MATLAB 入門

平野拓一

1. MATLAB と他のプログラミング言語

MATLAB は MATrix LABoratory の略だけあって、行列演算を得意とする。FORTRAN ほど面倒でなく、LINPACK などを使えるように開発された。FORTRAN と同様に科学技術計算を目的としているが、FORTRAN と比べると非常に高水準（ヒューマンフレンドリー）な言語である。GNU プロジェクトで開発されたフリーソフトウェア Octave は MATLAB と互換性がある。

同様の目的のソフトウェアは他にも Mathematica, Maple, Mathcad などがある。特に、Mathematica は記号数式処理（例えば、文字変数の表現で不定積分を行うなど）が得意である。Mathematica は厳密な記号計算や高精度な計算が得意であるが、精度は普通の倍精度や単精度でよいから高速にしたいというような目的には向かない。

代表的なプログラミング言語の年表

開発年	言語	開発者
1946: ENIAC 1947: 点接触型トランジスタ 1948: 接合型トランジスタ		
1954	FORTRAN	ジョン・バッカス(IBM)
1958	ALGOL (欧州が FORTRAN に対抗)	
1958-1959: 半導体 IC		
1958	LISP (理論から発展, emacs)	ジョン・マッカーシー(MIT)
1959	COBOL (会計事務処理用に統一)	アメリカ国防総省
1964	BASIC (FORTRAN 風で初心者向けに開発)	ジョン・ケメニー、トーマス・カーツ (米国ダートマス大学)
1965: ムーアの法則提唱		
1970	Pascal (教育用に開発, TeX)	ニクラウス・ヴィルト(チューリッヒ工科大学)
1972	C (UNIX 開発用, 1960s: Multics[MIT, GE, Bell Labs.], 1969: UNIX [Bell Labs.])	デニス・リッチー(AT&T ベル研究所)

1970 年代後半	MATLAB (Linpack を簡単に使用するために開発)	Cleve Moler(米国ニューメキシコ大学)
1983	C++ (C 言語では不足な大規模プログラムの開発に対応するため。オブジェクト指向)	ビャーネ・ストロヴストルップ(AT&T ベル研究所)
1987	Perl (正規表現。インタープリタ。実用性を重視)	ラリー・ウォール
1988	Mathematica (記号数式処理)	スティーブン・ウルフラム(ウルフラム・リサーチ社)
1991	Python (記号数式処理)	ヴァンロッサム
1993	Ruby	まつもとゆきひろ
1995	Java (仮想マシン。中間コード。Java アプレット)	ジェームズ・ゴスリン、ビル・ジョイなど(サン・マイクロシステムズ(現在:オラクル))
1995	PHP (WEB サーバー用。データベースを HTML 化)	ラスマス・ラードフ

各種言語の長所と短所（主観も入っているが）

言語	長所	短所
FORTRAN77	・世界初の高水準言語 ・コンパイラが用意に作れる ・言語で複素数をサポート ・g77 などフリーのコンパイラがある	・コードが汚い
FORTRAN90/95	・言語で複素数をサポート ・C, C++のよい部分を取り入れて、綺麗なプログラムが書けるようになった。	・フリーのコンパイラは見つからない。
BASIC	・（昔は）初心者の入門としてはよかった	・コードが汚い ・基本的にインタープリタ言語なので遅い ・複素数を言語でサポートしていない
C	・高水準言語ではあるが、コンパクトでアセンブリ言語を意識して効率も考えられている。(UNIX を書くための言語	・複素数を言語でサポートしていない

	だったから) ・いろいろなシステムにコンパイラがあるので移植性はかなり高い ・gcc などフリーのコンパイラがある	
C++	・C 言語においてオブジェクト指向（データとその処理ルーチンのカプセル化）を導入したもの ・g++などフリーのコンパイラがある	・複素数を言語でサポートしていない
Java	・Java コンパイラが乗ったシステムでは移植性は高い ・コンパイラ等の開発環境は Oracle が無料で提供している（企業が提供する「無料」は将来を考えると信頼できないが)	・中間コードを出力して仮想マシンで動作するが、ネイティブの機械語に比べると速度が少し犠牲になっている ・複素数を言語でサポートしていない
Mathematica	・記号数式処理が得意 ・数学者の開発によるものなので、一般性、精度に対する徹底的なこだわり、数学的な美しさが感じられる ・逆行列ルーチンなどを言語でサポートしているのでコードがシンプルで読みやすい	・インタープリタなので比較的遅い ・学生以外は高価
MATLAB	・FORTRAN の高速性と高水準言語とインタープリタの長所を組み合わせたもので、人間は扱いやすい ・コードがシンプルで読みやすい ・逆行列ルーチンなどを言語でサポートしているのでコードがシンプルで読みやすい	・FORTRAN よりは遅い（が、無視できる範囲？） ・学生以外は高価

