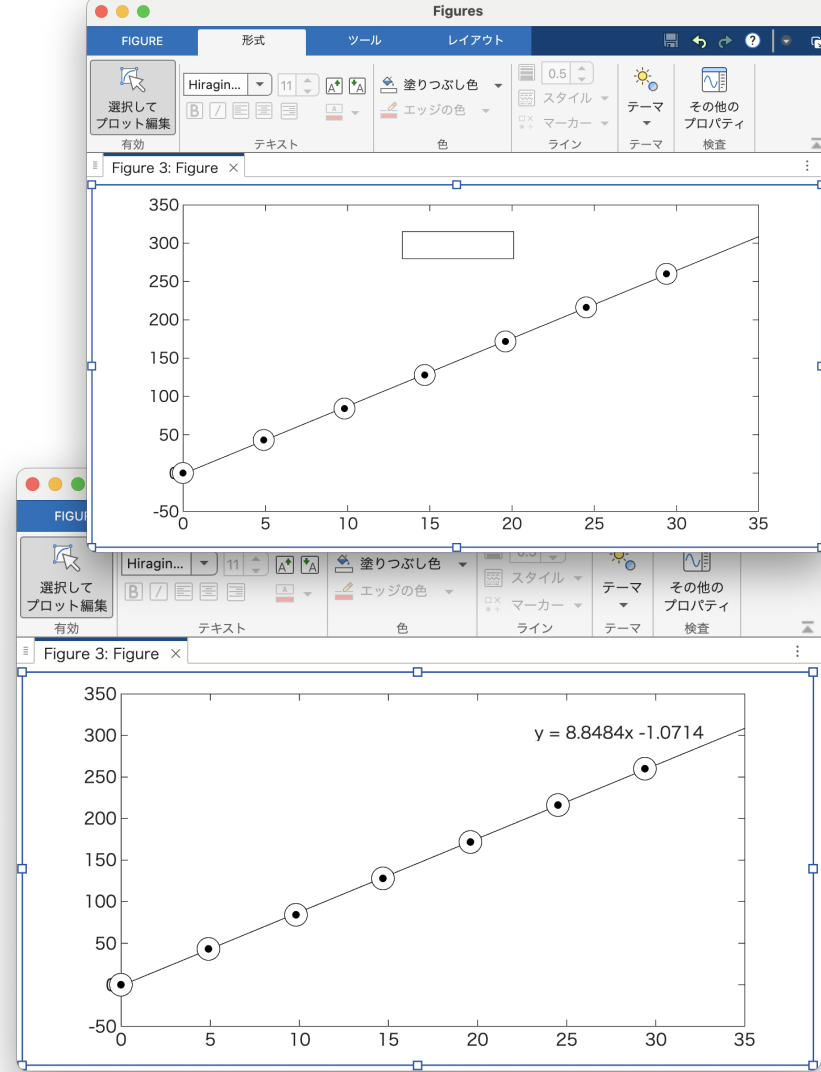
そうすると、四角い枠が置かれる
ので、これを編集していく。



近似式の表示

上部のタブから「形式」を選択し、「選択してプロット編集」をクリックする。

そうするとパワーポイントの要領で編集を行うことができる。

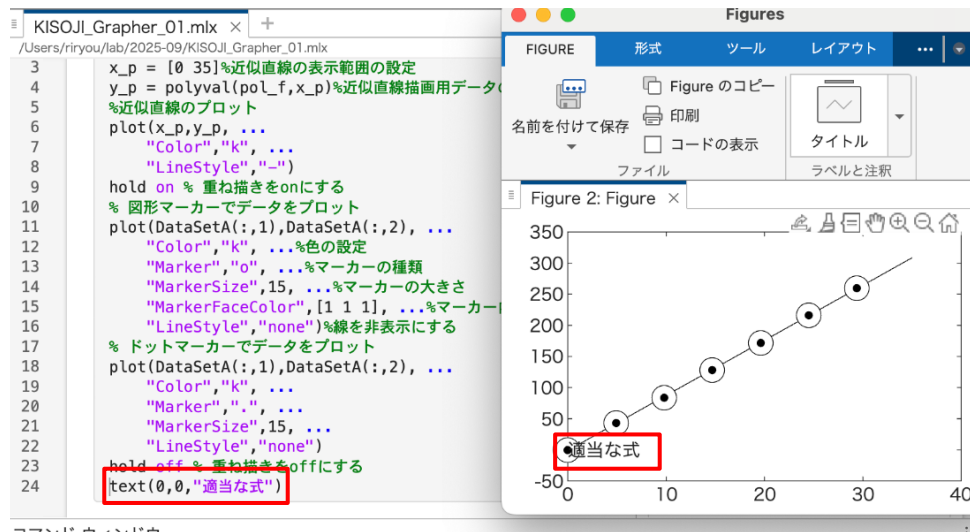


(Optional) 近似式の表示 (text関数)

以上で近似式の表示はできるが、式を入力が手動である。このため何らかの要因で、近似式の係数が変化した時には手動で修正しなければならない。

そこで`text(0,0,"適当な式")`という命令を適当な行につけておくことで、座標の(0,0)の位置に文字列を表示できる。

さらに次のスライドのようにすることで、計算結果を自動的に反映した式を書くようにできる。



(Optional) 近似式の表示 (text関数)

表示位置の座標 文字列に変換 1次の係数 文字列に変換 0次の係数

```
text(0,0,"y = " + string(pol_f(1)) + "x" + string(pol_f(2)))
```

表示内容

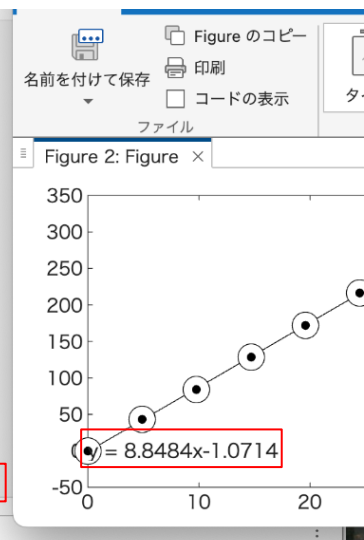
とすると、算出されて係数をグラフに書くことができる。

詳しくは省くが“y =”と“x”という文字列と各係数を結合することで表示している。数値の加算と文字列の結合は区別されるためstring(係数)とすることで数値を文字列に変換して結合している。

また座標はここに設定するよりも(0,0)にまとめて出力して、figureウィンドウから開いて直すのが楽なので今回はそうする。

```
3 x_p = [0 35]%近似直線の表示範囲の設定
4 y_p = polyval(pol_f,x_p)%近似直線描画用データの作成
5 %近似直線のプロット
6 plot(x_p,y_p,...
7      "Color","k",...
8      "LineStyle","-")
9 hold on % 重ね描きをonにする
10 % 図形マーカーでデータをプロット
11 plot(DataSetA(:,1),DataSetA(:,2),...
12      "Color","k",...%色の設定
13      "Marker","o",...%マーカーの種類
14      "MarkerSize",15,...%マーカーの大きさ
15      "MarkerFaceColor",[1 1 1],...%マーカー内部の塗りつぶし
16      "LineStyle","none")%線を非表示にする
17 % ドットマーカーでデータをプロット
18 plot(DataSetA(:,1),DataSetA(:,2),...
19      "Color","k",...
20      "Marker",".",...
21      "MarkerSize",15,...
22      "LineStyle","none")
23 hold off % 重ね描きをoffにする
24 text(0,0,"y = " + string(pol_f(1)) + "x" + string(pol_f(2)))
```

1マンドウィンドウ



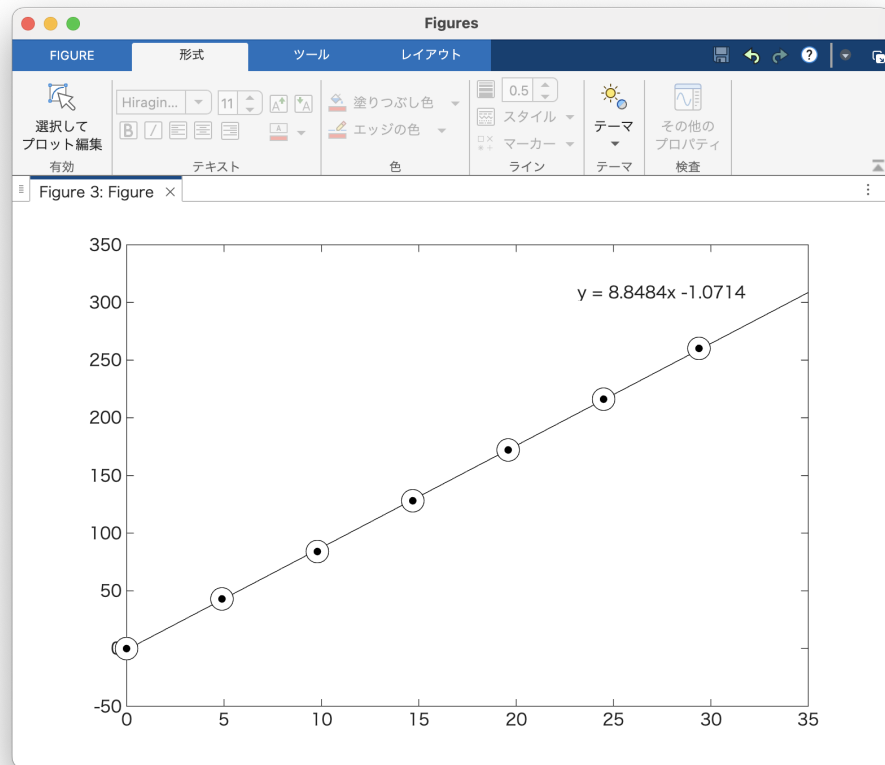
現状のグラフ

マーカーの体裁・サイズ

近似直線の計算、描画

近似直線がマーカーを貫通していない

近似直線の式を表示している



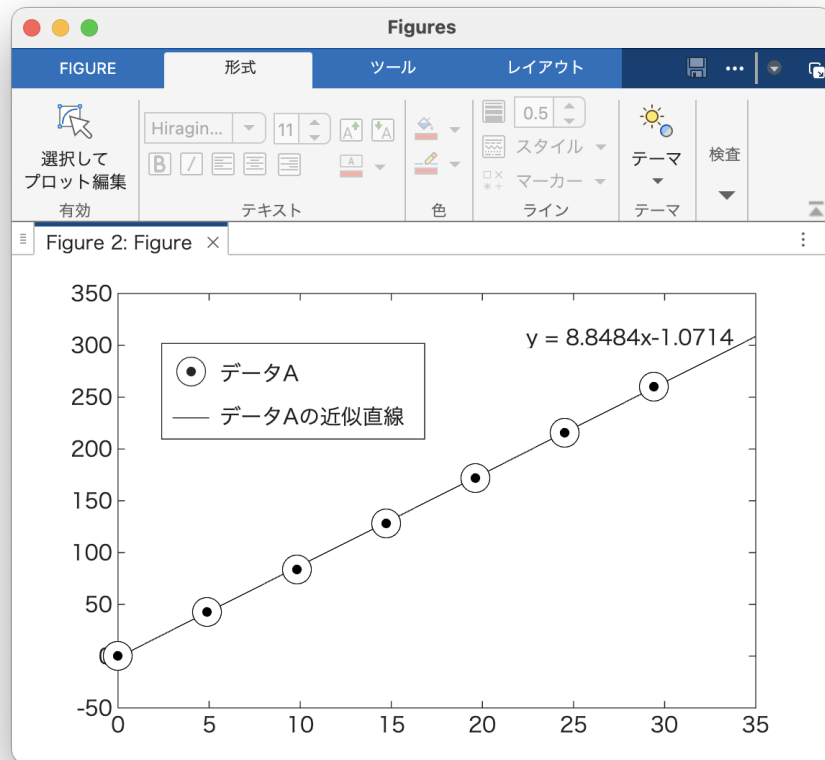
凡例の表示

凡例の表示は本来は簡単である。

本来はlegend関数を使うと凡例が表示されるのだが、この関数を使うと $\odot$, $\triangle$, $\square$ のようなオリジナルのマーカーを表示する方法がない（2025年時点）

