Matlab によるグラフ描画

西村竜一*
* 東北大学電気通信研究所ブレインコンピューティング研究部門

1　はじめに

東北大学情報シナジーセンターで提供している Matlab は、数値解析ソフトウェアです。ソフトウェアと言いましても、それ自身でプログラムを記述することも可能ですので、ある種のプログラミング言語と解釈していただいても構いません。インタープリタ型であることから、Basic や Perl のように、初心者にも比較的簡単に使い始めることができます。また、様々な研究分野のそれぞれで利用される関数群のライブラリも、Toolbox という Matlab の拡張モジュールとして存在しているため、自分の研究のためのライブラリを自作する必要がありません。ちなみに、東北大学情報シナジーセンターの Matlab には、Toolbox のほとんどが組込まれています。インタープリタ型ですので、これら Toolbox の各関数もソースコードの形を取っています。したがって、実際の処理アルゴリズムを自分の目で追うことも可能です。さらに Matlab の特徴的な点は、行列演算にあります。C 言語をはじめ多くのプログラミング言語では、行列を表現するときに配列を使用し、演算する際も各要素毎の演算で記述する必要があります。しかし、Matlab では、行列 X と行列 Y の積は、X*Y と書けばよいのです。C++ でも、オペレータを利用することで同様なことができますが、C++ がスカラ演算を基にして行列の演算を実現するのに対し、Matlab では全てを行列の演算で実現し、スカラは 1x1 の行列と解釈されます。Matlab の数値演算ソフトウェアとしての利用法については、既に多くの書籍が出版されています [1, 2]。そこで、本稿では、Matlab のもうひとつの機能について紹介して行こうと思います。

Matlab の “肩書” として、「数値解析・ビジュアライゼーションソフトウェア」という文句がしばしば使われています。生産性高くシミュレーションや解析プログラムを作成できるという理由で Matlab を利用されている方も少なくありませんが、私はビジュアライゼーションツールとしてだけでも使う価値があると考えています。計算機の演算能力の飛躍的な躍進や、記録媒体の大容量化・低価格化に伴い、近年の研究は膨大な計算機シミュレーションデータや測定データに基づいて解析を行うスタイルに大きく様変わりしました。これらの莫大なデータから、有意味な情報を人間が取り出すためには、データを可視化することが不可欠です。Matlab は、この可視化を簡単に行うことができます。数値解析ソフトウェアとしての機能とともに、可視化の機能が充実していることが、Matlab を有用なものにしていると思われます。これらを組み合わせることにより、計算途中のデータ状況をグラフで確認しながら計算機シミュレーションを実行することも可能になります。このことは、プログラム開発の高速化に大きく役立つものでしょう。これまで、ネットワーク経由での画像の描画は、時間が掛かることを理由に敬遠されがちでしたが、この問題も、学内ネットワークの高速化により次第に解消されつつあります。

一方、現在の研究者に求められるものは、短期間で良い研究を遂行する能力だけではなく、そこで得られた知見を他の人に伝えるプレゼンテーションの能力も必要です。分かり易いプレ

ゼンテーションをするには、見易いグラフが有効であることは言うまでもありません。初心者でも容易に “それなり” のものが実現できるソフトウェアは、えてして、それ以上のものを求めたときに極めて困難な状況に陥ります。ところが、Matlab のビジュアライゼーション機能は、容易にグラフを描けつつ、それ以上のものを求めたときにも、それに対応できる柔軟性をユーザに残しています。そこで本稿では、Matlab での簡単なグラフの表示方法から、それを自由自在に自分のイメージ通りのグラフに変形する手法について紹介して行きます。