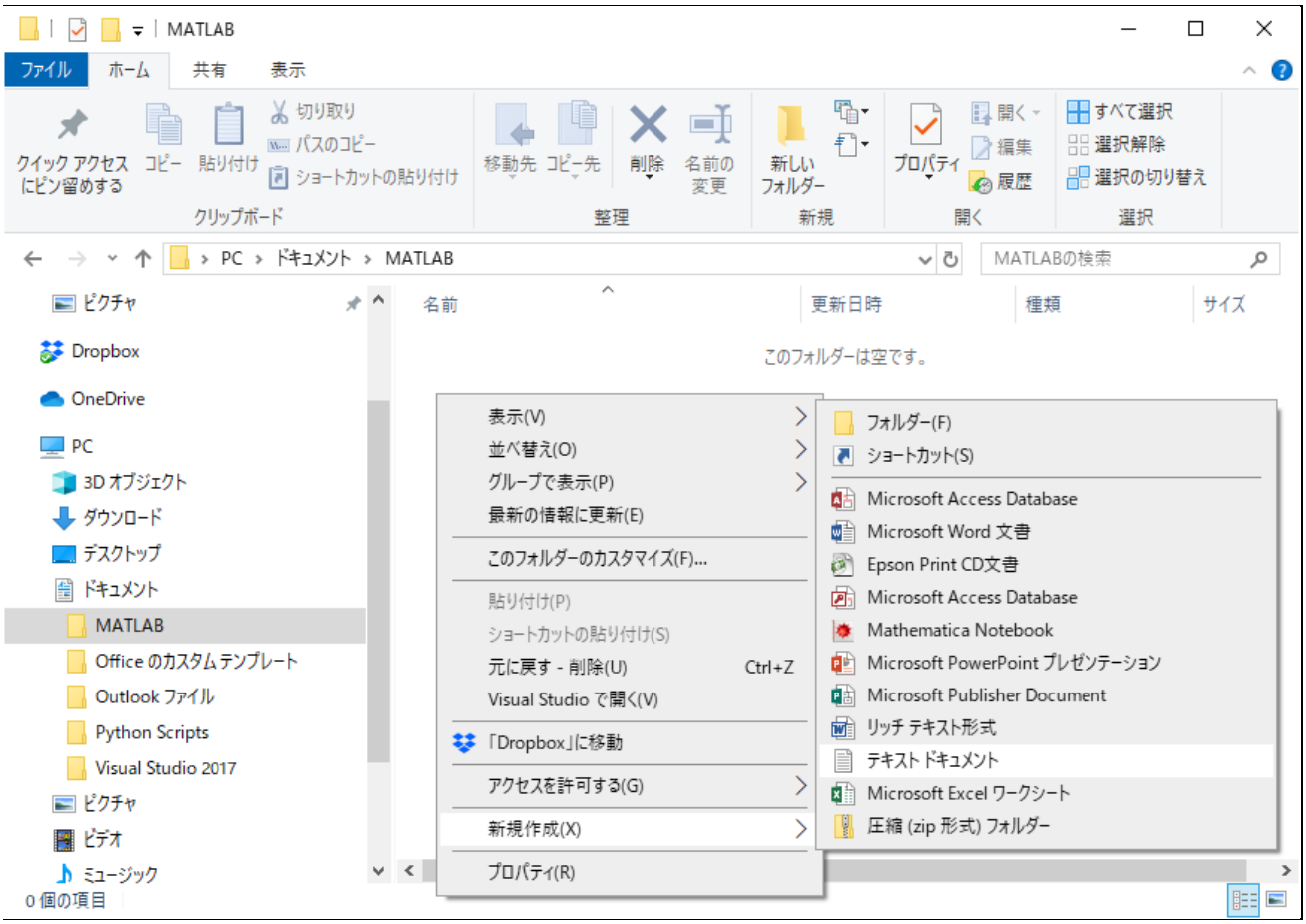
2. MATLAB 使用環境の説明

MATLAB はインタラクティブに使えるが、プログラムを組むまでもないコードでも、いくつかやりたいことをファイルにまとめておくと便利である。MATLAB で、C 言語のソースファイル(***.c)のように使うファイルを M ファイル(***.m)と言い、C 言語と同様に単なるテキストファイルになっている。作成は、いろいろな方法があるが、次のように原始的に作成する方法を紹介する。