なので手順としては、まず凡例の説明の部分だけ、近似式と同じ手順で表示する。

そして説明に対応するマーカーは、説明の隣に点をプロットすることで実現する。



凡例をこんな感じに表示する

凡例の表示（説明の表示）

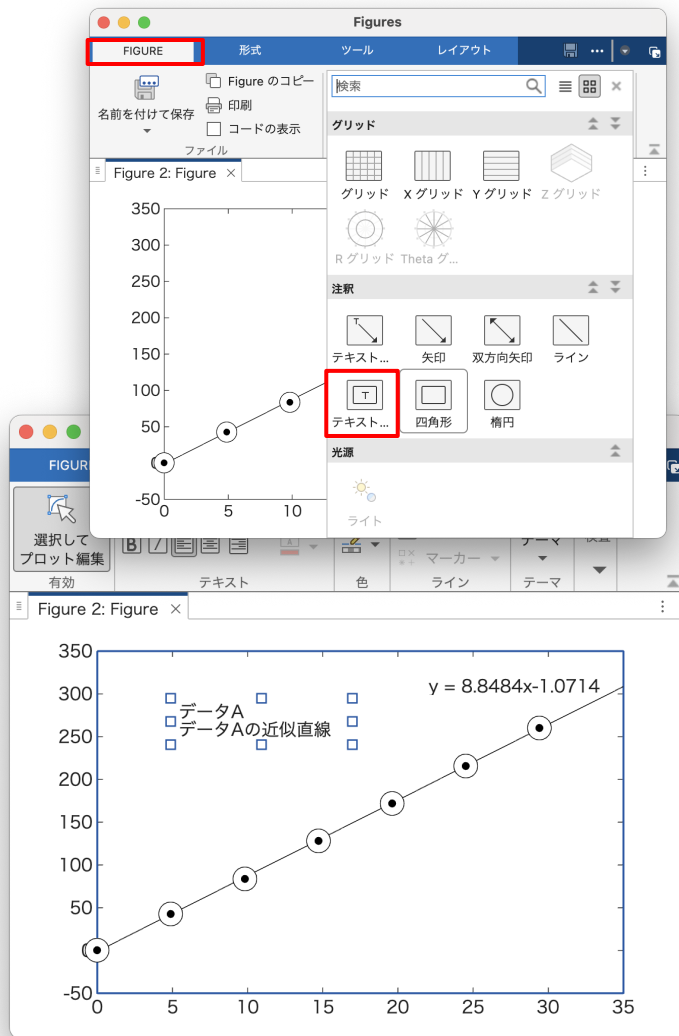
前スライドの「データA, データAの近似直線」の部分の表示を行う

これは近似式の表示と同じ手順で行える

上部タブの「FIGURE」、「ラベルと注釈」のプルダウンを開いて、「テキストをクリック」。さらにグラフの枠内をクリックして、凡例の説明を入力する。

Shift+Enterで内部での改行ができる

エッジの色をなしにする

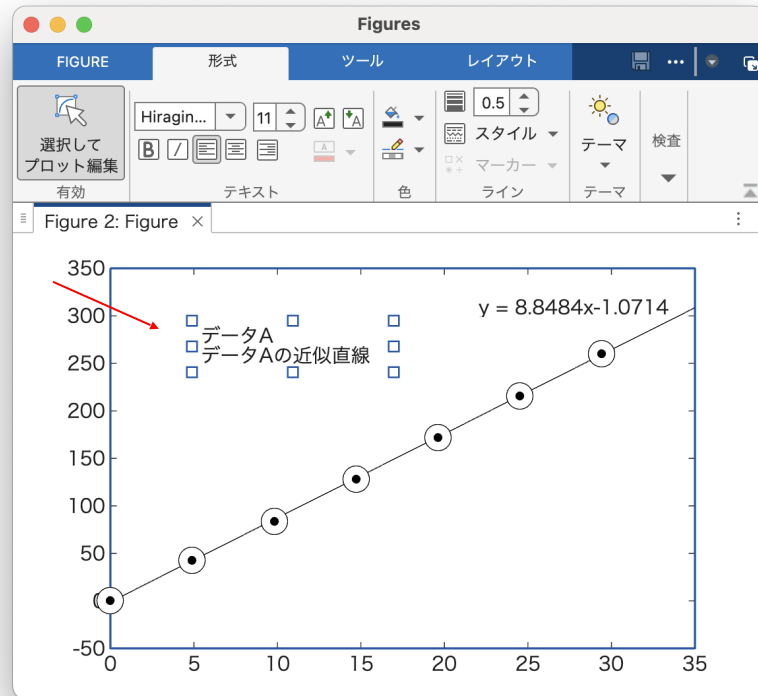


凡例の表示 (マーカの表示)

次に、先ほど書いた説明の隣のあたりに点をplotする。

例えば、右の図では矢印を置いたあたりにマーカが来て欲しいが、これは座標軸上では大体(4, 275)くらいにある。

なのでここでコマンドウィンドウを使って(4, 275)にデータをプロットしていく。

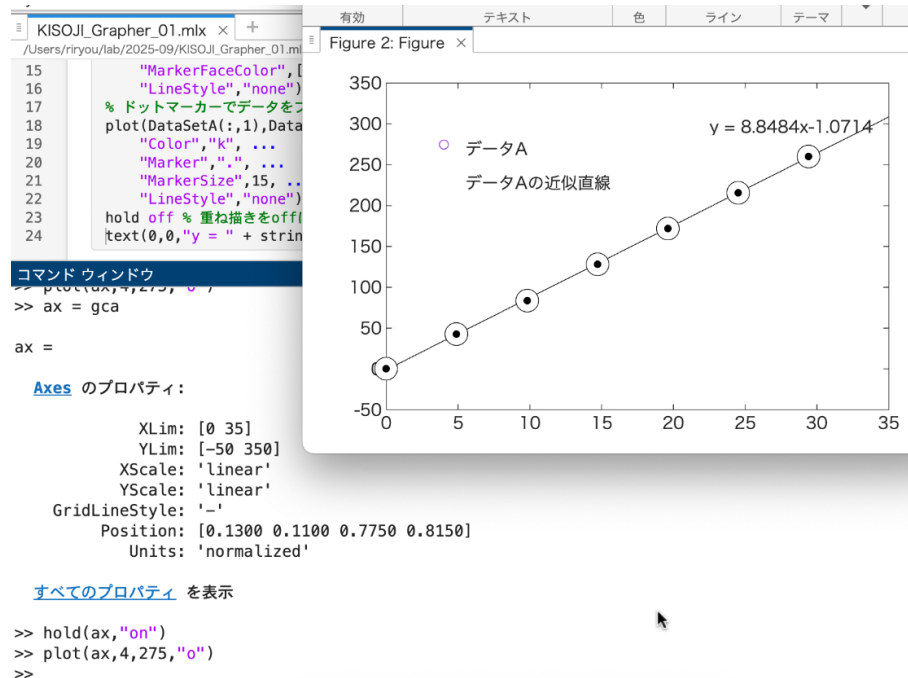


凡例の表示 (マーカーの表示)

次にコマンドウィンドウをクリックして、フォーカスし以降のスライドを参考に、次のコマンドを入力していく。

```
>> ax = gca
>> hold(ax,"on")
>> plot(ax,x座標,y座標,"o")
```

以降のスライドでコマンドについて説明していく

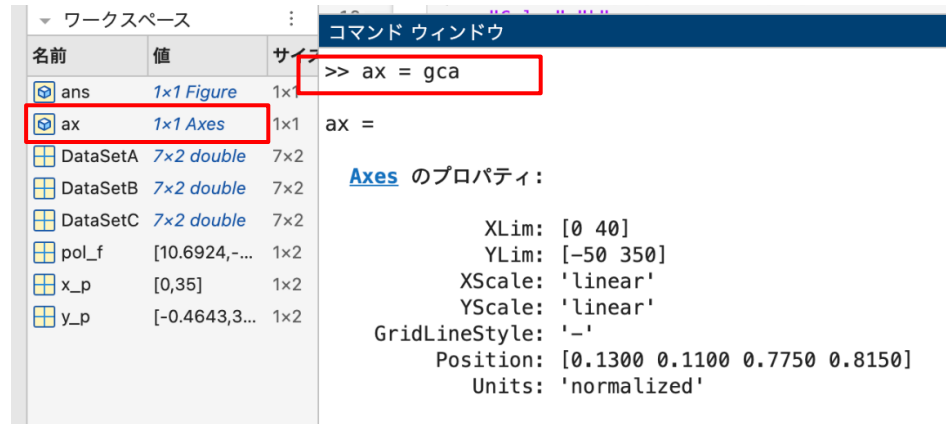


凡例の表示 gca関数

```
>> ax = gca
```

gcaはGet Current Axesをするコマンドで、「今の」座標軸の情報（ハンドル）を返す。

このコマンドにより、変数axを用いて座標軸の情報の参照や変更ができる。



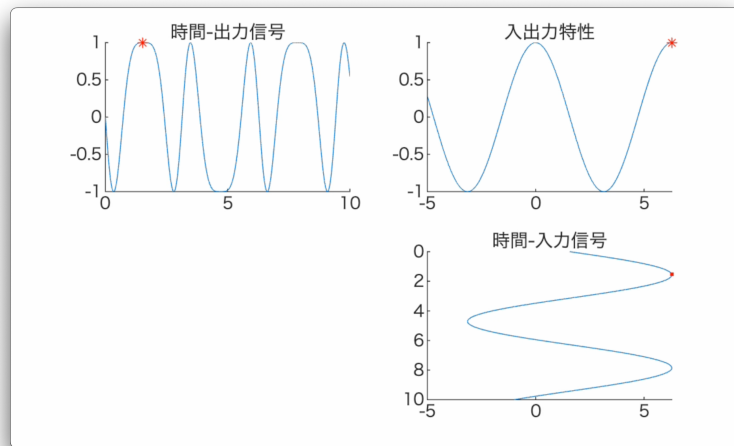
ax = gcaコマンドで軸が読み込まれワークスペースにaxが表示されている。

凡例の表示 (Axesを指定した関数の利用)

```
>>hold(ax,"on")
```

メインのコード内で最後にhold offしているため、ここで改めてhold onをする。

一つ目の引数にaxを渡してある。これにより「どのグラフについて操作するか」を指定が可能になる。複数のグラフを同時に書く場合などに便利。またholdだけでなく、plotやtextなど、様々な関数でこのような指定が可能である。



複数のグラフを並べて書くような場合は、Axesハンドルを用いるとコードが書きやすい

凡例の表示 (Axesハンドルを指定したplot)

```
>> plot(ax,4,275,"o")
```

(4,275)は目視でいい加減に決めたものである。凡例のテキストの位置に合わせて適宜数値を変更すること。

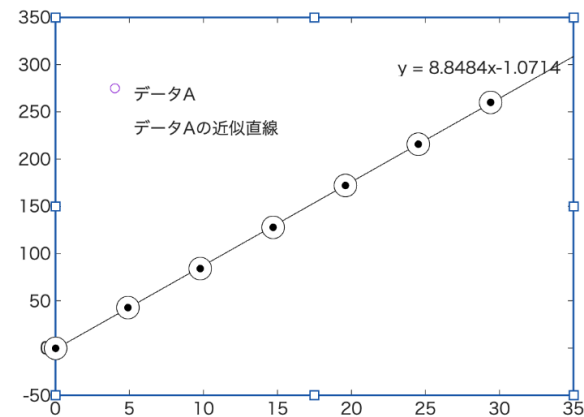
また"o"などを指定してマーカーの形を変更した方が作業がしやすい。

```
4 y_p = polyval(pol_f,x_p);
5 %近似直線のプロット
6 plot(x_p,y_p, ...
7      "Color","k", ...
8      "LineStyle","-")
9 hold on % 重ね描きをonに
10 % 図形マーカーでデータをプロット
11 plot(DataSetA(:,1),DataSetA(:,2), ...
12      "Color","k", ...
13      "Marker","o", ...
14      "MarkerSize",15, ...
15      "MarkerFaceColor","none", ...
16      "LineStyle","none")
17 % ドットマーカーでデータをプロット
18 plot(DataSetA(:,1),DataSetA(:,2), ...
19      "Color","k", ...
20      "Marker",".", ...
21      "MarkerSize",15, ...
22      "LineStyle","none")
23 % hold off % 重ね描きをoffに
24 text(0,0,"y = " + str2num(sprintf('%.4f',pol_f(1)) + str2num(sprintf('%.4f',pol_f(2)) + 'x')))
```

コマンドウィンドウ
GridLineStyle: -
Position: [0.1300 0.1300]
Units: 'normalized'