なお、本記事中における使用例のうち、太字で書かれいる部分は Matlab の関数名、斜字体は任意の変数名か (右辺に出現した場合は) 数字を表しています。

2 Matlab で図形を描く

2.1 Matlab の起動

まず、東北大学情報シナジーセンターで Matlab を利用する手順について記しておきます。Matlab を数値演算のためだけに利用するのであれば、どのような計算機環境を利用していても、サーバに接続さえできれば利用することができます。ただし、グラフを描画するとなると事情が異なります。Matlab で作成したグラフを描画するウィンドウを開くために、X-window システムを必要とします。Unix 環境を利用されている方は、まず間違いなく X を利用していると思いますが、Windows や Mac を利用されている方ですと、Astec-X や Exceed、MacX のような X サーバソフトが必要になります。また、接続コマンドも telnet や rlogin ではなく、ssh を利用したほうが確実です。セキュリティ上の理由により、通常の X の通信を遮断しているネットワークが近年増えていますが、ssh を利用することにより、そのような環境下でも X を利用することができます。東北大学情報シナジーセンターのアカウント名が a12345 の場合は、

% ssh a12345@gen.cc.tohoku.ac.jp

と入力します。すると、

% a12345's password:

と表示されますので、a12345 用のパスワードを入力して接続します。プロンプトが現れたら、

% matlab

で起動することができます。一瞬、Matlab のウィンドウが現れて消え、Matlab を起動したウィンドウには、以下のような文字が書き出されます。

```
                    < M A T L A B >
        Copyright 1984-2003 The MathWorks, Inc.
     Version 6.5.1.199709 Release 13 (Service Pack 1)
                         Aug  4 2003
```

```
  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, visit www.mathworks.com.

>>
```

一番下の >> が Matlab のプロンプトになります。ここに Matlab のプログラムを書いて行くと、リターンを押す度にその行が解釈され実行されます。Basic や バッチファイルのように、幾つものコマンドや関数をまとめて実行したい場合には、ファイルにそれを記述することも可能です。その場合は、拡張子を .m にしておきます。そうすると、Matlab のプロンプトに対して、このファイル名から拡張子を除いた部分を入力することにより、ファイルの中身が実行されます。Matlab を終了して、元のシェルに戻りたいときは、

```
>> quit
```

を入力します。

2.2 折線グラフの描画

基本的な 2 次元の折線グラフを例にとり、実際にグラフを描いてみましょう。例えば、表 1 に示す測定データをグラフ化することを考えます。最も簡単な描画方法は、このデータを変数

表 1: 測定データ

	3 月	4 月	5 月	6 月	7 月
アップル	0.23	0.56	0.90	0.77	0.43
レモン	0.06	0.24	0.78	0.60	0.38

に代入し、plot 関数を呼ぶことです。

```
y = [0.23 0.06 ; 0.56 0.24 ; 0.90 0.78 ; 0.77 0.60 ; 0.43 0.38];
plot( y );
```

この 2 行を実行することで、図 1 に示すグラフが得られます。このように 2 行 (変数に代入する必要は必ずしもないので 1 行でも可能) で簡単にグラフを描画することが可能ですが、プレゼンテーションで使用するには、このままでは不十分です。そこで、この図を操作して、プレゼンテーションで使用するにふさわしいグラフに変形して行くことにしましょう。

3 描画構造の基本

Matlab におけるグラフの描画は、最初に元となるグラフをデフォルトの設定に任せて作成し、その後で自分の思ったように変形して行くと意外と簡単に描くことができます。ここで

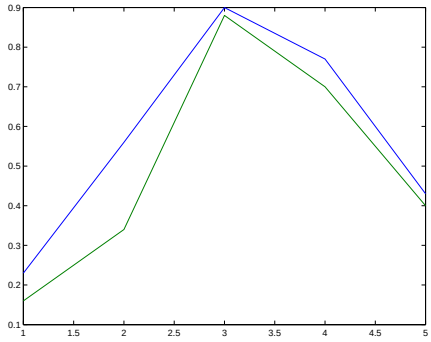

図 1: デフォルトの設定で描画したグラフ

は、Matlab における描画構造について解説します。実践的な使い方のほうに興味があるという方は、この章は省略して 4 章へ飛んでいただいても構いません。

図中のオブジェクトには、オブジェクトハンドルと呼ばれる ID タグが付されます。このオブジェクトハンドルを指定することで、各オブジェクトに対する操作を行うことができます。オブジェクトの現在の特性を表示するには、そのオブジェクトのオブジェクトハンドル (*handle*) を引数にして、

get(*handle*);