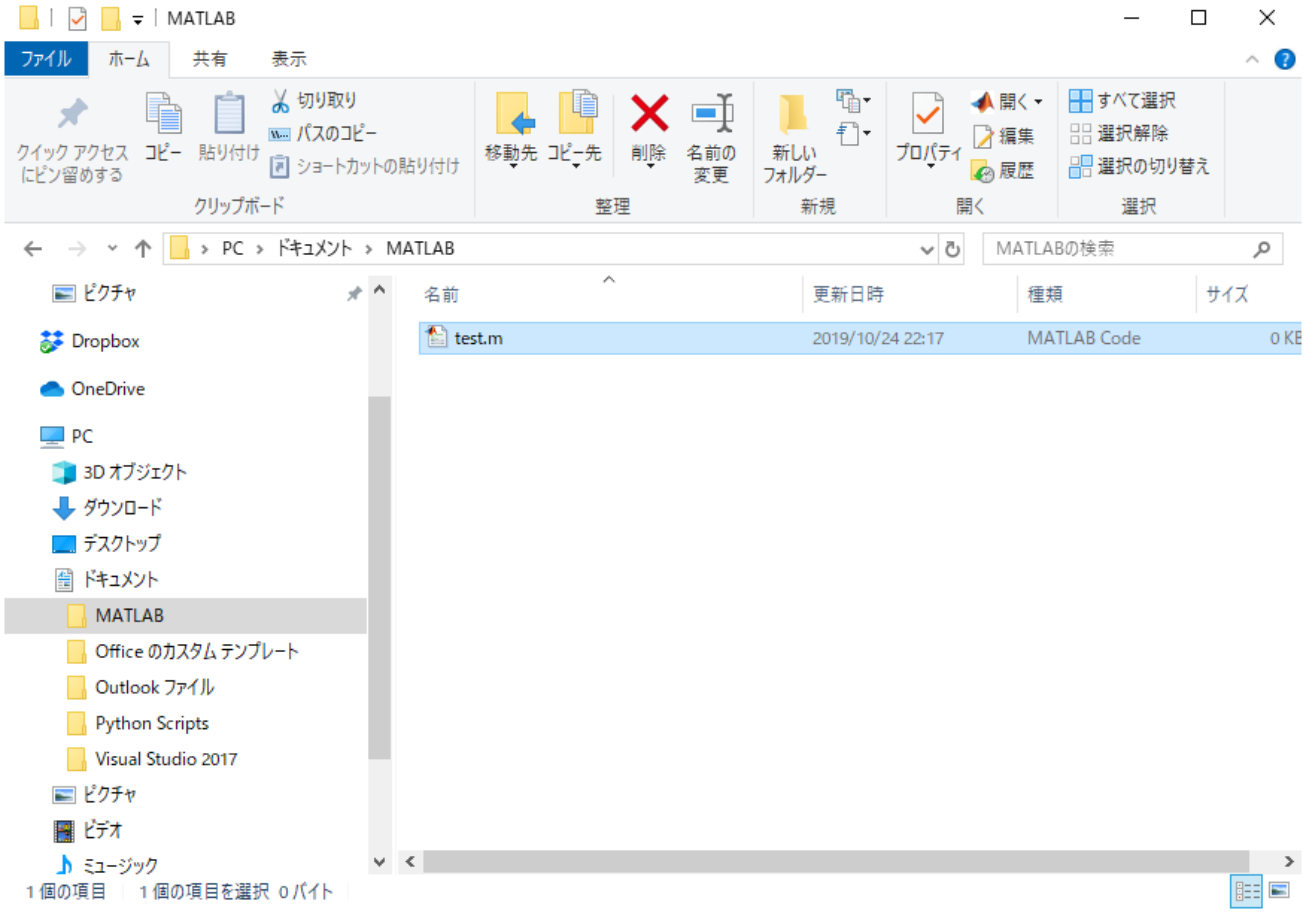
(1) 空の M ファイルの作成

・エクスプローラで M ファイルを作成したいフォルダに移動。

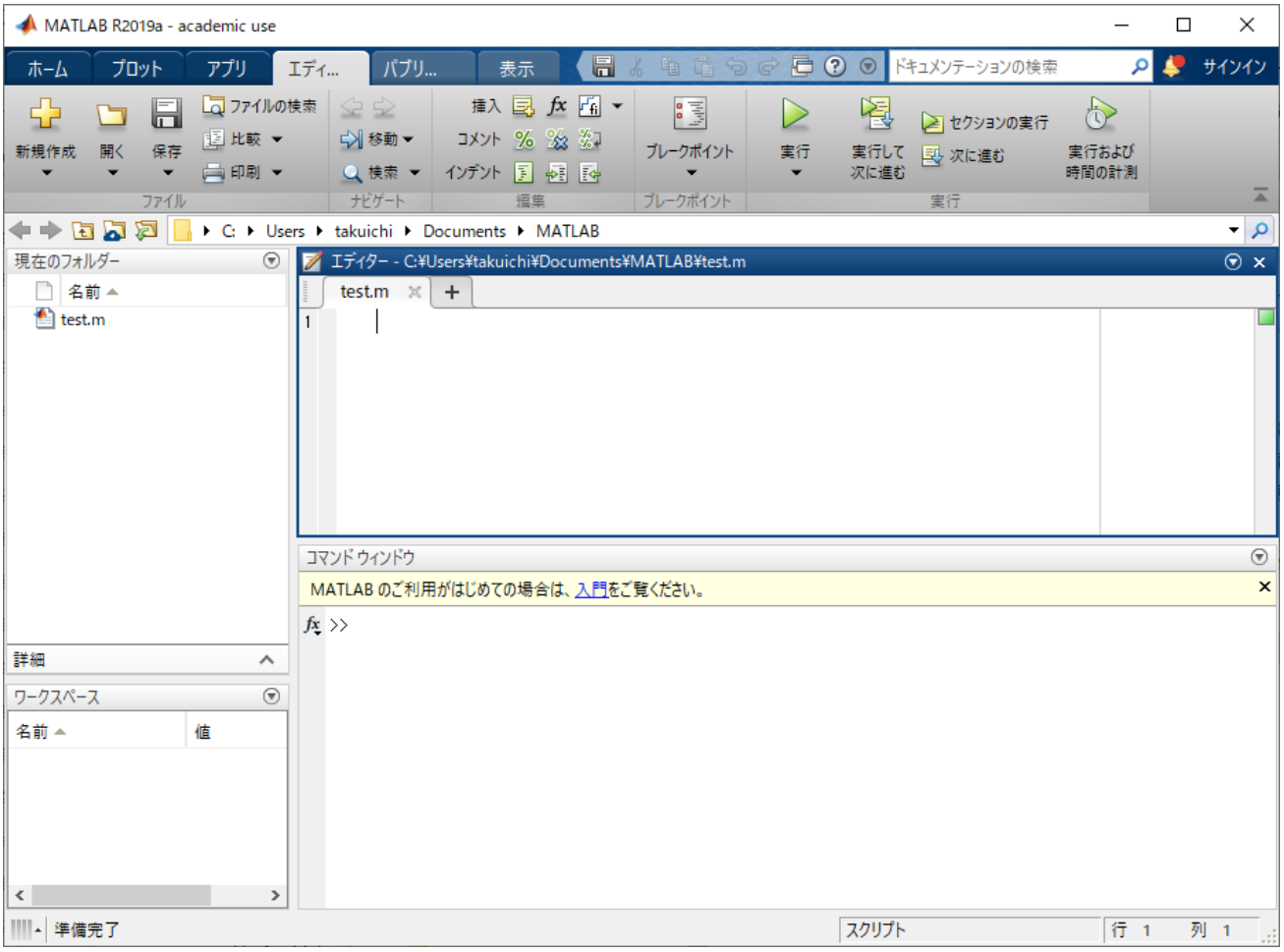
・ファイル表示部分で右クリック、「新規作成→テキスト ドキュメント」と選び、test.m というファイルを作成(test の部分は自分の好きな名前でいい)。



次のように、test.m が作成される。MATLAB がインストールされていれば、拡張子(.m)は MATLAB が扱うように設定されている（Mathematica などの他のソフトと競合しない場合は)。



ここで表示されている test.m をダブルクリックすると、MATLAB が起動する。



上の画面の左上の「現在のフォルダー」枠には、MATLAB が設定している作業フォルダ（上のバーに表示されている）にあるファイルが表示されている。直接 M ファイルをダブルクリックして起動すると、自動的に M ファイルのフォルダに設定される（ここが楽である）。

右上の「エディター」枠には、test.m のファイル内容（今は空）が表示されている。

右下の「コマンドウィンドウ」は MATLAB のインタープリタのコマンドラインとなる。ここでいろいろコマンドを実行する。例えば、test.m という M ファイルは test という関数として認識されるようになっているので、「>>test↵」と入力すると、test.m の内容が上から実行されることになる。

使わなくても済むのであるが、左下の「ワークスペース」には使用している変数の値や定義した関数の名前などが表示されるようになっている。

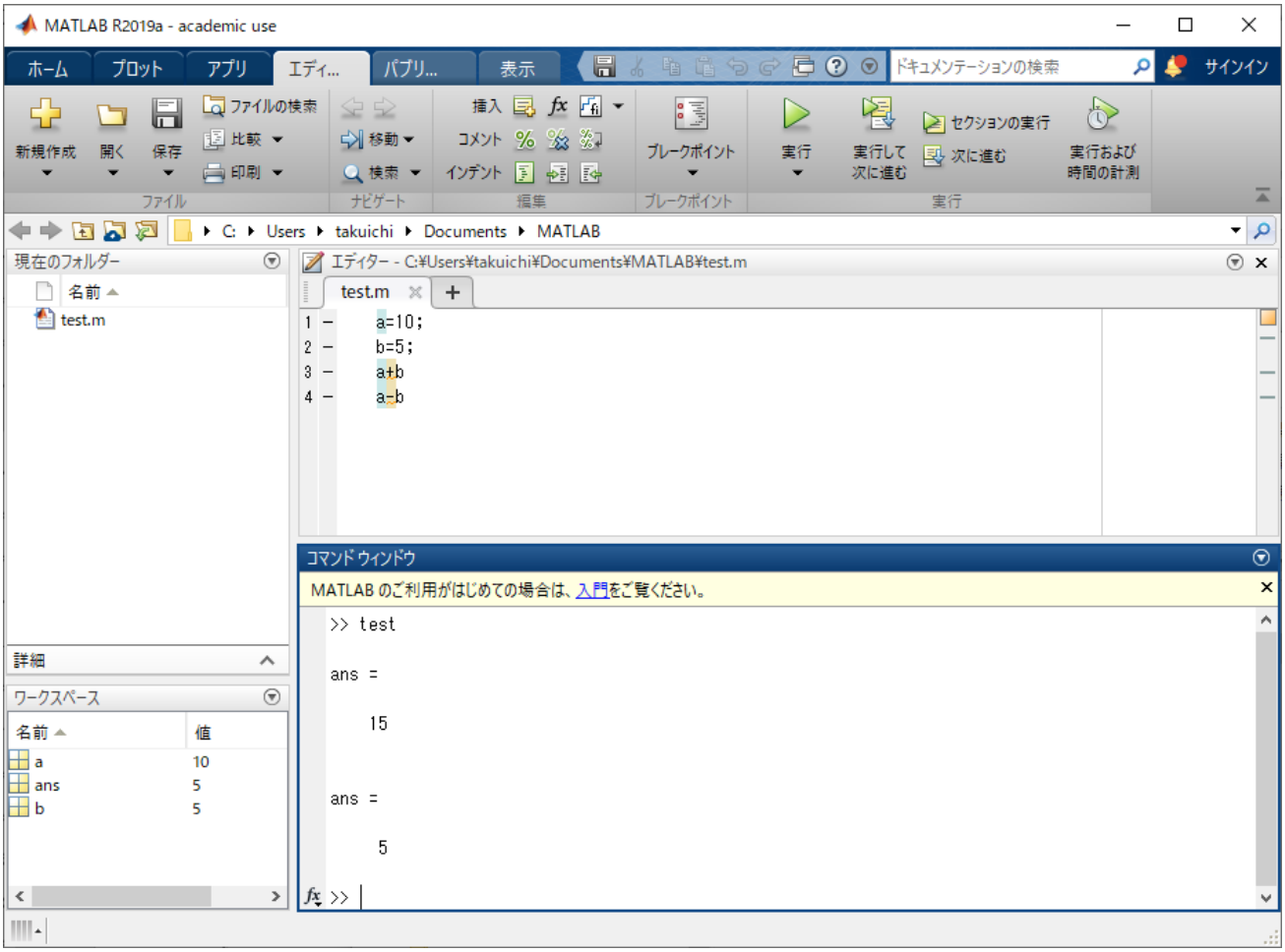
例えば、「エディター」で test.m に

```
a=10;
b=5;
a+b
a-b
```