[すべてのプロパティ](#) を表示

```
>> hold(ax,"on")
>> plot(ax,4,275,"o")
>>
```



凡例の表示（その他のプロパティの編集）

figureウィンドウの「形式」タブ>「選択してプロット編集」をクリックした後、先ほどプロットした点をクリックする。

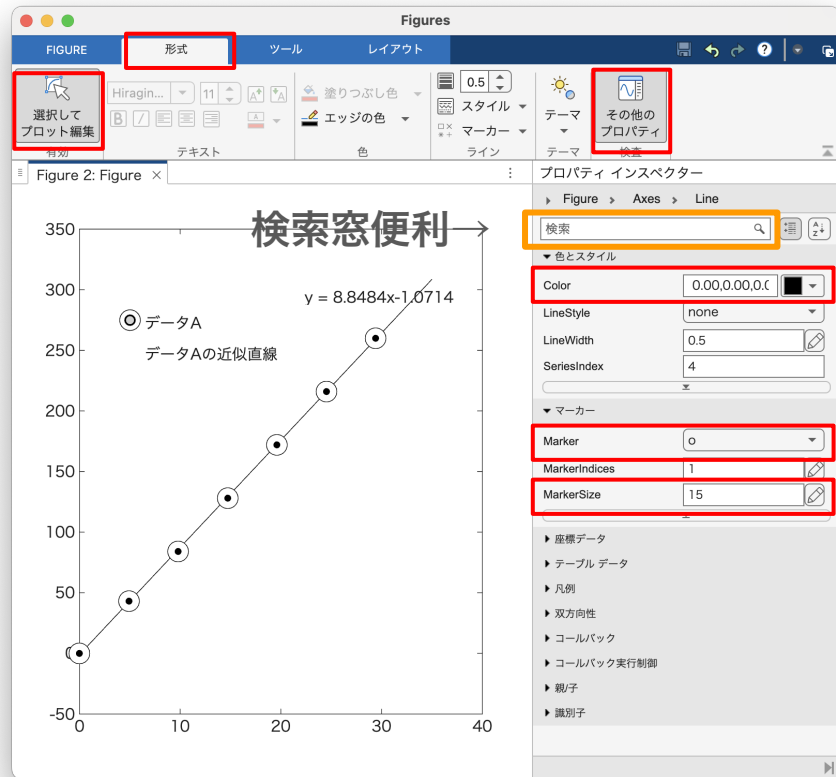
「その他のプロパティ」をクリックして、プロパティインスペクターをひらく。

Colorを黒に

MarkerSize、Markerを適宜変更する。

作業性のために何らかのMarkerを設定しておくこと推奨

また、該当のプロパティ（設定項目）が見つからない場合は、検索窓を適宜使うと便利



凡例の表示（その他のプロパティの編集）

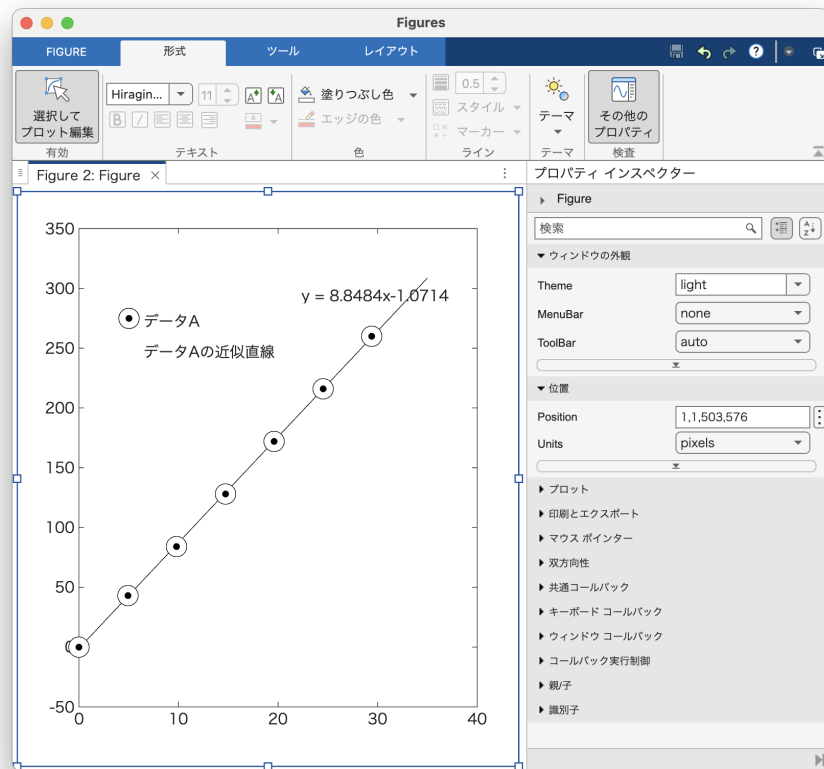
さらにコマンドウィンドウで

```
>> plot(ax,4,275, ".")
```

を実行する。

点がプロットされるので同様にプロパティ
の値を編集する。

右の図のように、「データA」の隣に、マー
カーがプロットされる。



凡例の表示（線のプロット）

同じような手順で「データAの近似直線」の隣にも短い線をプロットする。

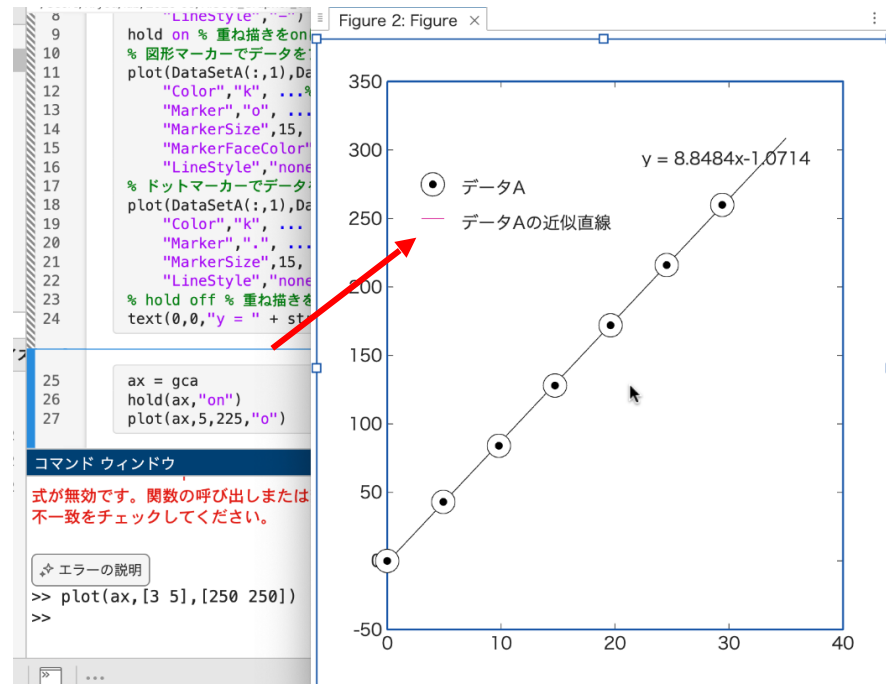
まず適当にグラフを眺めて線の始点と終点の座標を決める。今回は(3,250),(5,250)の間に線を引くことにする。

コマンドウィンドウに

```
>> plot(ax,[3 5],[250 250])  
>>
```

を入力する

線がプロットされると思う



凡例の表示（線のプロット）

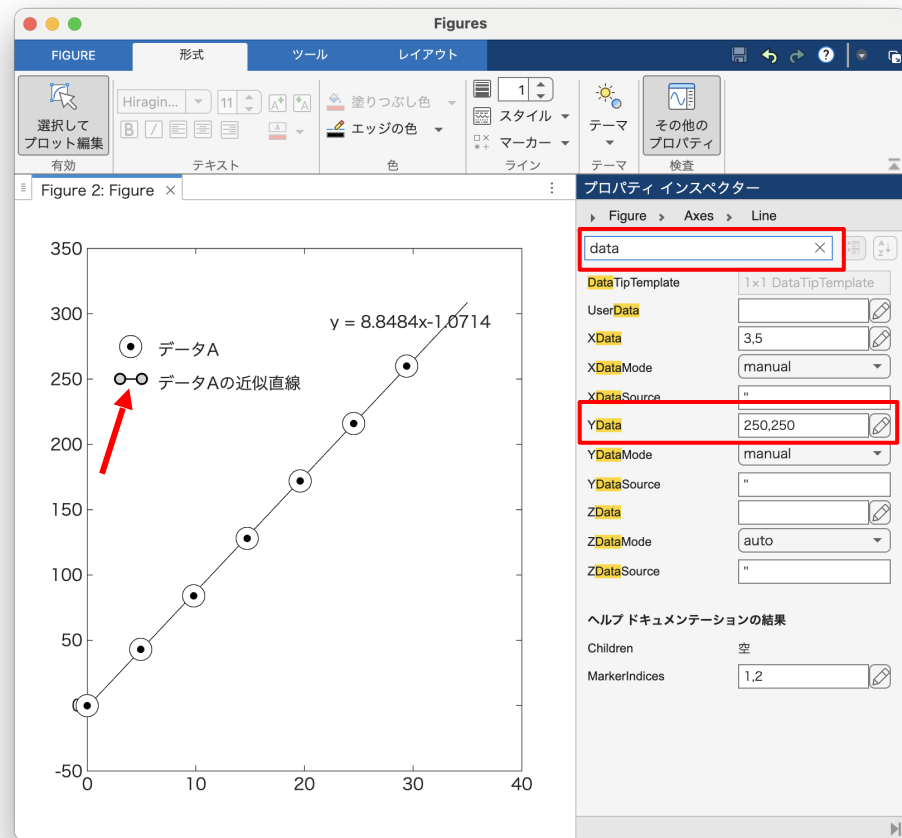
マーカーの時と同様に、「その他のプロパティ」から種々の設定をする。

線をクリックして

Colorを黒に、LineStyle, LineWidthを適宜設定する。

座標の修正もここからできる。

x座標のデータはXDataにY座標のデータはYDataにある、試しにYDataを変えて250から247に数値を書き換えてみる。



凡例の表示（線のプロット）

YDataを

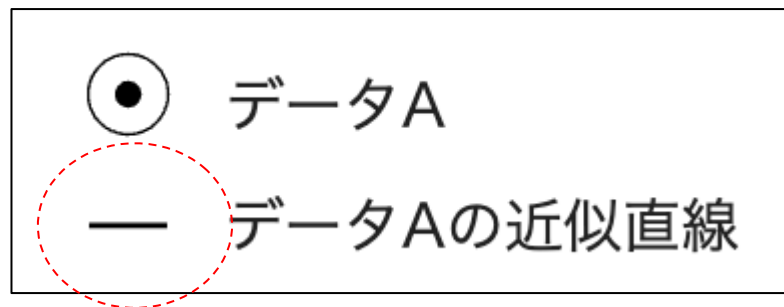
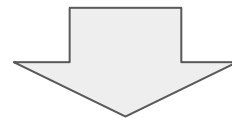
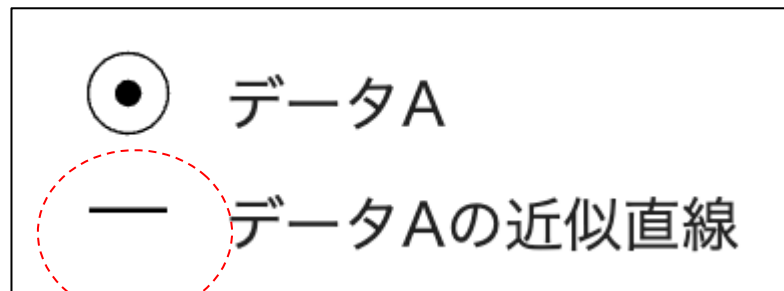
250,250

から

248,248

に書き換えると、プロットの線が移動して、
少し見栄えの良い位置に移ったことがわかる
だろうか。

スタイルや色、位置を間違えてしまっても
plotし直すのは手間がかかるケースがある。
そのような場合はfigureウィンドウからプロ
パティを編集してみると良いかもしれない