を実行します。また、そのオブジェクトのプロパティを変更するときには、オブジェクトハンドルに続けて変更したいプロパティ名と実際にそのプロパティに与える値を引数とし、

set(*handle*, property, *value*);

を実行します。

Matlab では、図 2 に示すような構造でグラフの描画が行われます。座標軸はウインドウ上に描かれるので、ウインドウを親とする子オブジェクトとなります。また、その座標軸上に描かれる曲線は、その座標軸を親とする子オブジェクトとなります。オブジェクトによって、それが保有するプロパティも異なります。子オブジェクトのハンドルは、たいてい 'Children' というプロパティに格納されます。ただし、軸タイトルのような一部のものについては、独立のプロパティにハンドルが格納されます。

基本となる幾つかのハンドルについては、予め変数名が割り当てられています。よく使用するのは gca と gcf のふたつです。それぞれ、座標軸とウィンドウのハンドルに対応しています。試しに座標軸のみが描かれている図に対して、get(gcf) を実行した結果の一部を示すと、次のようになります。

Children = [100.001]

ハンドルである gcf は ID 番号ですので、これ自体も変数になっています。実際、gcf の値は 1 になっていますので、get(gcf) も get(1) も同じ結果を出力します。また、この例では、

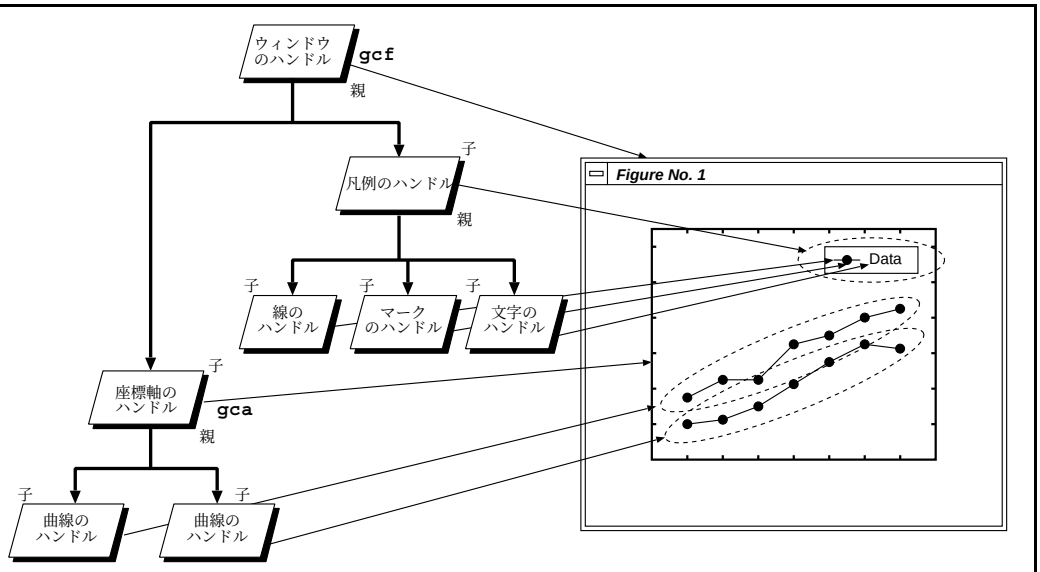

図 2: Matlab における描画構造

'Children' プロパティの中の、100.001 が座標軸ハンドルに対応しています。したがって、gca は実際には 100.001 という数字です。

ここで、ひとつ注意しなければならないのは、Matlab での数字表現の精度です。Matlab の内部での数字の取り扱いは、全て倍精度浮動小数点型になります。したがって、それを表示した場合には、表示可能桁数によって丸めて表示されます。そのため、'Children' プロパティの値が 100.001 と表示されているからといって、get(100.001) が同じ結果を出力するとは限りません。ちなみに、演算結果の表示桁数は、

format long

と入力することにより、増やすことができます。ただし、残念ながらこの表示精度は、プロパティの表示には反映されません。ID 番号の値そのものが必要になることはまずありませんが、必要ならば、