と入力して保存し、「コマンドウィンドウ」で「>>test↵」と入力すると下のように表示され

る。または、M ファイルを最初から全部実行するためには、上の 

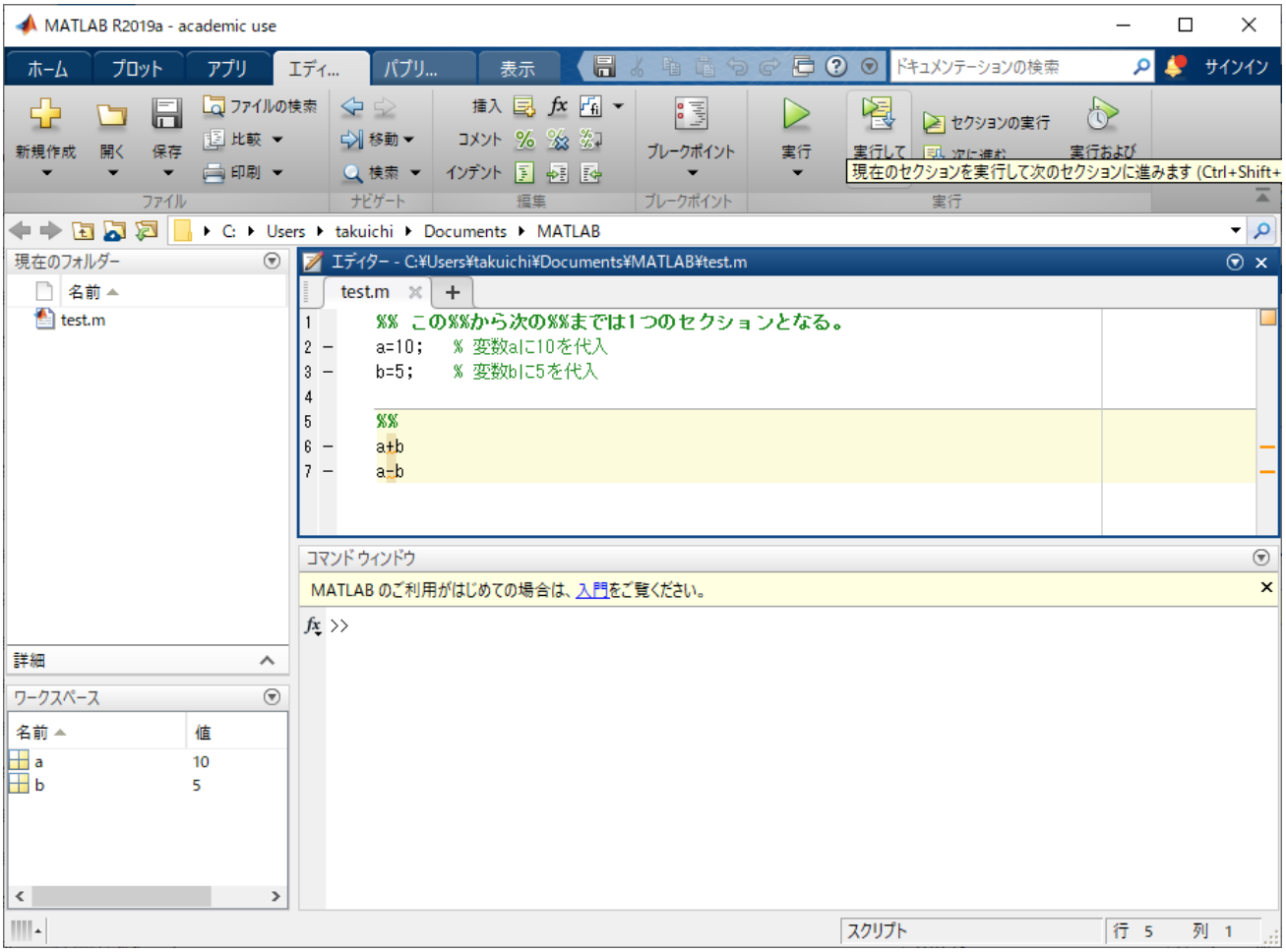
をクリックしてもよい。



コメントアウトとセクション

M ファイル内で、%の後、改行まではコメントアウト（実行命令ではなく、ユーザーのメモ書き）となる。また、行頭に%%があるとセクション区切りとなり、次の行頭に%%がある行までの間が 1 つのセクションという単位になる。MATLAB では、上の 

を利用して、セクション毎に実行することも可能となる。少しずつプログラムを組み上げるときに便利な機能ではある（無くても問題ないが）。


MATLAB R2019a - academic use
ホーム
プロット
アプリ
エディ...
パブリ...
表示
ドキュメンテーションの検索
サインイン
新規作成
開く
保存
ファイルの検索
比較
印刷
移動
検索
挿入
コメント
インデント
ブレークポイント
実行
実行して
セクションの実行
実行および
現在のセクションを実行して次のセクションに進みます (Ctrl+Shift+
ファイル
ナビゲート
編集
ブレークポイント
実行
C: ▸ Users ▸ takuichi ▸ Documents ▸ MATLAB
現在のフォルダー
名前
test.m
エディター - C:¥Users¥takuichi¥Documents¥MATLAB¥test.m
test.m
1 %% このセクションから次のセクションまでは1つのセクションとなる。
2 a=10; % 変数aに10を代入
3 b=5; % 変数bに5を代入
4
5 %%
6 a+b
7 a-b
コマンド ウィンドウ
MATLAB のご利用がはじめての場合は、入門をご覧ください。
>>
詳細
ワークスペース
名前
値
a 10
b 5
スクリプト
行 5
列 1

3. MATLAB 言語（機能）の説明

■行列

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

A =

     1     2
     3     4

>> b=[5;6]

b =

     5
     6
```

行列 A と列ベクトル b を定義する。

```
>> A*A

ans =

     7    10
    15    22
```

行列の掛け算

```
>> A.*A

ans =

     1     4
     9    16
```

「.*」は要素同士の掛け算。「.*」はスカラーの掛け算の演算子としても使われる。つまり、MATLAB ではデフォルトで行列を扱うようにできている。

```
>> A^3

ans =

    37    54
    81   118

>> A*A*A

ans =

    37    54
    81   118
```

```
>> c=[1;2;3]

c =

     1
```

```
     2
     3

>> A*c
??? エラー ==> mtimes
内部行列の次元は一致しなければなりません。
```

当然、行列同士の掛け算は m×n と n×1 でないとならない。エラーが出るのでよい。

```
>> C=zeros(3)

C =

     0     0     0
     0     0     0
     0     0     0

>> C(1,2)=2

C =

     0     2     0
     0     0     0
     0     0     0
```

全要素が 0 の正方行列を生成する。また、要素に値を代入する。

```
>> zeros(3,1)

ans =

     0
     0
     0
```

全要素が 0 のベクトルを生成する。

```
>> inv(A)

ans =

   -2.0000    1.0000
    1.5000   -0.5000

>> inv(A)*A

ans =

    1.0000         0
    0.0000    1.0000
```

行列 A の逆行列を求める。逆行列の計算は行列の次元を n とすると $O(n^3)$である（かなり効率が悪い）。

```
>> inv(A)*b

ans =

   -4.0000
    4.5000
```

A **x**=**b** の x を求める。

```
>> linsolve(A,b)

ans =

   -4.0000
    4.5000
```

同じく、A **x**=**b** の **x** を求める。逆行列直接求めているわけではないので、上の方法よりも効率がよいかもしれない。次の記法も同様。

```
>> A ¥ b

ans =

   -4.0000
    4.5000
```

¥は US キーボードではバックスラッシュになる。

```
>> eig(A)

ans =

   -0.3723
    5.3723

>> [P, D]=eig(A)