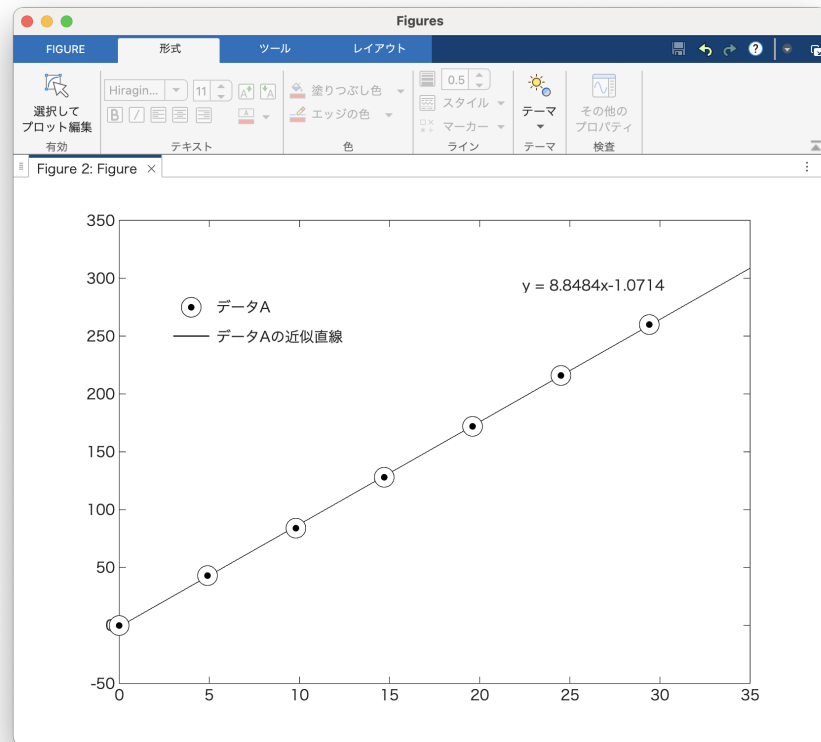
現状のグラフ

マーカーの形を合わせた

近似直線の描画した

近似式の描画した

凡例を載せている

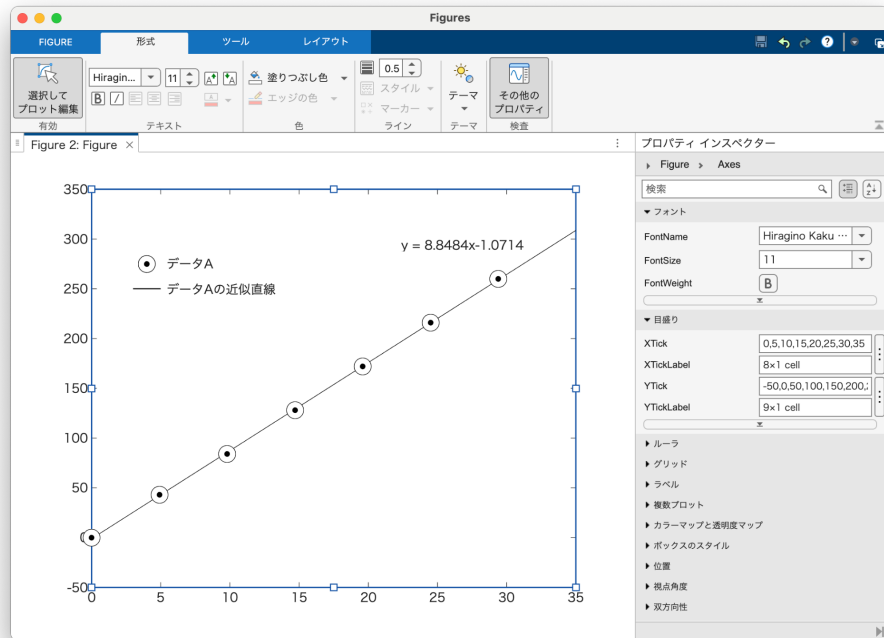


細かい設定

とりあえず、一つだけのプロットで完成度を高めていく。次の点について修正していく。

- 軸の範囲
- ラベルの表示
- タイトルの表示
- グリッドの表示

まず「選択してプロットを編集」「その他のプロパティ」をクリックしておく。



軸の設定 Axes.XLim, YLim

グラフの縦軸を0から350までにしたい

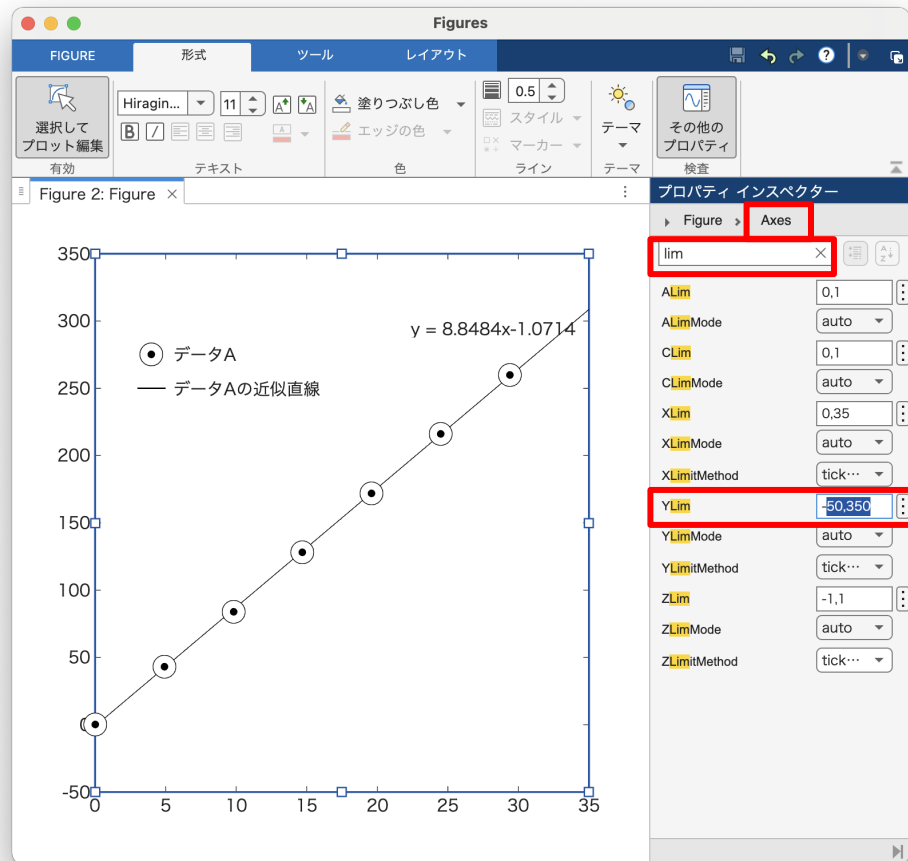
グラフの枠をクリックする。

「プロパティインスペクター」の上端に現在どのオブジェクトが選択されているかが表示されている。「Axes」が表示されていることを確認する。

AxesのYLimの値を

-50, 350から

0, 350に修正する



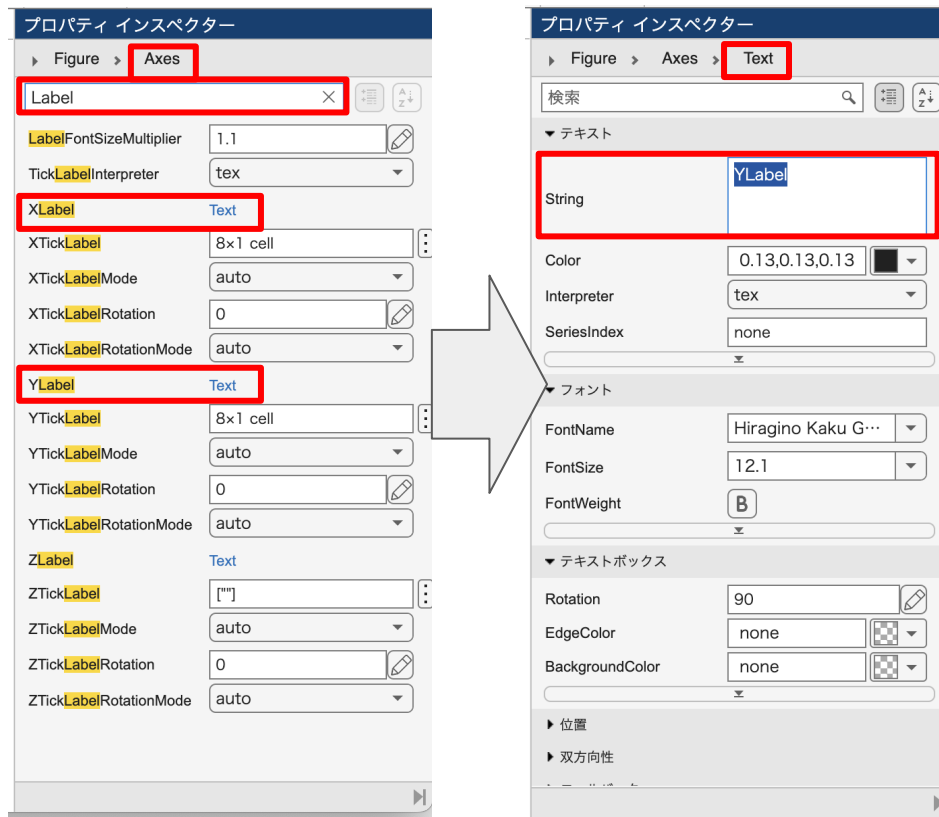
軸ラベルの表示 Axes.XLabel,YLabelプロパティ

同様にAxesオブジェクトを選択して、
XLabel,YLabelを探す

XLabelプロパティにあるTextの部分
をクリックして、textオブジェクトを
選択。

textオブジェクトのStringの部分の
値を表示したいラベルにする

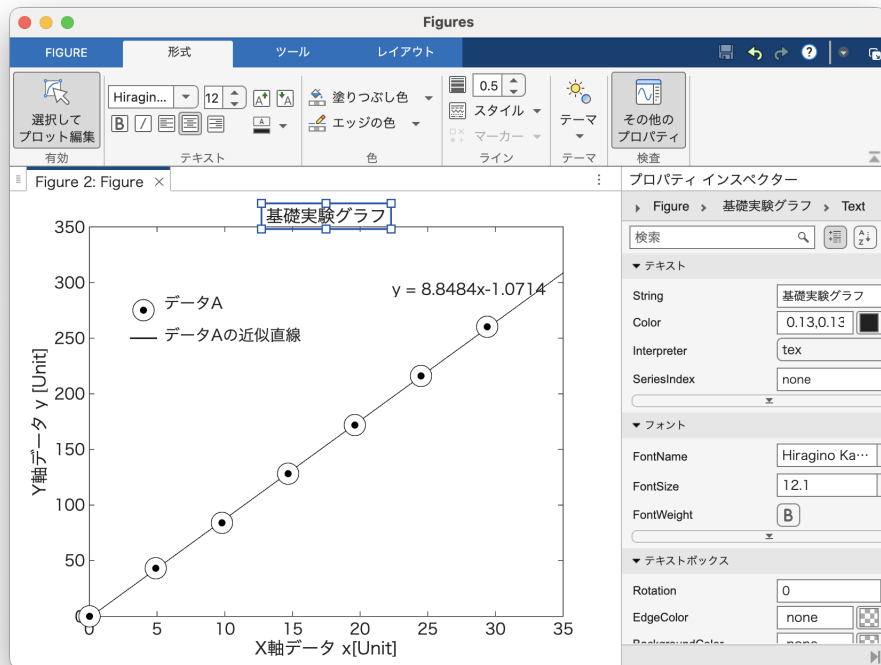
フォントなどもここで変更できる



タイトルの表示

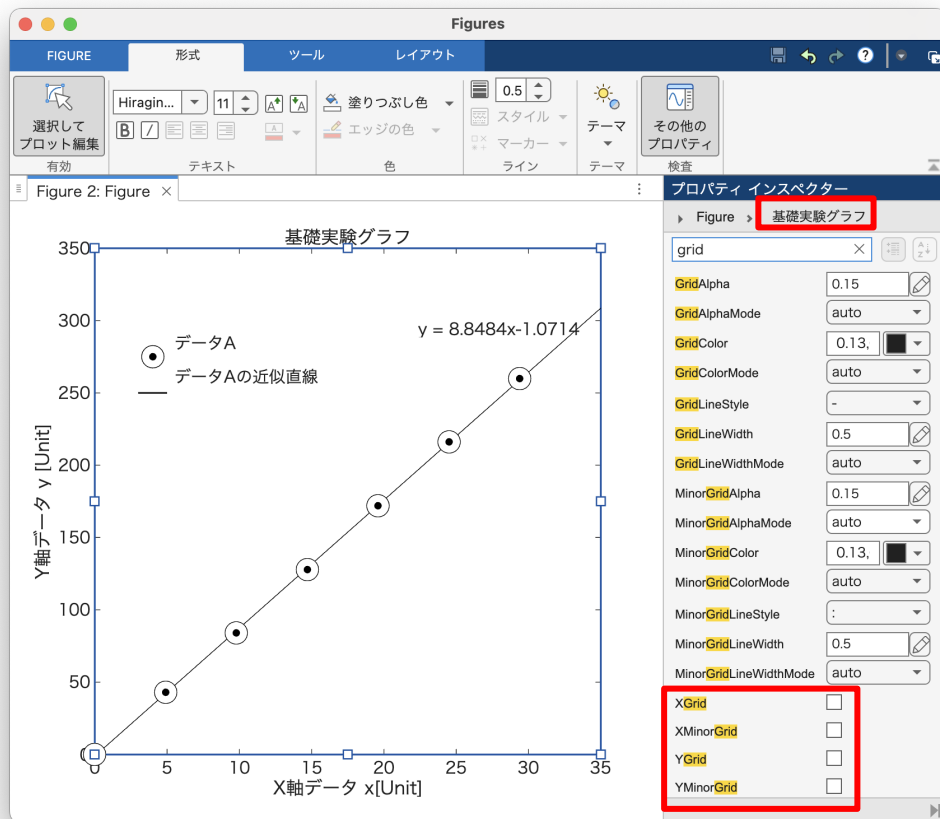
軸ラベルと同様にしてタイトルを表示できる。

ただレポートのフォーマットの的には使うか微妙かもしれない



グリッドの表示 Axes.XGrid,YGrid

Axesオブジェクトの
XGrid,YGridプロパティで
有効・無効の設定ができる。



軸の端に指数表示する `Axes.XAxis.Exponent`

これはプロパティインスペクターから触れないようであるが、コマンドを入力して編集できる。

コマンドウィンドウに

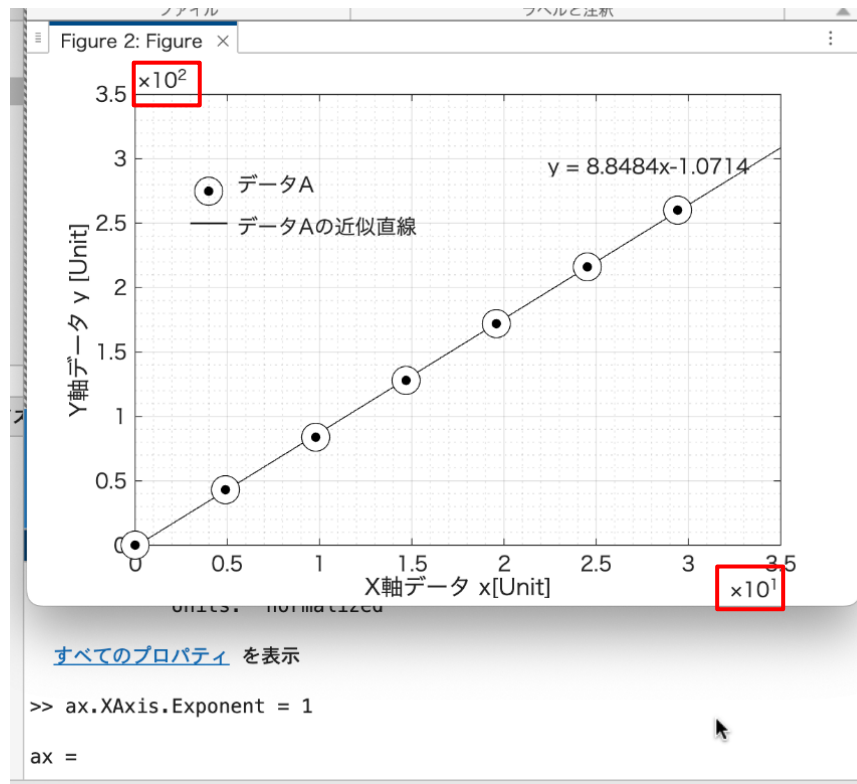
```
ax.XAxis.Exponent = n
```

とすると、指数部だけ別に表示できる。

y軸についても同様にコマンドウィンドウで