$val = \mathbf{get}(\ handle, \text{'Children'}\ );$

を実行して変数に代入してから、その変数の値を表示することにより、確認することができます。

ソフトウェアは、どのような構造でその処理が行われているのかを理解すれば、たいていの場合、自分の思い通りに扱えるようになります。

4 プロパティの調整

4.1 座標軸

Matlab は、描画しようとしているデータの全てが座標軸内に納まるように軸を形成します。したがって、他とは大きく異なる値を持つ、いわゆる “外れ値” がデータに含まれていると、データの傾向がグラフからは読み取り難くなります。また、折線の全体が座標軸の中に納まってはいるが、なんとなくバランスが悪いということもしばしばあります。このような場合には、座標軸の範囲を次の方法で変更します。

set(gca, ’Xlim’, [*xmin xmax*], ’Ylim’, [*ymin ymax*]);

ここで、*xmin* は x 軸の左端の値、*xmax* は x 軸の右端の値です。同様に、*ymin* は y 軸の下端の値、*ymax* は y 軸の上端の値になります。

また、座標軸の目盛に振られる数字は、任意の文字列に変更可能です。ただし、次の例にあるように、文字列の長さを同じにする必要があります。もし、文字数が少ないものがある場合には、空白を追加して同じ長さに調節しましょう。下記の例では、March と April が 5 文字で一番長いので、他の文字も 5 文字になるように空白を追加します。

set(gca, ’XTickLavel’, [’March’ ; ’April’ ; ’␣May␣’ ; ’June␣’ ; ’July␣’]);

4.2 曲線

測定点に対するマークとして、13 種類のマークが準備されています。ただし、実測によるデータについては測定誤差を含むので、面積のあるマークを使用するように、と学生時代に教わった記憶があります。デフォルトでは、中抜きのマークが使用されるので、

set(*handle*, ’MarkerFaceColor’, *color*);

のようにして、マークの中を塗り潰しておくのが良いでしょう。色については、表 2 に示したもの以外にも、[0 0 0] の黒色から、[1 1 1] の白色まで、RGB 表現を用いることにより自由に指定することが可能です。また、デフォルトのマークの大きさや線の太さは、プレゼンテーションで使用するにはやや小さく細いので、

set(*handle*, ’MarkerSize’, *size*);
set(*handle*, ’LineWidth’, *width*);

を利用して、太く大きくすると見易くなります。これらのプロパティの指定は、上記のように各項目を別々に指定することもできますし、まとめて 1 回で指定することもできます。

set(*handle*, ’MarkerFaceColor’, *color*, ’MarkerSize’, *size*, ’LineWidth’, *width*);

また、以下の例のように、plot 関数の中で指定することも可能です。

plot(*x*, *y*, ’MarkerFaceColor’, *color*, ’MarkerSize’, *size*, ’LineWidth’, *width*);

表 2: 利用可能な色・マーク・線種

色		マーク		線種	
b	青	.	・ (点)	-	実線
g	緑	o	○ (マル)	:	点線
r	赤	x	× (バツ)	-.	鎖線
c	青緑	+	＋ (十字)	–	破線
m	赤紫	*	＊ (アスタリスク)		
y	黄	s	□ (四角形)		
k	黒	d	◇ (菱形)		
w	白	v	▽ (逆三角形)		
		^	△ (三角形)		
		<	◁ (左向き三角形)		
		>	▷ (右向き三角形)		
		p	☆ (五角星形)		
		h	✡ (六角星形)		

4.3 文字

ギリシャ文字や算術記号については、組版ソフトの TEX で利用されているコマンドの幾つかが、Matlab でも利用できるようになっています。利用可能な LATEX コマンドと実際に表記される文字を表 3 にまとめました。また、書体の変更についても、LATEX と同様に以下のコマンドを利用して指定することが可能です。