P =

   -0.8246   -0.4160
    0.5658   -0.9094


D =

   -0.3723         0
         0    5.3723
```

固有値問題 A **x** = λ **x** の固有値 λ とそれに対応する固有ベクトル **x** を求める。

```
>> inv(P)*A*P

ans =

   -0.3723    0.0000
         0    5.3723
```

行列が対角化されていることが確認できる。

```
>> v1=[1;2];
>> v2=[3;4];
>> dot(v1,v2)

ans =

    11
```

ベクトルの内積。

```
>> v1=[1;2;3];
>> v2=[4;5;6];
>> cross(v1,v2)
```

```
ans =

    -3
     6
    -3
```

ベクトルの外積。

他にも、行列、ベクトルの操作として

- svd: 特異解
- transpose: 転置

などがある

■複素数、円周率

```
>> c1=complex(1,2); c2=complex(2,3);
(>> c1=1+j*2; c2=2+j*3)
>> c1+c2

ans =

   3.0000 + 5.0000i

>> c1*c2

ans =

  -4.0000 + 7.0000i
```

複素数の扱い。

```
>> pi

ans =

    3.1416
```

円周率。

```
>> 1e-3.*1000

ans =

     1
```

指数表現。

■スクリプト（M ファイル）

上のようなコマンド列をまとめて実行したい場合、".m"のテキストファイルにコマンド列を書いて実行すると便利である。

```
>> pwd

ans =

C:¥Users¥hira¥Documents¥MATLAB
```

上のように”pwd”コマンド（UNIX と類似）で表示されるフォルダに下のように記述したテキストファイルを test.m というファイル名で保存する。なお、UNIX のシェルと同様に cd (change directory)コマンドで作業フォルダは移動できる。

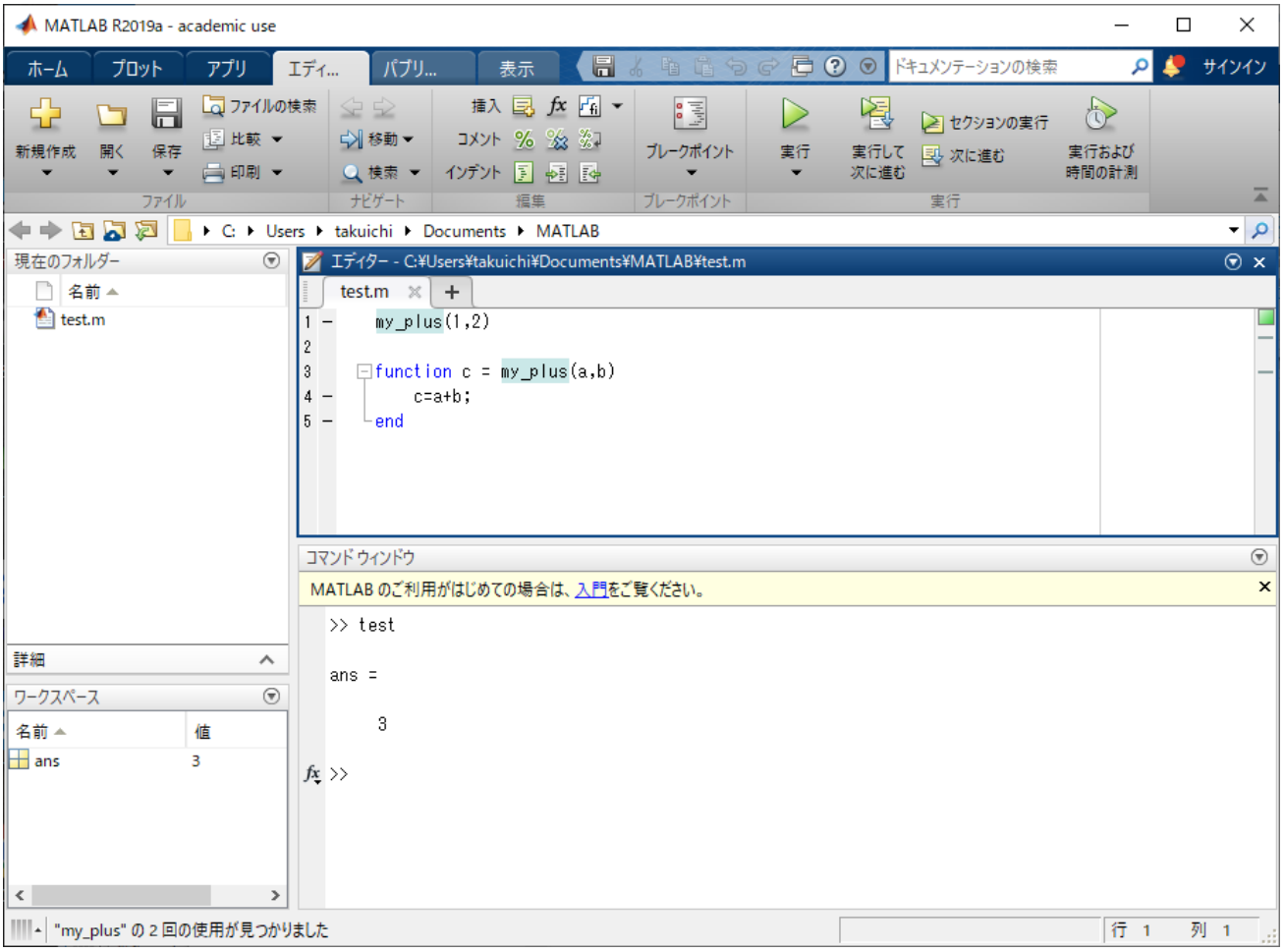
```
A=[1,2; 3,4];
b=[5;6];
inv(A)
inv(A)*A
```

上の内容をメモ帳などで test.m というファイル名で作業フォルダに保存し、以下のように”.m”を抜かして”test”と入力するとバッチファイル的に実行できる。

```
>> test

ans =

   -2.0000    1.0000
    1.5000   -0.5000


ans =

    1.0000         0
    0.0000    1.0000
```

■関数の定義

関数定義は M ファイルの下部で行う。



M ファイルをコマンドラインから呼び出す 1 つの関数とすることもできる。

```
function y = func1(x)
  y = x.^2;
end
```

上のように記述したテキストファイルを作業フォルダに func1.m というファイル名で保存する（関数名と M ファイルの名前は同じでなければならない)。

関数の呼び出しは次のように行う。

```
>> func1(2)

ans =

     4
```

複数の結果あるいはベクトルを返す関数は次のように定義する。

```
function [fx,fy,fz] = func2(x,y,z)
  r=sqrt(x.^2+y.^2+z.^2);
  f=1./(r.^2);
  fx=f.*(x./r);
  fy=f.*(y./r);
  fz=f.*(z./r);
end
```

関数の呼び出しは次のように行う。