```
ax.YAxis.Exponent
```

とすればできる



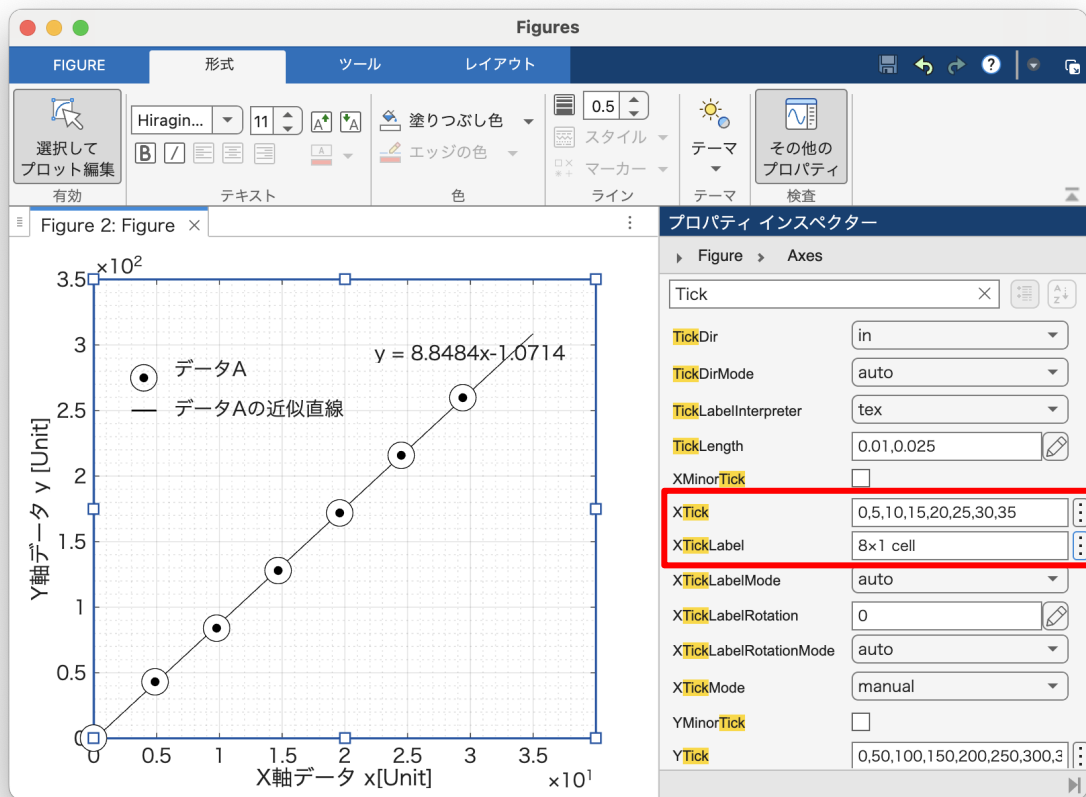
あらかじめ `ax = gca` をしておこう

目盛りのラベルをいじる Axes.XTickLabel

Axes.XTickでは目盛りを置く座標を、

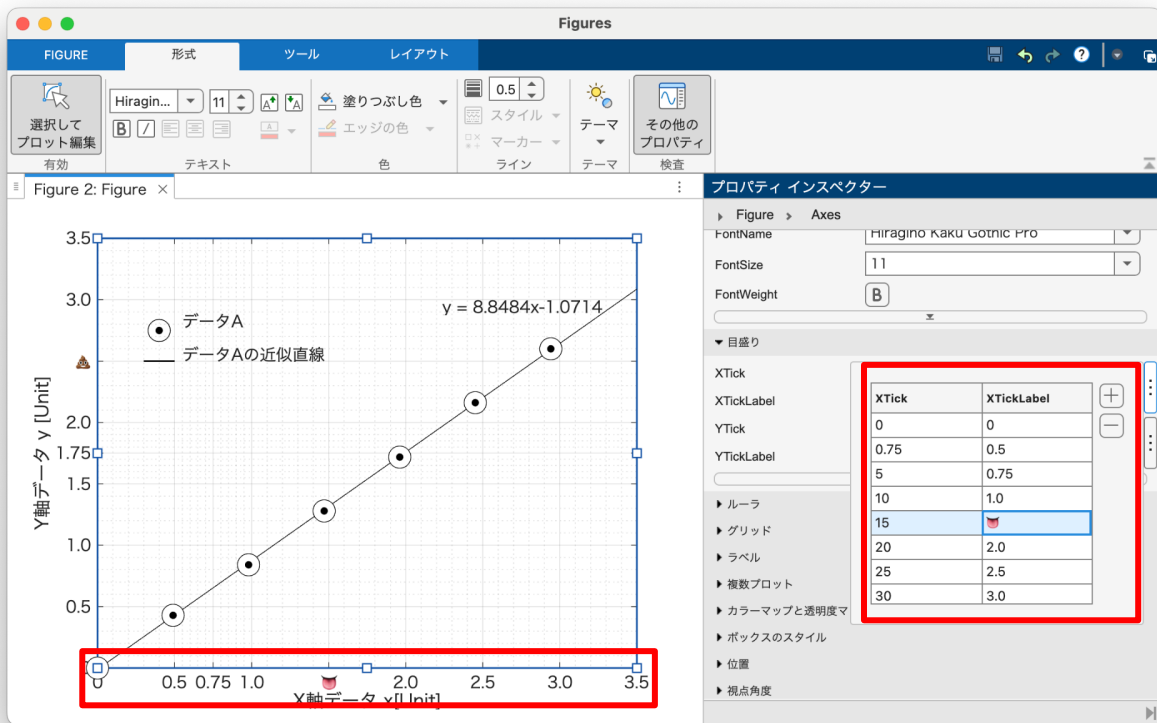
Axes.XTickLabelでは各目盛りに表示する値を変えられる。

値を間違えるとおかしいグラフになるので注意。



目盛りのラベルをいじる Axes.XTickLabel

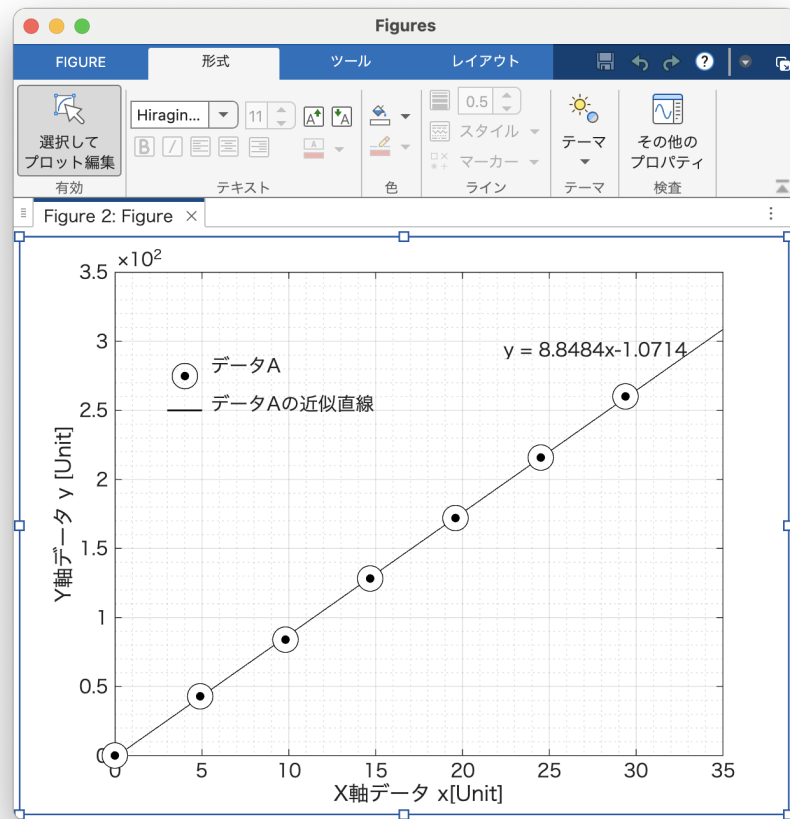
全ての目盛りラベルについて手作業で入力する必要があるが、不規則な目盛りや桁数の調整なども自由にできる。



現状のグラフ

色々いじっていたところ、軸やラベルの設定がよくわからなくなり、手で戻せなくなってしまった。

ExponentModeやXTickModeといったModeが末尾についたプロパティを"manual"から"auto"に直していったら戻ったので、参考までにメモしておく



複数のデータの重ね描き

これまでの手順を通して一つのデータについて体裁を守ったプロットができたと思う。では3つのデータの重ね描きではどうするか。

一番シンプルなのは三倍コードを描き、三倍コマンドを実行して三倍figureウィンドウで作業することである。他にはコードの一部を適宜書き換えて繰り返し実行することでもできる。もっと再利用可能な形のコードを書くことも可能かもしれない。

このスライドの締めには前者の方法で3つのデータのプロットする手順を改めて説明していきたい。が、その前にfig形式のグラフの保存、読み込みについて触れる。

各工程で得られたグラフを保存・読み込みできれば、ミスや手戻りを繰り返してわけがわからなくなった時に、読み込み直して作業を続けられる。

Figureファイルの保存と読み込み

グラフはPNGファイルで保存することができるが、この形式では保存後に「文字を大きくしたい」「誤字を直したい」といった時に、データの取り込みからやり直さなくてはならない。

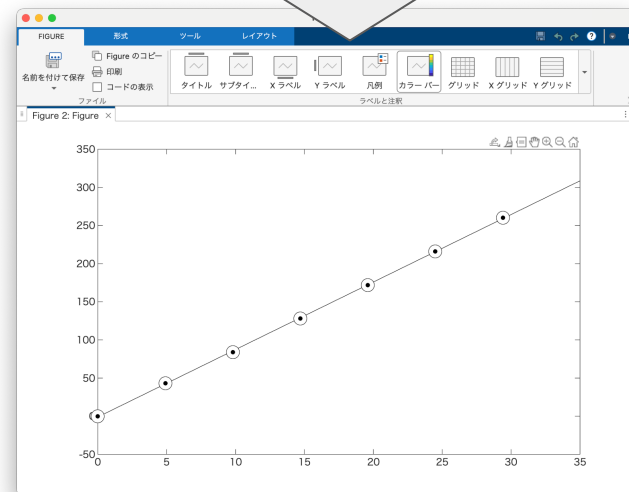
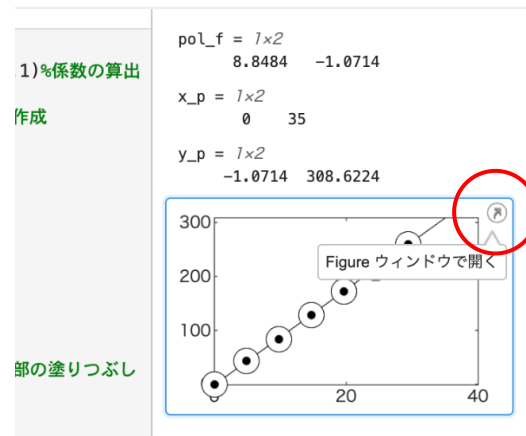
.figファイルとして保存することで、後からプロットの追加や削除などを含めた柔軟な変更を加えることが可能になる。