\bf – **bold face**
\it – *italic*
\rm – roman

この他にも、上付き文字 (^) や下付き文字 (_) を表示するときに、LATEX と同じコマンドが利用できます。

4.4 背景

Matlab で描画したグラフの背景は、デフォルトでは白地になります。しかし、PowerPoint などのプレゼンテーションソフトウェアでは多彩なテンプレートが準備されており、背景が白地でないものも少なくありません。この場合は、Matlab で描画したグラフをそのまま取り込むと、そこの背景だけが白地になってしまいます。それを逆手に取って、うまくグラフを浮き立たせることもできますが、なかなか容易ではありません。Matlab でグラフを描画するときに背景色も変更するには、次のようにします。

set(gca, 'BackGroundColor', *color*); ← 座標軸の内側の背景色変更
set(gcf, 'Color', *color*); ← 座標軸の外側の背景色変更

表 3: Matlab で利用可能な LaTeX コマンド

ギリシャ文字 (小文字)							
\alpha	α	\beta	β	\gamma	γ	\delta	δ
\epsilon	ϵ	\zeta	ζ	\eta	η	\theta	θ
\iota	ι	\kappa	κ	\lambda	λ	\mu	μ
\nu	ν	\xi	ξ	\o	$\o$	\pi	π
\rho	ρ	\sigma	σ	\tau	τ	\upsilon	υ
\phi	ϕ	\chi	χ	\psi	ψ	\omega	ω
ギリシャ文字 (変体文字)							
\vartheta	ϑ	\vapi	ϖ	\varsigma	ς		
ギリシャ文字 (大文字)							
\Gamma	Γ	\Delta	Δ	\Theta	Θ	\Lanmbda	Λ
\Xi	Ξ	\Pi	Π	\Sigma	Σ	\Upsilon	Υ
\Phi	Φ	\Psi	Ψ	\Omega	Ω		
演算子							
\equiv	$\equiv$	\sim	$\sim$	\cong	$\cong$	\approx	$\approx$
\propto	$\propto$	\neq	$\neq$	\mid	$\mid$	\perp	$\perp$
\rfloor	$\rfloor$	\lfloor	$\lfloor$	\rceil	$\rceil$	\lceil	$\lceil$
\wedge	$\wedge$	\vee	$\vee$	\cdot	$\cdot$	\circ	$\circ$
\bullet	$\bullet$	\otimes	$\otimes$	\oplun	$\oplus$	\oslash	$\oslash$
\cap	$\cap$	\cup	$\cup$	\subset	$\subset$	\supset	$\supset$
\subseteq	$\subseteq$	\supseteq	$\supseteq$	\in	$\in$	\ni	$\ni$
\pm	$\pm$	\times	$\times$	\div	$\div$	\langle	$\langle$
\rangle	$\rangle$	\leq	$\leq$	\geq	$\geq$		
その他の記号							
\Re	$\Re$	\Im	$\Im$	\aleph	$\aleph$	\wp	$\wp$
\forall	$\forall$	\exists	$\exists$	\infty	∞	\partial	∂
\int	$\int$	\nabla	∇	\neg	$\neg$	\surd	$\surd$
\ldots	$\ldots$	\clubsuit	$\clubsuit$	\heartsuit	$\heartsuit$	\spadesuit	$\spadesuit$
\diamondsuit	$\diamondsuit$	\leftarrow	$\leftarrow$	\uparrow	$\uparrow$	\rightarrow	$\rightarrow$
\downarrow	$\downarrow$	\leftrightarrow	$\leftrightarrow$	\prime	$\prime$	\O	$\O$
\copyright	©						

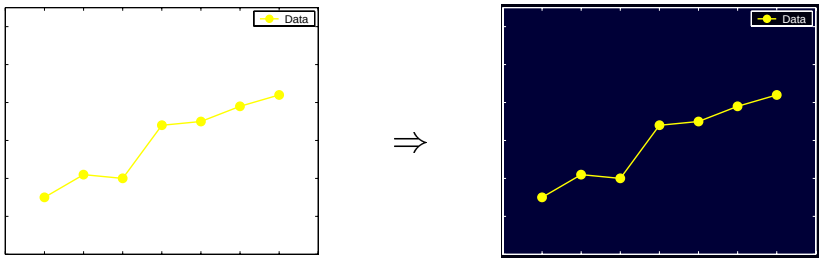

図 3: 背景色の変更

座標軸の内側は gca のハンドルを用い、座標軸の外側は gcf のハンドルを用いていることに注意してください。また、このようにして描画したグラフを、PS ファイルや JPEG ファイルへ出力する際に、デフォルトでは背景を白に変更してしまう設定が Matlab ではされています。そこで、画像ファイルへの出力後も、画面の背景色を保持するために、

set(gcf, 'InvertHardCopy', 'off');