```
>> [fx,fy,fz]=func2(1,2,3)

fx =

    0.0191


fy =

    0.0382


fz =

    0.0573
```

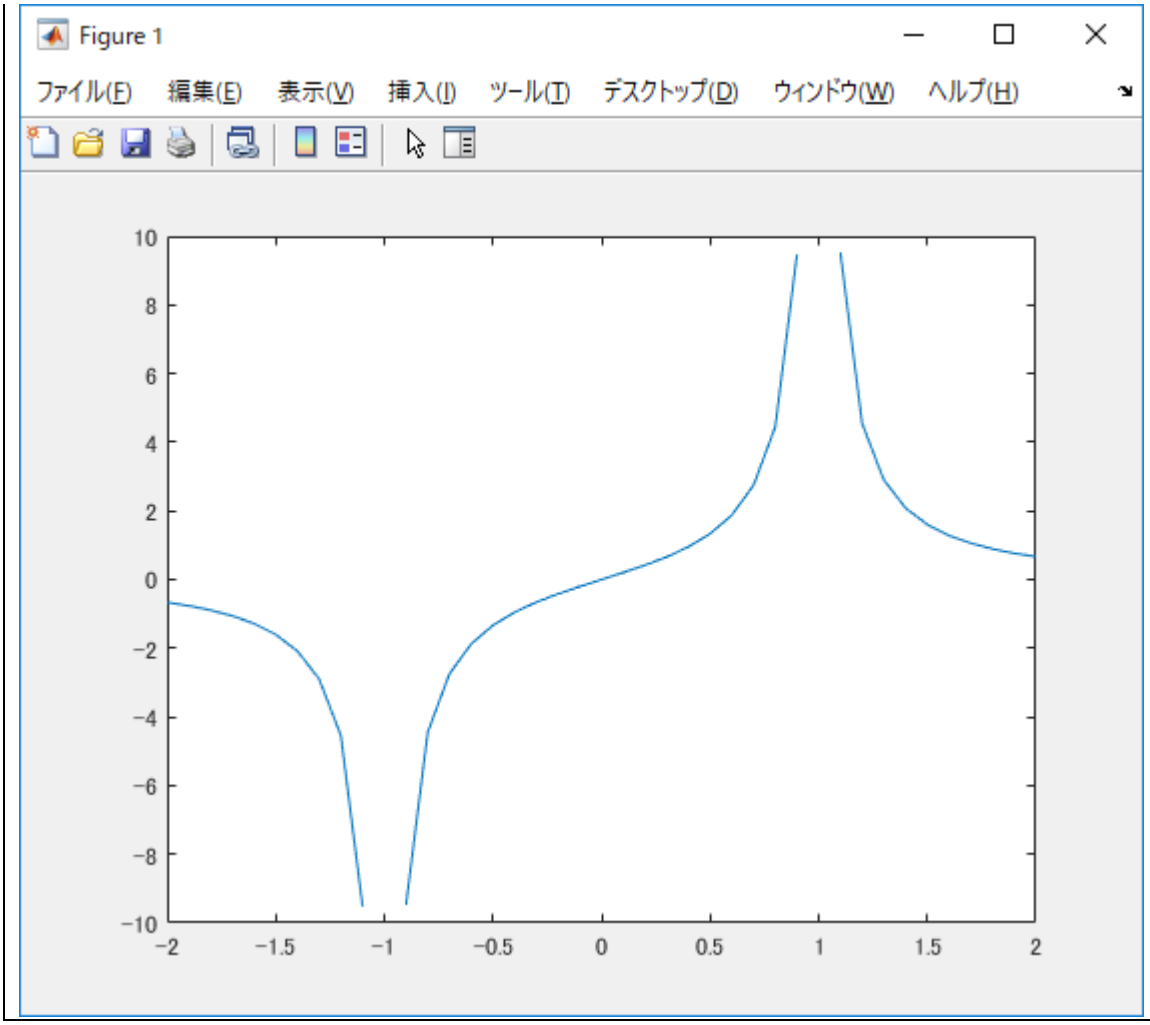
コマンドラインから関数を定義するには次のようにする。M ファイルの関数定義記述はコマンドラインでは使えない。コマンドラインからは次のように無名関数という定義方法がある。ただし、１行しか記述できない。

```
func1=@(x) x.^2;
```

このような、関数定義が面倒（ファイルを分けないとならない）問題の対処法としては、仕事内容に対して M ファイルを作ってしまい、次のようにそのファイル内に使う関数を記述していく方法がある（plot コマンドは後のページを参照）。

```
function test01
    q1=-1.;
    x1=-1.;
    y1=0.;
    z1=0.;

    q2=1.;
    x2=1.;
    y2=0.;
    z2=0.;

    x=-2:0.1:2;
    y=potential(q1,radius(x-x1,y1,z1))+potential(q2,radius(x-x2,y2,z2));
    plot(x,y);
end

function r = radius (x,y,z)
    r = sqrt(x.^2+y.^2+z.^2);
end

function v = potential (q,r)
    v = q./r;
end
```

```
>> test01
```



■数値積分

```
>> unit_circ =@(x) sqrt(1.-x.^2);
```

単位円の関数 unit_circ を定義して数値積分している。

```
>> integral(unit_circ,-1,1)

ans =

    1.5708
```

trapz(台形公式), quad(シンプソンの公式)もある。
他にも、２重数値積分、３重数値積分等の関数が用意されている。

■計算時間とメモリ使用量の測定

example.m に次のように書く。

```
function example
    n=2000;              % 行列方程式の次元

    tic;                 % 時間測定開始
    A=rand(n);           % nxnの乱数を要素にもつ行列を生成
    b=rand(n,1);         % 乱数を要素にもつ列ベクトルを生成
    %x=linsolve(A,b);     % A x=bを解く
```

```
    x=A¥b;              % A x=bを解く
    toc;                % 時間測定終了

    whos;               % メモリ使用量を表示
end
```

すると、example 関数を実行すると次のようにかかった時間と使用メモリを容量が表示される。

```
>> example
経過時間は 0.456801 秒です。
  Name          Size                Bytes  Class     Attributes

  A          2000x2000            32000000  double
  b          2000x1                  16000  double
  n             1x1                      8  double
  x          2000x1                  16000  double
```

■特殊関数

```
>> besselj(1,2+j*3)

ans =

   3.7807 - 0.8128i
```

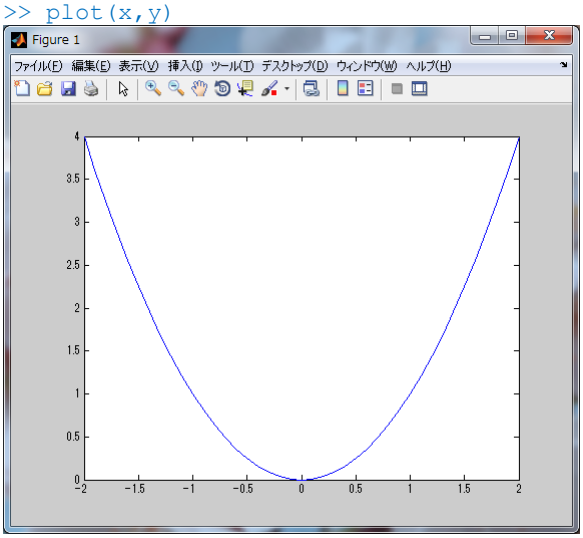
ベッセル関数の呼び出し例。他にも様々な特殊関数が用意されている。

airy	エアリー関数
besselh	第 3 種ベッセル関数 (ハンケル関数)
besseli	第 1 種変形ベッセル関数
besselj	第 1 種ベッセル関数
besselk	第 2 種変形ベッセル関数
bessely	第 2 種ベッセル関数
beta	ベータ関数
betainc	不完全ベータ関数
betaincinv	ベータ逆累積分布関数
betaln	ベータ関数の対数
ellipj	ヤコビ楕円関数
ellipke	第 1 種と第 2 種の完全楕円積分
erf	誤差関数
erfc	相補誤差関数
erfcinv	逆相補誤差関数
erfcx	スケーリング相補誤差関数

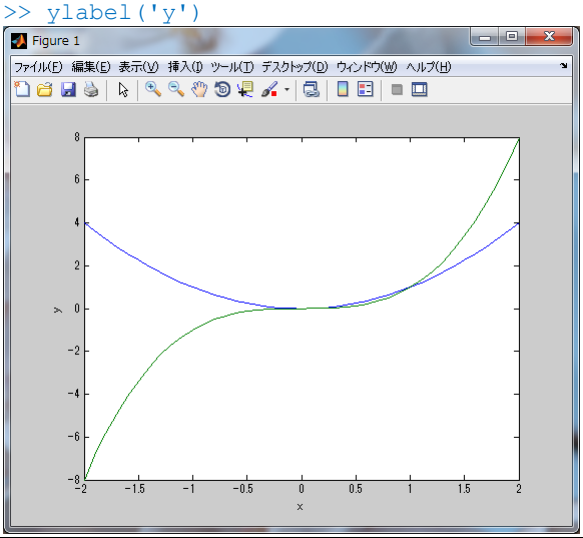
erfinv	逆誤差関数
expint	指数積分
gamma	ガンマ関数
gammainc	不完全ガンマ関数
gammaincinv	逆不完全関数
gammaln	関数 gamma の対数
legendre	随伴関数 Legendre
psi	Psi (ポリガンマ) 関数

■グラフィック