figureウィンドウから.fig形式で保存する方法と、openfig関数を使った読みだし。そこから変更を加える手順について説明する。

Figureファイルの保存1

出力に表示されているグラフにカーソルを載せると、グラフの右上にマークが表示されるのでこれをクリックする。

するとfigureウィンドウが表示される。



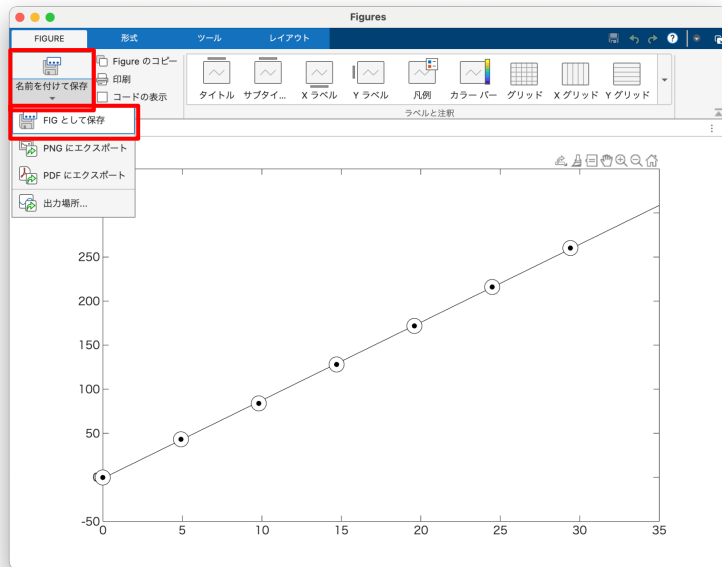
Figureファイルの保存2

figureウィンドウ右上の名前をつけて保存。からFIGとして保存する。

右のグラフの名前は「GraphA.fig」として保存した。違うファイル名をつけた場合は適宜読み替えること。

また今回の説明においては、実験データ、ライブスクリプトと一緒にフォルダにファイルを保存する。

異なるフォルダに保存した場合もファイル名を適当なパスに読み替えること。



figファイルの読み込み openfig

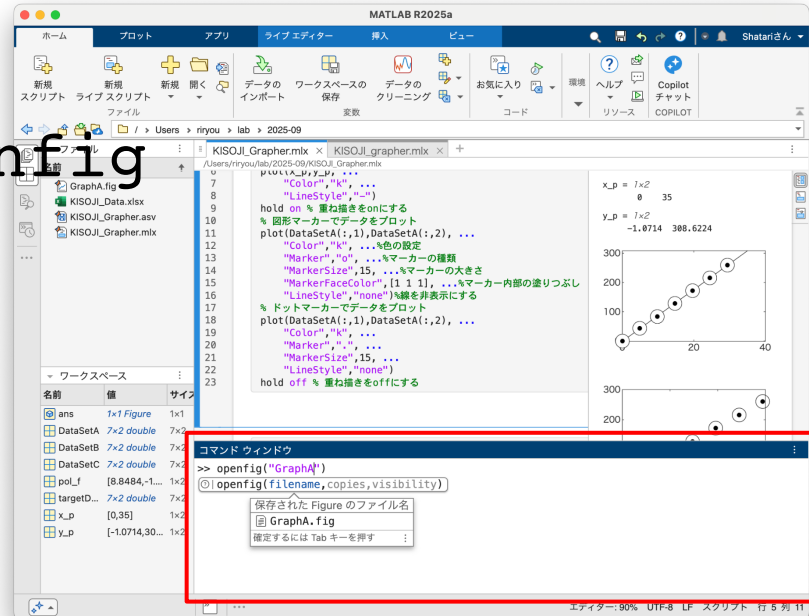
コマンドウィンドウをクリックして
フォーカスする

コマンドウィンドウではライブスクリ
プトのコードなどを逐一入力して実行
することができる。

ここでfigファイルを読み込む関数

`openfig("ファイル名")`

を入力した後、エンターキーを押して
実行する。



コマンドウィンドウ

```
>> openfig("GraphA")
```

```
①| openfig(filename,copies,visibility)
```

保存された Figure のファイル名

GraphA.fig

確定するには Tab キーを押す

openfig

openfigを実行し、ファイルの読み込みが成功したら、読み込んだグラフを表示したfigureウィンドウがポップアップされる。

```
コマンド ウィンドウ
>> openfig("GraphA.fig")

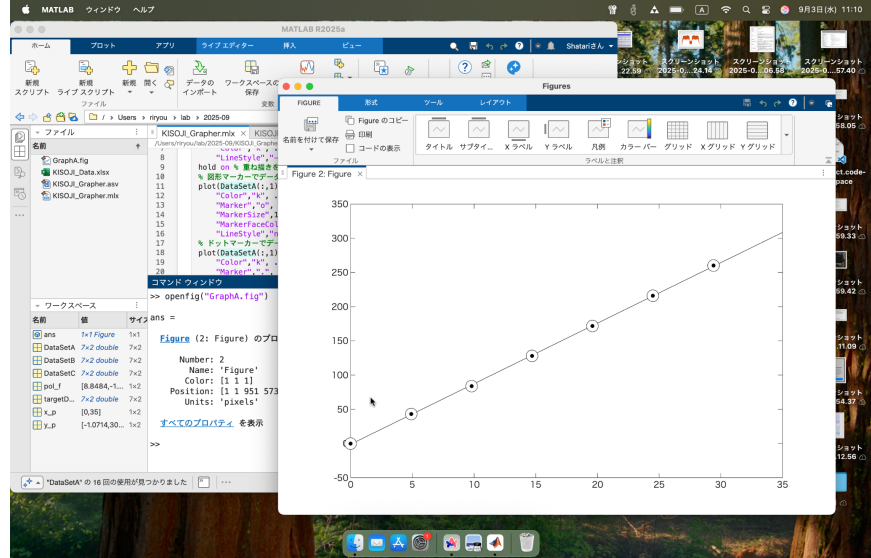
ans =

Figure (2: Figure) のプロパティ:

    Number: 2
      Name: 'Figure'
     Color: [1 1 1]
Position: [1 1 951 573]
    Units: 'pixels'

すべてのプロパティ を表示

>>
```



コマンドウィンドウにはこんな感じに表示されているはず。

openfigで開いた軸にプロット

openfigしてfigureを開いた後、コマンドウィンドウやコードの実行でplot関数を実行してもうまくいかない。新しく別のfigureが作成されてそこにプロットされたり、あるいは上書きされたりしまったりする

```
>> openfig("GraphA.fig")
```

```
>> ax = gca
```

や

```
>> f = openfig("GraphA.fig")
```

```
>> ax = f.Children
```

あるいは.figをダブルクリックした直後にコマンドウィンドウで

```
>> ax = gca
```

などとしてAxesハンドルを取得し、plot時にこれを引数として与えれば良い。

3つのデータのプロット

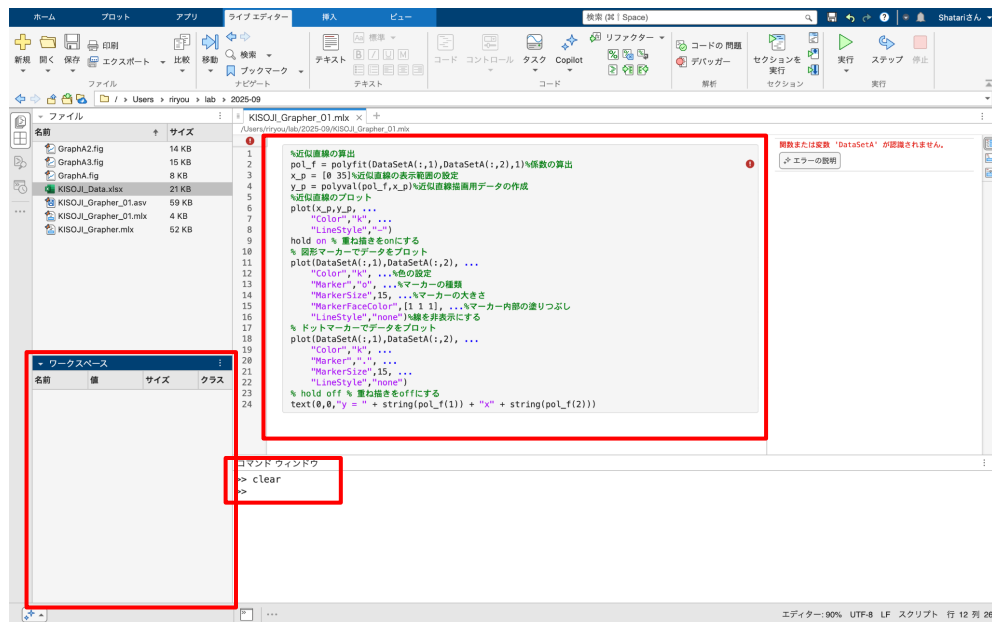
最後におさらいをかねて3つのデータをまとめてプロットしていく。

せっかくなのでデータのインポートからやり直す。コマンドウィンドウで

```
>> clear
```

これで変数などメモリに読み出した各種情報をリセットできる。心配な方は「ワークスペースの保存」をしておく
と現状に戻せるので安心

次にデータのプロットに用いたライブスクリプトを開いておく



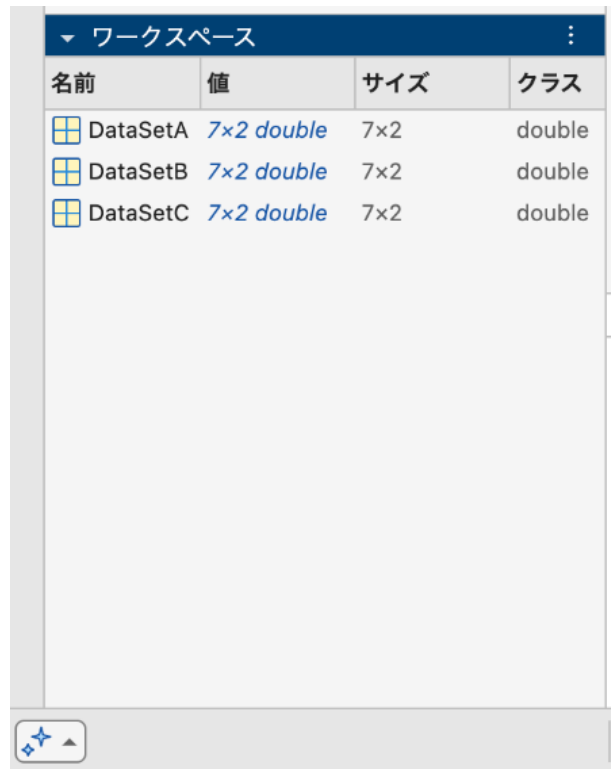
ワークスペースが
空っぽになる

3つのデータのプロット（インポート）

データの入ったエクセルファイルをインポートツールでひらく

範囲選択、変数名入力、数値行列形式でデータを変数に保存する。

数値行列形式で読み込まなくてもコードで処理はできるが、形式によってコードの細かい部分が異なってくるので今回のコードではエラーがでてしまう



The screenshot shows a software interface with a table titled 'ワークスペース' (Workspace). The table has four columns: '名前' (Name), '値' (Value), 'サイズ' (Size), and 'クラス' (Class). It contains three rows of data, each with a small grid icon to the left of the name.

名前	値	サイズ	クラス
DataSetA	7×2 double	7×2	double
DataSetB	7×2 double	7×2	double
DataSetC	7×2 double	7×2	double

3つのデータのプロット（コピペ）

今回はコードをコピペしてそれぞれのデータについて個別に処理をするようにコードを書く。関数にしてみたりfor文にしても良いと思う。

まずコード全体をコピーして、末尾に2回ペースとする。

またプロットするデータやコードに合わせて適宜編集を加える（次のページに詳細）

```
025-09
KISOJL_Grapher_01.mlx
/Users/riryoub/2025-09/KISOJL_Grapher_01.mlx

1
2 %近似直線の算出
3 pol_f = polyfit(DataSetA(:,1),DataSetA(:,2),1)%係数の算出
4 x_p = [0 35]%近似直線の表示範囲の設定
5 y_p = polyval(pol_f,x_p)%近似直線描画用データの作成
6 %近似直線のプロット
7 plot(x_p,y_p, ...
8     "Color","k", ...
9     "LineStyle","-")
10 hold on % 重ね描きをonにする
11 % 図形マーカーでデータをプロット
12 plot(DataSetA(:,1),DataSetA(:,2), ...
13     "Color","k", ...%色の設定
14     "Marker","o", ...%マーカーの種類
15     "MarkerSize",15, ...%マーカーの大きさ
16     "MarkerFaceColor",[1 1 1], ...%マーカー内部の塗りつぶし
17     "LineStyle","none")%線を非表示にする
18 % ドットマーカーでデータをプロット
19 plot(DataSetA(:,1),DataSetA(:,2), ...
20     "Color","k", ...
21     "Marker",".", ...
22     "MarkerSize",15, ...
23     "LineStyle","none")
24 % hold off % 重ね描きをoffにする
25 text(0,0,"y = " + string(pol_f(1)) + "x" + string(pol_f(2)))
```

コマンドウィンドウ

025-09

KISOJL_Grapher_01.mlx +
/Users/riryoub/2025-09/KISOJL_Grapher_01.mlx

```
1
2 %近似直線の算出
3 pol_f = polyfit(DataSetA(:,1),DataSetA(:,2),1)%係数の算出
4 x_p = [0 35]%近似直線の表示範囲の設定
5 y_p = polyval(pol_f,x_p)%近似直線描画用データの作成
6 %近似直線のプロット
7 plot(x_p,y_p, ...
8     "Color","k", ...
9     "LineStyle","-")
10 hold on % 重ね描きをonにする
11 % 図形マーカーでデータをプロット
12 plot(DataSetA(:,1),DataSetA(:,2), ...
13     "Color","k", ...%色の設定
14     "Marker","o", ...%マーカーの種類
15     "MarkerSize",15, ...%マーカーの大きさ
16     "MarkerFaceColor",[1 1 1], ...%マーカー内部の塗りつぶし
17     "LineStyle","none")%線を非表示にする
18 % ドットマーカーでデータをプロット
19 plot(DataSetA(:,1),DataSetA(:,2), ...
20     "Color","k", ...
21     "Marker",".", ...
22     "MarkerSize",15, ...
23     "LineStyle","none")
24 % hold off % 重ね描きをoffにする
25 text(0,0,"y = " + string(pol_f(1)) + "x" + string(pol_f(2)))
26
27 %近似直線の算出
28 pol_f = polyfit(DataSetA(:,1),DataSetA(:,2),1)%係数の算出
29 x_p = [0 35]%近似直線の表示範囲の設定
30 y_p = polyval(pol_f,x_p)%近似直線描画用データの作成
31 %近似直線のプロット
32 plot(x_p,y_p, ...
33     "Color","k", ...
34     "LineStyle","-")
35 hold on % 重ね描きをonにする
36 % 図形マーカーでデータをプロット
37 plot(DataSetA(:,1),DataSetA(:,2), ...
38     "Color","k", ...%色の設定
39     "Marker","o", ...%マーカーの種類
40     "MarkerSize",15, ...%マーカーの大きさ
41     "MarkerFaceColor",[1 1 1], ...%マーカー内部の塗りつぶし
42     "LineStyle","none")%線を非表示にする
43 % ドットマーカーでデータをプロット
44 plot(DataSetA(:,1),DataSetA(:,2), ...
45     "Color","k", ...
46     "Marker",".", ...
47     "MarkerSize",15, ...
48     "LineStyle","none")
49 % hold off % 重ね描きをoffにする
50 text(0,0,"y = " + string(pol_f(1)) + "x" + string(pol_f(2)))
51 %近似直線の算出
52 pol_f = polyfit(DataSetA(:,1),DataSetA(:,2),1)%係数の算出
```