を実行して、この機能を停止させておく必要があります。出力例を図 3 に示します。

4.5 凡例

どの線が何のデータであるかは、キャプションにも書けますが、凡例で示したほうが分かり易くなります。Matlab で凡例を書くためには、以下の関数を利用します。

lgnd = **legend**(*text1*, *text2*)

この関数の戻り値は、凡例を描画するための箱のハンドルになります。凡例は、座標軸の外側に置かれることもありますので、このハンドルは座標軸のハンドルの子供ではなく、ウィンドウのハンドルの直接の子供になります。実際の凡例の線や文字は、その子供として作成されるので、これらを思い通りに変更するには、ここで作成した凡例ボックスの子供のハンドルを、下記のようにして取得しなければなりません。

children = **get**(*lgnd*, 'Children')

4.6 位置

文字を大きくすると、軸タイトルがウィンドウからはみ出してしまうことがよくあります。この場合、このまま画像ファイルに出力してしまうと、ウィンドウからはみ出している部分が切れてしまいます。この問題を解決するには、ウィンドウ上の座標軸の位置をずらすのが有効です。

pst = **get**(gca, 'Position');

は、座標軸の位置情報に関する 4 つの要素からなる配列を返します。したがって、先ず、$pst(1)$ あるいは $pst(2)$ の値を増減して調節し、それでも納まらない場合には、$pst(3)$ あるいは $pst(4)$ の値を増減して調節すれば、ウィンドウの中に全てが入るようになります。

$pst(1)$：x 軸の左端の位置

$pst(2)$：y 軸の下端位置

$pst(3)$：x 軸の長さ

$pst(4)$：y 軸の長さ

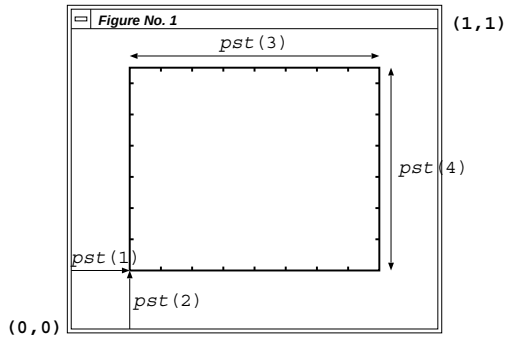

図 4: ウィンドウ内での座標軸の位置調整

4.7 ファイルへの出力

描画したグラフをプレゼンテーションソフトウェアで使用するには、一度、Matlab から画像ファイルに出力した後に、そのファイルを読み込むことになります。ハンドアウトを Postscript プリンタで出力することなどを考えると、Encapslated Postscript 形式 (.eps) で保存したいところです。ところが、よく利用される PowerPoint では Postscript をプレビューできないため、適切な位置に配置できないばかりでなく、プレゼンテーションでは役に立ちません。この問題は、TIFF 形式を含んだ Postscript ファイルを生成することにより解決できます。Matlab で、TIFF 形式を含んだ Encapslated Postscript 形式で出力する場合には、次のように入力します。

print -depsc -tiff filename.eps

この場合は、filename.eps というファイル名のファイルへ出力されます。ファイル名を変数で与えたい場合には、次のようにします。

eval(['print␣-depsc␣-tiff␣'␣*varfile*]);

print コマンドのオンラインマニュアルにも書かれていますが、デフォルトの TIFF 形式の解像度は 72 DPI に固定されています。この解像度ですと、プロジェクターに投影したときに、ギザギザ感が残ってしまいます。解像度は、