```
>> x=-2:0.1:2;
>> y=func1(x);
>> plot(x,y)
```

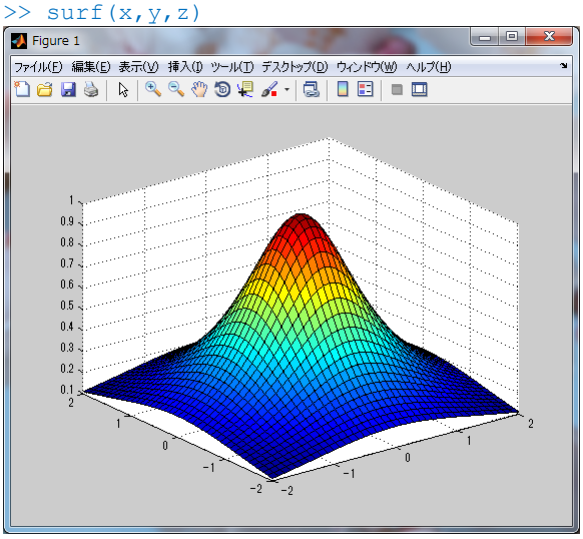



```
>> y2=x.^3;
>> plot(x,y,x,y2)
>> xlabel('x')
>> ylabel('y')
```

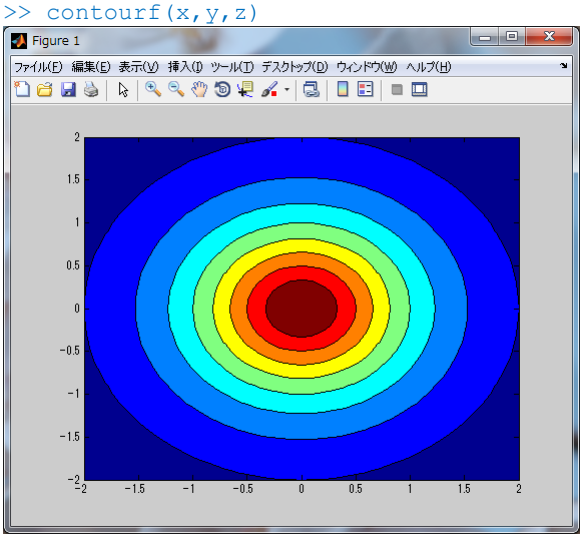


“-2:0.1:2”は-2 から 2 まで 0.1 ずつ増加させたときの行ベクトルの生成を行う。グラフはいろいろカスタマイズできるが、その方法は Mathworks のホームページマニュアルを参照。

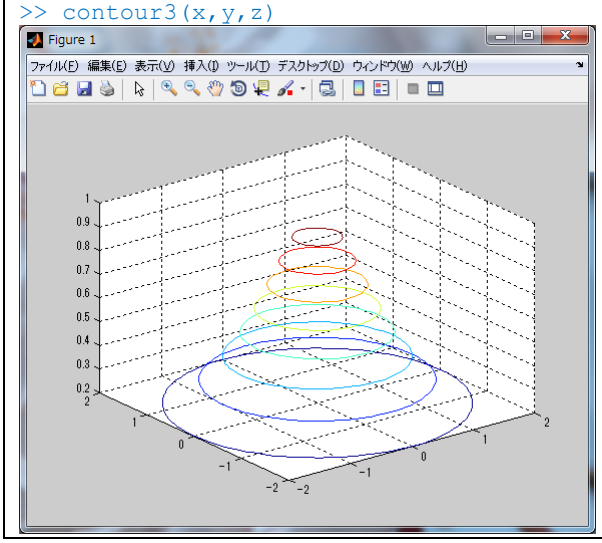
```
>> [x,y] = meshgrid(-2:.1:2, -2:.1:2);
>> z=1./(x.^2+y.^2+1.^2);
>> surf(x,y,z)
```




```
>> contourf(x,y,z)
```




```
>> contour3(x,y,z)
```



■制御命令

```
>> sum=0;
for i=1:100
  if rem(i,2)==1
    continue
  end
  sum=sum+i;
end
sum

sum =

       2550
```

1 から 100 までの奇数の和を計算するプログラム例。if, for, while, continue, break などの制御命令が使えるため、普通のプログラミング言語と同等な制御が可能。

■ファイル入出力

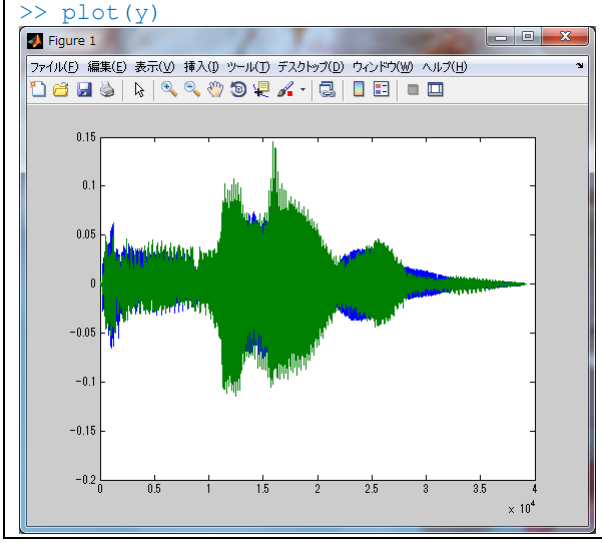
```
>> m=rand(4,3)

m =

    0.4854    0.9157    0.0357
    0.8003    0.7922    0.8491
    0.1419    0.9595    0.9340
    0.4218    0.6557    0.6787

>> csvwrite('testmat.csv',m)

>> m2=csvread('testmat.csv')

m2 =

    0.4854    0.9157    0.0357
    0.8003    0.7922    0.8491
    0.1419    0.9595    0.9340
    0.4218    0.6557    0.6787
```

csv ファイルに行列を読み書きする。他にもいろいろなファイルを扱う関数が用意されている。

■その他の機能

■音声（サウンド）