3つのデータのプロット（適宜修正）

修正する箇所としては、

- コード中のhold onは最初のもののみ残り、二番目以降をコメントアウトor削除
- コード中のhold offは全部削除、または最後の一つのみ残して他をコメントアウトor削除
- DataSetA等の変数名を適宜変更
- マーカー・ラインスタイルを適宜変更

などがある、自身のコードやグラフの体裁、実行時のエラーなどを考慮して適当に修正すること

```
KISOJI_Grapher_01.mlx × +
/Users/riryou/lab/2025-09/KISOJI_Grapher_01.mlx

21     "MarkerSize",15, ...
22     "LineStyle","none")
23     % hold off % 重ね描きをoffにする
24     text(0,0,"y = " + string(pol_f(1)) + "x" + string(pol_f(2)))
25
26 %DataSetBについての処理
27 %近似直線の算出
28 pol_f = polyfit(DataSetB(:,1),DataSetB(:,2),1)%係数の算出
29 x_p = [0 35]%近似直線の表示範囲の設定
30 y_p = polyval(pol_f,x_p)%近似直線描画用データの作成
31 %近似直線のプロット
32 plot(x_p,y_p, ...
33     "Color","k", ...
34     "LineStyle","-")
35 % hold on % 重ね描きをonにする
36 % 図形と近似直線をデータとプロット
37 plot(DataSetB(:,1),DataSetB(:,2), ...
38     "Color","r", ...%色の設定
39     "Marker","^", ...%マーカーの種類
40     "MarkerSize",15, ...%マーカーの大きさ
41     "MarkerFaceColor",[1 1 1], ...%マーカー内部の塗りつぶし
42     "LineStyle","none")%線を非表示にする
43 % ドットマーカーでデータをプロット
44 plot(DataSetB(:,1),DataSetB(:,2), ...
45     "Color","k", ...
46     "Marker",".", ...
47     "MarkerSize",15, ...
48     "LineStyle","none")
49 % hold off % 重ね描きをoffにする
50 text(0,0,"y = " + string(pol_f(1)) + "x" + string(pol_f(2)))
51
52 %DataSetCについての処理
53 %近似直線の算出
54 pol_f = polyfit(DataSetA(:,1),DataSetA(:,2),1)%係数の算出
55 x_p = [0 35]%近似直線の表示範囲の設定
```

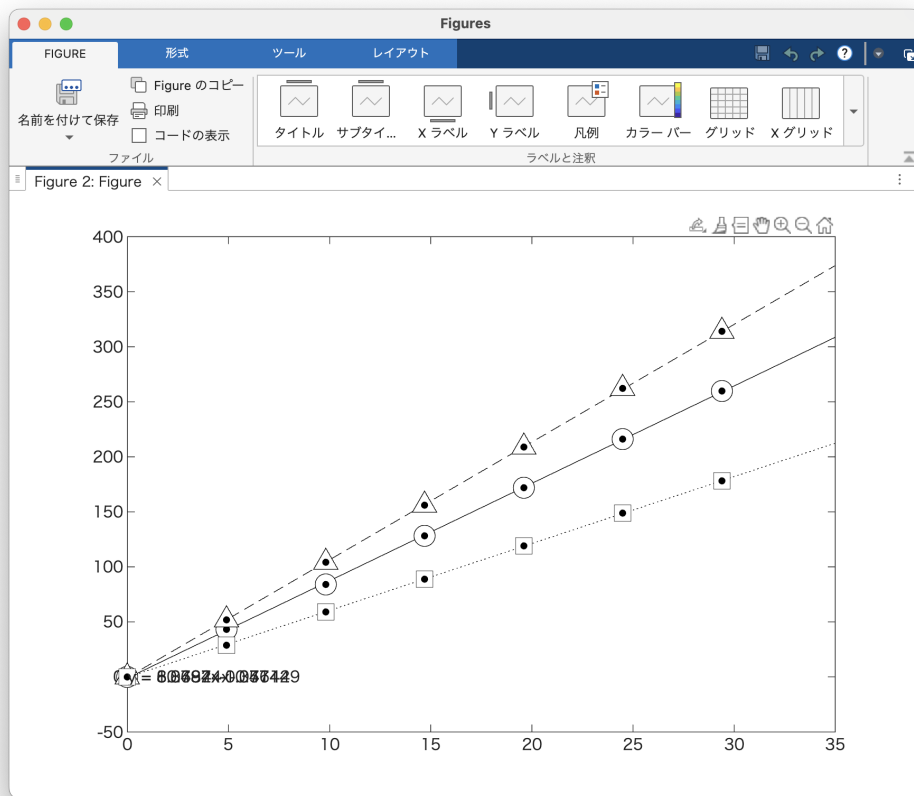
グラフの修正

次のようなグラフが得られた

近似式が重なってしまったり、軸の範囲が $[-50, 400]$ だったりしているので適宜修正していく。最初やった時とは順番が前後するが、プロパティエディタを用いて

- 近似式の位置
- 軸の範囲
- 凡例の説明の表示 etc...

などを先に済ませる。コードで設定し損ねた部分があればこの段階で適宜修正する。



グラフの修正

近似式はどの近似直線に対応しているのかを、傾きなどから適当に判断しても良いし、高次の近似を行っていて判断がつかない場合には、出力から確認しても良い。

ライブスクリプトの出力の部分をクリックすると、その出力を得た行にカーソルが飛んでいくので、対応するコードの場所や出力の数値から判断できる。

The screenshot shows the KISOJL Grapher interface. On the left, the MATLAB code is displayed with line numbers 34 to 60. A red box highlights the following code block:

```
50 %近似直線の算出
51 pol_f = polyfit(DataSetC(:,1),DataSetC(:,2),1)%係数の算出
52 x_p = [0 351]%近似直線の表示範囲の設定
53 y_p = polyval(pol_f,x_p)%近似直線描画用データの作成
54 %近似直線のプロット
55 plot(x_p,y_p, ...
56      "Color","k", ...
57      "LineStyle","-");
58 % 図形マーカでデータをプロット
59 plot(DataSetC(:,1),DataSetC(:,2), ...
60      "Color","k", ...%色の設定
```

On the right, the output is displayed. A red box highlights the output table, with a red arrow pointing to it and the text "クリック!". The output table is as follows:

pol_f = 1x2	
1	2
6.0787	-0.3571

Below the table, the polynomial equation is shown: $y_p = 1 \times 2 - 0.3571 \times 212.3988$. At the bottom, a plot is visible with data points and a fitted line.

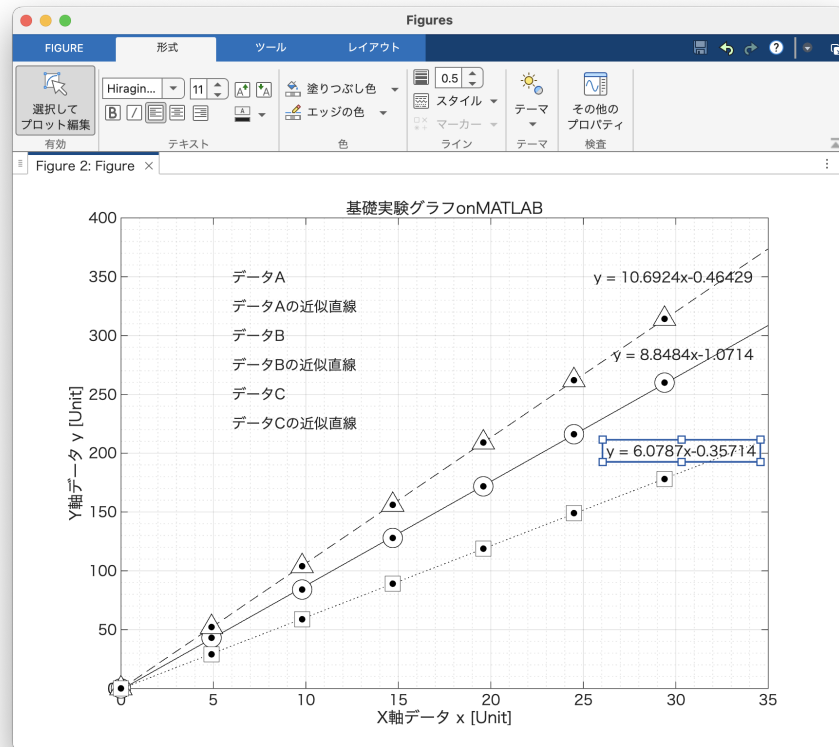
ここにテキストカーソルが移動するので、コード上の位置や自分でつけたコメントなどを頼りにすると、DataSetCの近似式が $y = 6.0787x - 0.3571$ に対応するとわかる。

出力と対応する行の頭にテキストカーソルが移動する。

グラフの修正

このようなグラフを得られた。

適度にfig形式での保存を挟んでおく
と、誤字やミスの修正が楽になる。



凡例のマーカの表示

グラフのサイズや、凡例テキストの位置を適当に動かして位置を調整する。

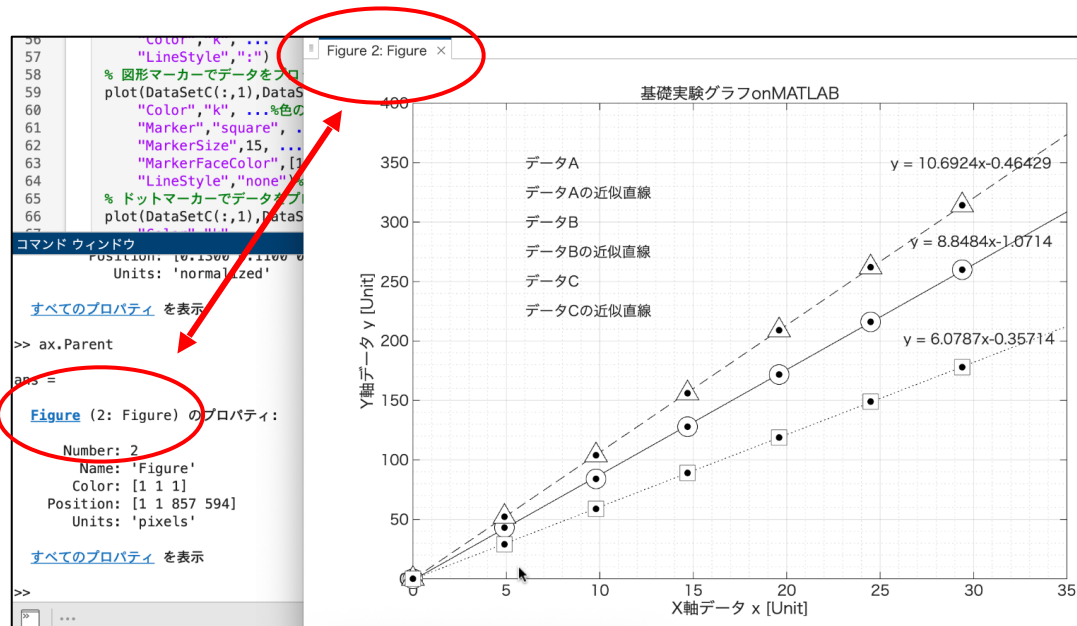
コマンドウィンドウで

```
>> ax = gca
```

```
>> hold(ax,"on")
```