MATLAB_TOP_DIR/toolbox/matlab/graphics/private/render.m

内で指定されています。ここで、MATLAB_TOP_DIR は、Matlab がインストールされている

トップディレクトリです。例えば、東北大学情報シナジーセンターの場合は、/usr/ap/matlab6p5になります。このファイル内の数字を書き換えることで、より高解像度の TIFF 形式データを含ませることも可能です。研究室や個人で Matlab を導入されているところは、試してみると良いかもしれません。このファイルを個人のディレクトリにコピーして編集し、そのディレクトリが上記のディレクトリよりも先に探索されるようにする方法もあります。

5 サンプルプログラム

これまでに説明した方法を用いて、図 1 のグラフをプレゼンテーションに使用可能なグラフに変更するプログラムと、そのプログラムを実行した結果を次に示します。

```
  % 基本図形の描画
y = [0.23 0.06 ; 0.56 0.24 ; 0.9 0.78 ; 0.77 0.6 ; 0.43 0.38];
y = y * 100;
hndl = plot( y, 'o-', 'MarkerSize', 10, 'LineWidth', 2 );
  % 凡例の描画とフォントサイズの設定
lgnd = legend( 'Apple', 'Lemon' );
set( lgnd, 'FontSize', 16 );
  % 折線の色変更
set( hndl(1), 'Color', 'r', 'MarkerFaceColor', 'r' );
set( hndl(2), 'Color', [1 0.8 0.3], 'MarkerFaceColor', [1 0.8 0.3] );
  % 座標軸の範囲、フォントサイズ、目盛の設定
set( gca, 'Xlim', [0 6], 'Ylim', [0 100] );
set( gca, 'FontSize', 18 );
set( gca, 'XTickLabel', ['␣␣␣␣␣' ; 'March' ; 'April' ; '␣May␣' ; 'June␣' ; 'July␣'
   ; '␣␣␣␣␣'] );
  % 軸タイトルの描画
xlabel( 'Month' );
ylabel( 'Observed data (\times 10^{-2})' );
  % 座標軸と背景の色設定
set( gca, 'Xcolor', 'w' );
set( gca, 'Ycolor', 'w' );
set( gca, 'Color', [0 0 55]/255 );
set( gcf, 'Color', [0 0 15]/255 );
  % 位置の調整
pst = get( gca, 'Position' );
pst(1) = pst(1) + 0.02;
pst(2) = pst(2) + 0.02;
set( gca, 'Position', pst );
  % 色反転出力抑止とファイルへの出力
```

```
set( gcf, 'InvertHardCopy', 'off' );
print -deps -tiff filename.eps
```

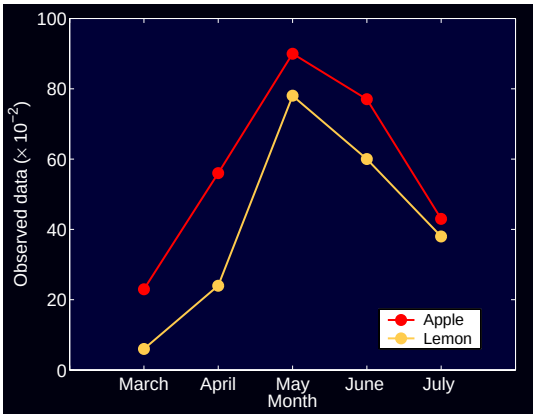

図 5: 本記事で紹介した方法で書き直したグラフ

6　おわりに

今回紹介した方法は、私が Matlab でグラフを描画する際に通常使っているアプローチです。これまで Matlab を利用する中で経験的に習得したものですので、きちんとマニュアルを読めば、もっと簡単な方法があるかもしれません。Matlab は、オンラインヘルプも充実しており、help コマンドで参照できます。みなさん自身でも、ここで紹介されなかった様々な技法を見付け出し、活用してください。本文を参考に、大切な研究成果を印象的なグラフに仕上げて、研究会や学会で発表していただければうれしく思います。

参考文献

[1] 樋口龍雄、川又政征、「MATLAB 対応 ディジタル信号処理」、昭晃堂、2000 年。

[2] 上坂吉則、「MATLAB プログラミング入門」、牧野書店、2000 年。