```
>> [y,Fs]=wavread('WindowsLogonSound.wav');
>> plot(y)
```

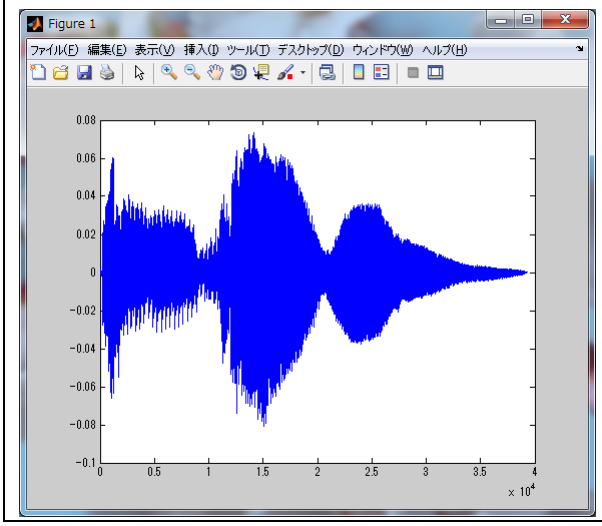


wave(.wav)音声ファイルを読み込み、グラフに左右の音の波形をプロットした。（これは Windows 7 のログイン時の音のファイル C:¥Windows¥Media¥Windows ログオン サウンド.wav である）

```
>> sound(y,Fs)
```

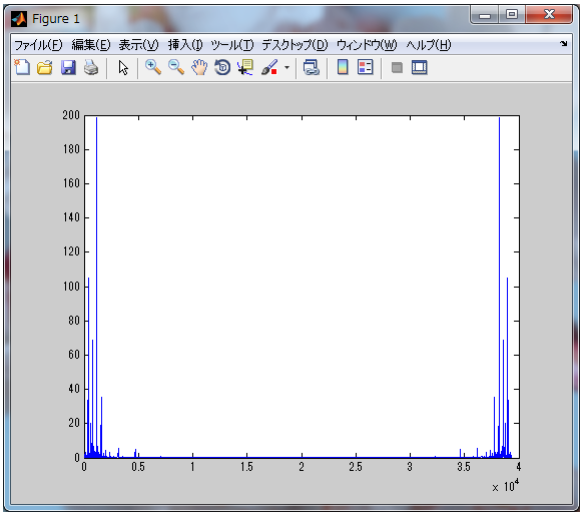
上のコマンドで wave ファイルの音を聞いて確認することができる。

```
>> y1=y(:,1)
```

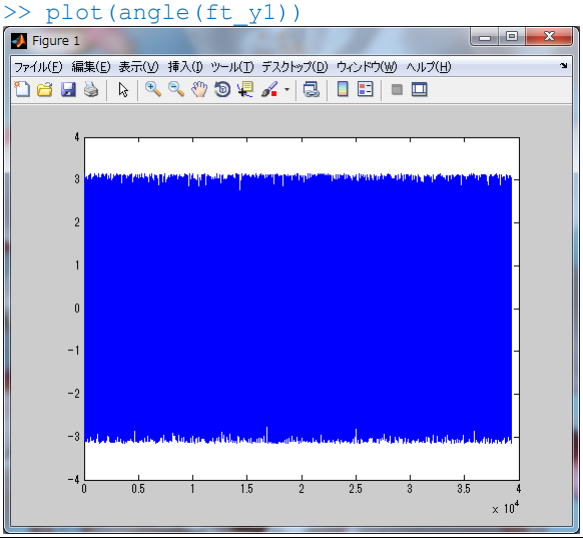


1 列目(左の音)を抽出して波形をプロットした。

```
>> ft_y1=fft(y1)
>> plot(abs(ft_y1))
```



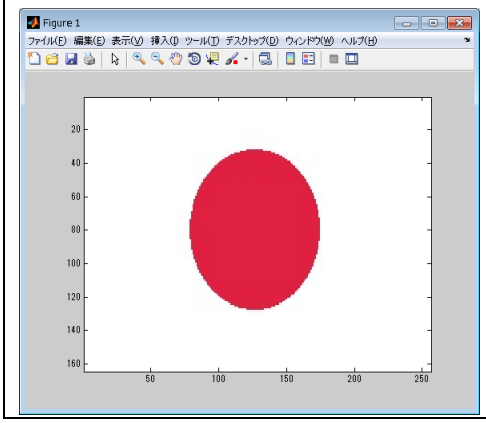

```
>> plot(angle(ft_y1))
```



波形を高速フーリエ変換(FFT)してスペクトルを見る。fft 関数は、上の例では時間波形 y1(t)に対して時間 t の概念が消えているように見えるが、これは離散フーリエ変換だからである。fft 後の横軸はインデックス m(0,1,2, … , N-1)の物理的意味を再現したい場合は、時間波形 y1(t)の再生時間が T であったとすると、横軸のインデックス値 m に対応する周波数は(1/T)*m であると解釈すれば良い。

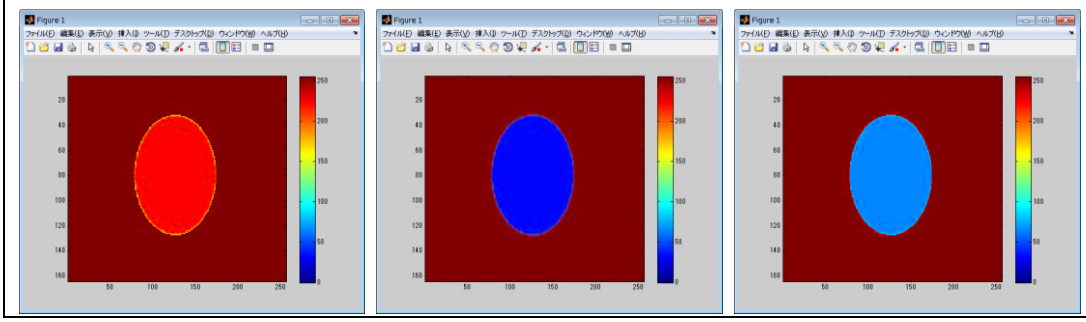
■画像（イメージ、ピクチャ）

```
>> rgb = imread('rising_sun.jpg');
>> image(rgb)
```



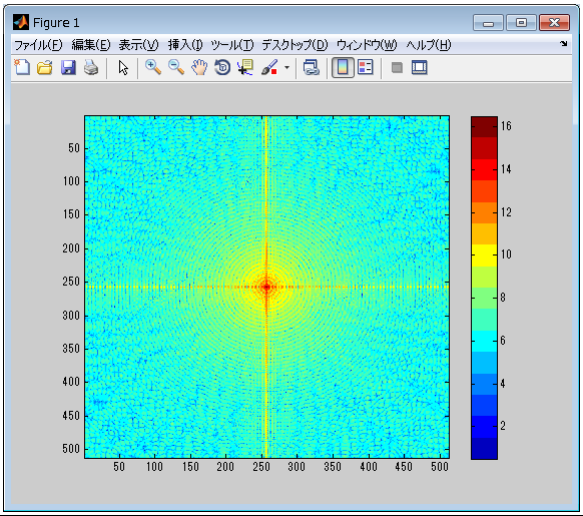
イメージファイルを読み込んで表示する。

```
>> colormap(jet(256))
>> image(rgb(:,:,1)); colorbar
>> image(rgb(:,:,2)); colorbar
>> image(rgb(:,:,3)); colorbar
```



RGB 成分ごとに表示する。

```
>> f=fft2(rgb(:,:,2),512,512);
>> f2=log(abs(f));
>> f3=fftshift(f2);
>> max(f3(:))

ans =

   16.0231

>> image(f3); colormap(jet(16)); colorbar
```



画像の G(緑)成分の 2 次元 FFT の結果の表示。