を実行する。

コマンドウィンドウで適当なプロットをして凡例の表示をする。



tipsとして

```
>> ax.Parent
```

とすると取得したAxesハンドルが属するfigureの情報を確認できる。

上の赤丸の部分を確認するとどのグラフと対応するハンドルを取得したのかわかる。

何枚かfigureがでてきてよくわからなくなった時は活用しよう

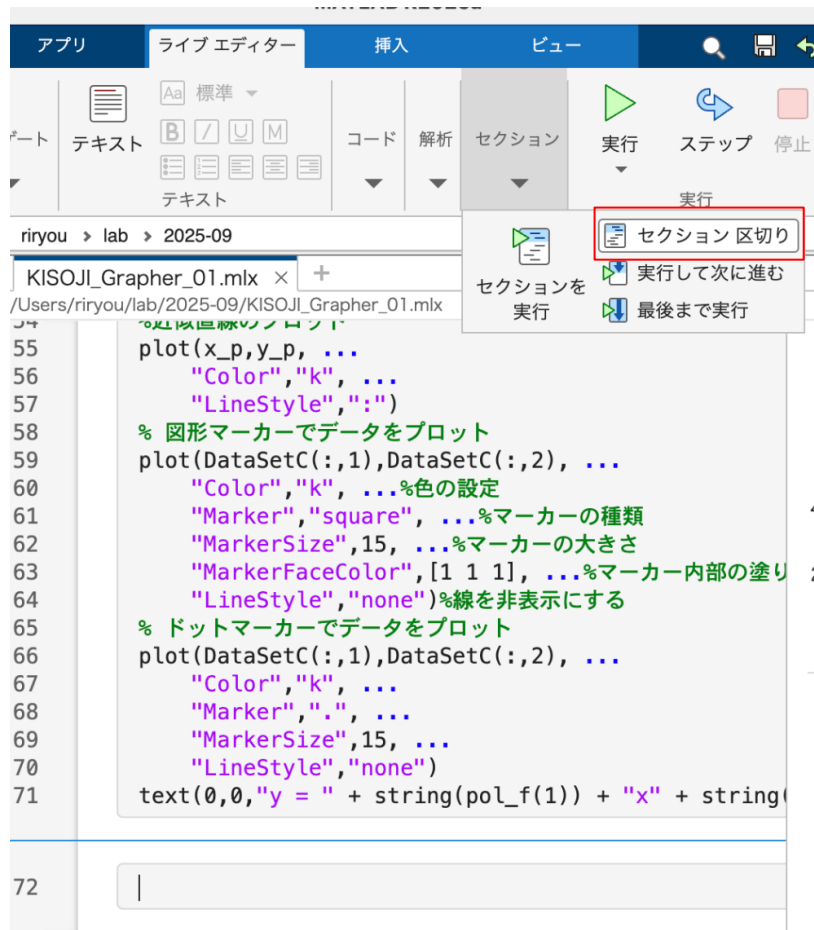
凡例のマーカの表示

コマンドウィンドウでマーカをおく方法は既に説明したため、せっかくなので少し違う方法を紹介しておく

テキストカーソルをコードの末尾において、「ライブエディター>セクション>セクション区切り」をクリックする。

するとテキストカーソル位置にセクション区切りが挟まる。

次にここにプロットするコマンドを書き「セクションを実行」を行う（次ページ）



凡例のマーカの表示

凡例の位置を踏まえて今回は、次のように書いた

MATLABのコードは全体実行だけでなく、セクションのみの実行を行うことができる。

上のリボンから「セクションを実行」を探してクリックする。こちらの手順でも `ax=gca` を事前に実行しておいた方が良い。



```
rs > riryou > lab > 2025-09

KISOJI_Grapher_01.mlx * x
/Users/riryou/lab/2025-09/KISOJI_Grapher_01.mlx

66 plot(datasets(:,1), datasets(:,2), ...
67     "Color", "k", ...
68     "Marker", ".", ...
69     "MarkerSize", 15, ...
70     "LineStyle", "none")
71 text(0,0,"y = " + string(pol_f(1)) + "x" + string(

72 hold(ax,"on")
73 %DataSetAの凡例用
74 plot(ax,5,350,"o",MarkerSize=15)
75 plot(ax,5,350,".",MarkerSize=15)
76 plot(ax,[4 6],[325 325],"--")
77 %DataSetBの凡例用
78 plot(ax,5,300,"^",MarkerSize=15)
79 plot(ax,5,300,".",MarkerSize=15)
80 plot(ax,[4 6],[275 275],"--")
81 %DataSetCの凡例用
82 plot(ax,5,250,"square",MarkerSize=15)
83 plot(ax,5,250,".",MarkerSize=15)
84 plot(ax,[4 6],[225 225],":")

コマンド ウィンドウ
```

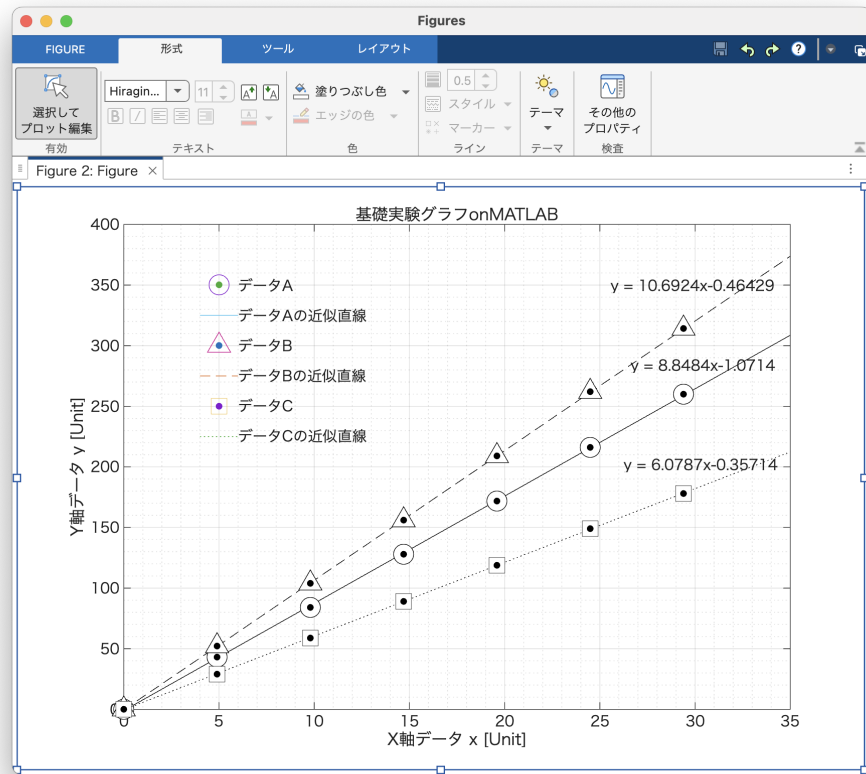
最後の修正

こんな感じに表示された。

色の指定を忘れていたので、カラフルになっている。

最後にプロパティをいじって色と凡例の位置を少し修正し、四角形を置いて凡例を囲う。レポートの本文ではセリフ付きのフォントが指定されていることがあるので、合わせてグラフもセリフ付きのフォントに修正しておく。

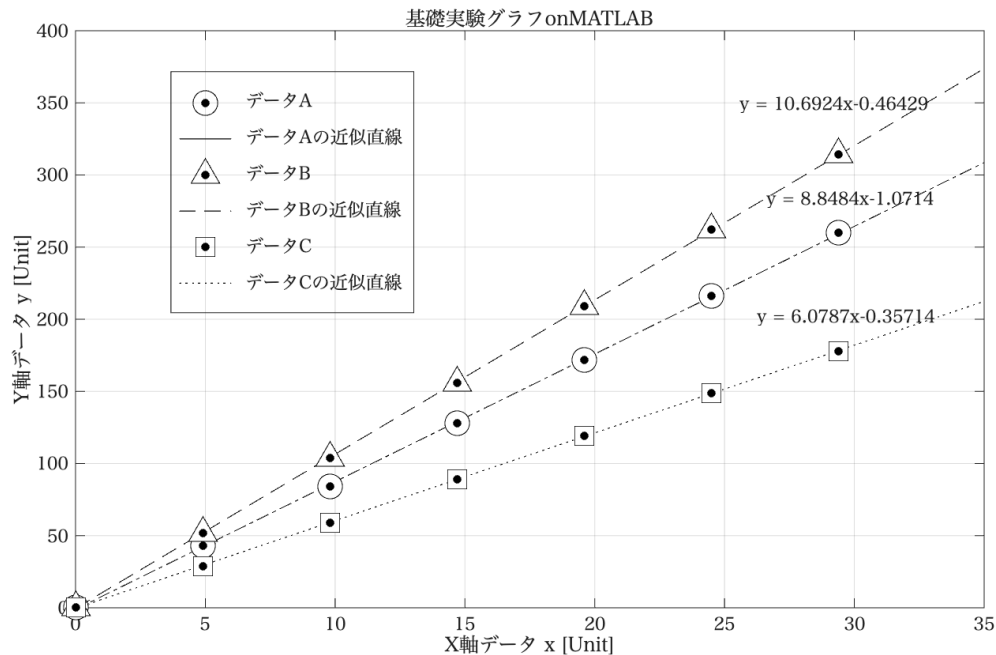
完成したグラフを次のスライドに示す



エクスポートして完成

「名前をつけて保存」から「PNGにエクスポート」を選択し、出力して得られた完成のグラフを右に示す。

問題なくここまでの手順を終えられただろうか？私はエクスポートの段階になってエラーが表示されたため、保存したファイルを読み込んで修正した。やはり作業途中でこまめに保存しておくことをおすすめする（特にMac版はWin版よりも動作が不安定な気がする。）



<https://jp.mathworks.com/help/matlab/index.html>

エクスポートして完成

うまく行かない場合は各自Mathworksの資料を確認して欲しい。

MATLABは有償ソフトのため、Qiitaや各種SNS、その他有志による情報は無償ソフトに劣る。一方で、公式のドキュメントは大変丁寧に作られており、非常に参考になる。

果たしてどれくらいの方がここまで目を通してくれるだろうか。もしこの資料が役に立ってくれたのなら大変嬉しい。

<https://jp.mathworks.com/help/matlab/index.html>

↑公式ドキュメントへのリンク

サンプルコードについて

この資料は研究室のサイトで共有してもらえると聞いている。なのでサンプルとして私が作成した`.mlx`ファイルを一緒にサイトにアップロードして共有できればと考えているが、作成時点ではどうなるかが不透明である。

なので今回作成したコードをコピーして以降のスライドに貼っておく。

ただPDFはテキストを良い感じにコピーできず、勝手に改行が入ったり色々するので、次のスライドをコピーしてもいくつかエラーが出る。あくまで参考程度に

```

%近似直線の算出
pol_f = polyfit(DataSetA(:,1),DataSetA(:,2),1)%係数の算出
x_p = [0 35]%近似直線の表示範囲の設定
y_p = polyval(pol_f,x_p)%近似直線描画用データの作成
%近似直線のプロット
plot(x_p,y_p,'...',
     "Color","k",',...
     "LineStyle","_")
hold on %重ね描きをonにする
% 図形マーカーでデータをプロット
plot(DataSetA(:,1),DataSetA(:,2), ...
     "Color", 'k', ...%色の設定
     "Marker", "o", ...%マーカーの種類
     "MarkerSize",15, ...%マーカーの大きさ
     "MarkerFaceColor",[1 1 1], ...%マーカー内部の塗りつぶし
     "LineStyle", "none")%線を非表示にする
% ドットマーカーでデータをプロット
plot(DataSetA(:,1),DataSetA(:,2), ...
     "Color","k",',...
     "Marker","^", ...%マーカーの種類
     "MarkerSize",15, ...%マーカーの大きさ
     "MarkerFaceColor",[1 1 1], ...%マーカー内部の塗りつぶし
     "LineStyle", "none")%線を非表示にする
% ドットマーカーでデータをプロット
plot(x_p,y_p,'...',
     "Color","k",',...
     "LineStyle","_")% 図形マーカーでデータをプロット
plot(DataSetB(:,1),DataSetB(:,2), ...
     "Color", 'k', ...%色の設定
     "Marker", "x", ...%マーカーの種類
     "MarkerSize",15, ...%マーカーの大きさ
     "MarkerFaceColor",[1 1 1], ...%マーカー内部の塗りつぶし
     "LineStyle", "none")%線を非表示にする
% ドットマーカーでデータをプロット
plot(x_p,y_p,'...',
     "Color","k",',...
     "LineStyle","_")% 図形マーカーでデータをプロット
plot(DataSetC(:,1),DataSetC(:,2), ...
     "Color", 'k', ...%色の設定
     "Marker", "square", ...%マーカーの種類
     "MarkerSize",15, ...%マーカーの大きさ
     "MarkerFaceColor",[1 1 1], ...%マーカー内部の塗りつぶし
     "LineStyle", "none")%線を非表示にする
% ドットマーカーでデータをプロット
plot(DataSetC(:,1),DataSetC(:,2), ...
     "Color","k",',...
     "Marker","^", ...%マーカーの種類
     "MarkerSize",15, ...%マーカーの大きさ
     "MarkerFaceColor",[1 1 1], ...%マーカー内部の塗りつぶし
     "LineStyle", "none")%線を非表示にする
text(0,0,"y =" + string(pol_f(1)) + "x" + string(pol_f(2)))
ax = gca
hold(ax,"on")
%DataSetAの凡例用
plot(ax,5,350,"o",MarkerSize=15)
plot(ax,5,350,"x",MarkerSize=15)
plot(ax,[4 6],[325 325],"_")
%DataSetBの凡例用
plot(ax,5,300,"^",MarkerSize=15)
plot(ax,5,300,"x",MarkerSize=15)
plot(ax,[4 6],[275 275],"_")
%DataSetCの凡例用
plot(ax,5,250,"square",MarkerSize=15)
plot(ax,5,250,"x",MarkerSize=15)
plot(ax,[4 6],[225 225],"_")

```