MATLAB 利用の手引き

TSUBAME Computing Services, Global Scientific Information and Computing Center

2024-01-16

Table of contents

1. はじめに

本書は、MATLABを東京工業大学学術国際情報センターの TSUBAME3 で利用する方法について説明しています。 また、TSUBAME3 を利用するにあたっては、「TSUBAME利用の手引き」もご覧下さい。
利用環境や注意事項などが詳細に記述されております。
MATLABの開発元ではMATLABに関するWebページを公開しています。
次のアドレスを参照してください。

http://www.mathworks.co.jp/

1.1. 利用できるバージョン

TSUBAME3で利用可能な最新バージョンについてはTSUBAME計算サービスWebサイトの アプリケーション ページをご確認下さい。
研究に支障がない限り、バグ修正の入っている最新版をご利用下さい。

1.2. 概要

MATLABは、テクニカルコンピューティングのための高性能ランゲージです。
MATLABは、問題やそれに対する解を、よく知られた数学の記法で表現し、 計算、可視化、プログラミングなどを利用しやすい環境で統合しています。
典型的な利用形態としては、次のようなものがあります。

・数学と計算

・アルゴリズムの開発

・データの収集

・モデリング、シミュレーション、プロトタイピング

・データ解析、データ補間、データ可視化

・科学、工学でのグラフィックス

・グラフィカルユーザインタフェース構築などのアプリケーションの開発

MATLABは、対話型システムで、その基本的な要素に次元を必要としない配列を持っています。
これにより、CやFortranなどのスカラ的な非対話型言語でプログラムを書く時間をかけることなく、 多くの技術計算の問題、特に行列とベクトルの形式を用いた問題を解くことが可能になります。
MATLABという名前は、matrix laboratoryを意味しています。
MATLABは、当初LINPACK、EISPACKプロジェクトによって開発された行列ソフトウェアへの アクセスを容易にするという目的で書かれました。
現在、MATLABは、LAPACK、ARPACKプロジェクトによって開発されたソフトウェアを使用しています。
LAPACK、ARPACKは、ともに、行列計算のためのソフトウェアにおける最先端技術を代表するものです。
MATLABは、多くのユーザによる使用で、長年にわたり発展してきました。
大学では、数学、工学、科学分野で、入門コース、上級コースのための標準的な教育用ツールとなっています。
工業的には、MATLABは、高生産性の研究、開発、解析に対しての優れたツールです。
MATLABの特色は、特定分野の解決策としての ツールボックス群があることです。
多くのMATLABユーザにとって非常に重要なこととして、ツールボックスを使うと、 ユーザは特定のテクノロジーについて、学び、 適用 することができるということが挙げられます。
ツールボックスは、MATLAB関数(M-ファイル)を広く集めたもので、 特定分野の問題を解くのにMATLAB環境を拡張したものです。
ツールボックスが利用できる信号処理、制御システム、 ニューラルネットワーク、ファジィロジック、ウェーブレット、シミュレーション、 その他多くの分野があります。

1.3. MATLABシステム

MATLABシステムは、次の5つの主要部分から成ります。

1.3.1. デスクトップツールと開発環境

これは、ユーザがMATLAB関数やファイルを使うためのツールや機能の集まりです。
これらのツールの多くは、グラフィカルユーザインタフェースです。 グラフィカルユーザインタフェースには、MATLABデスクトップとコマンドウィンドウ、コマンド履歴、 さらに、ヘルプ、ワークスペース、ファイル、サーチパスなどをみるためのブラウザがあります。

1.3.2. MATLAB 数学関数ライブラリ

合計、正弦、余弦、複素計算などの基本的な関数から、 逆行列、行列の固有値、ベッセル関数、高速フーリエ変換などの 高度な関数まで幅広い計算アルゴリズムを含みます。

1.3.3. MATLAB言語

MATLAB言語は、フローコントロールステートメント、関数、データ構造、入力/出力、 オブジェクト指向プログラミングの特色などをもつ、行列/配列を基にした高水準言語です。
このため、手軽に動かせるその場限りのプログラムを作成する小規模プログラミングも、 大きく複雑で、まとまったアプリケーションプログラムを作成する 大規模プログラミングにも対応しています。

1.3.4. Graphics

MATLABは、ベクトルや行列をグラフで表示するための様々な機能を持っています。
2次元、3次元データの可視化イメージプロセッシング、アニメーション、 プレゼンテーショングラフィックスなどのための高水準コマンドを含みます。
さらに、MATLABのグラフィックスでは、低水準コマンドを使用することもでき、 ユーザのMATLABアプリケーションについての完全なグラフィカルユーザインタフェースを構築するのと同様に、 グラフィックスの外観をカスタマイズすることができます。

1.3.5. MATLAB External Interfaces/API

このMATLABアプリケーションプログラムインタフェース(API)は、 MATLAB対話型でCやFortranプログラムを書くためのライブラリです。
これは、MATLABからのルーチンを呼び出す機能(ダイナミックリンク)、 計算エンジンとしてMATLABを呼び出す機能、MATファイルを読み込んだり 書き込んだりするための機能などを含みます。

2. 利用方法

2.1. TSUBAMEでの使用方法

2.1.1. インタラクティブ実行

ログインノードは計算ノードとは別構成となっており、ログインノード上でアプリケーションを実行することは想定されておりま せん。
ログイン方法を参考にログインノードにログイ ン後、インタラクティブノードを利用したX転送を 参考にノードをX転送付きで確保して下さい。
以下以降の例では、全て計算ノードにログインした状態で行います。

コマンド実行例

下記の例では、2時間接続で、割り当てノードとしてr0i0n0が割り当てられた場合を想定しております。
割り当てノードはコマンド実行時に空いているノードですので、明示的にノードを指定することはできません。

```
#qrshの実行
$ qrsh -g [TSUBAMEグループ] -l s_core=1 -l h_rt=2:00:00
Thu Sep 21 08:17:19 JST 2017
r0i0n0:~>

r0i0n0:~> module load matlab/R2017a
# GUIの起動例
r0i0n0:~> matlab
#CUIの起動例
r0i0n0:~> matlab -nodisplay
```

2.1.2. Univa Grid Engineによるバッチ実行

下記がバッチ実行に使用するシェルスクリプトのテンプレートです。
予めスクリプトファイルであるMファイルを準備して下さい。

```
シェルスクリプトの例 (sample.sh)
#!/bin/bash
#$ -cwd
#$ -l f_node=2
#$ -l h_rt=0:30:0

#moduleのロード
. /etc/profile.d/modules.sh
module load matlab/R2017a

#実行したいソフトウェアをバッチモードで実行 (AlignMultipleSequencesExample.mが必要)
matlab -nodisplay -r AlignMultipleSequencesExample
```

以下のコマンドでジョブを投入します。

```
$ qsub -g [TSUBAMEグループ] sample.sh
```

2.2. Windowsでの使用

TSUBAME上での起動方法を先に紹介しましたが、TSUBAME上ではなく端末側で起動したほうが問題の発生が抑えられます。
Windowsにインストールする場合は、こちらをご参照下さい。

2.3. ライセンス利用状況の確認

以下のコマンドにより確認を行います。

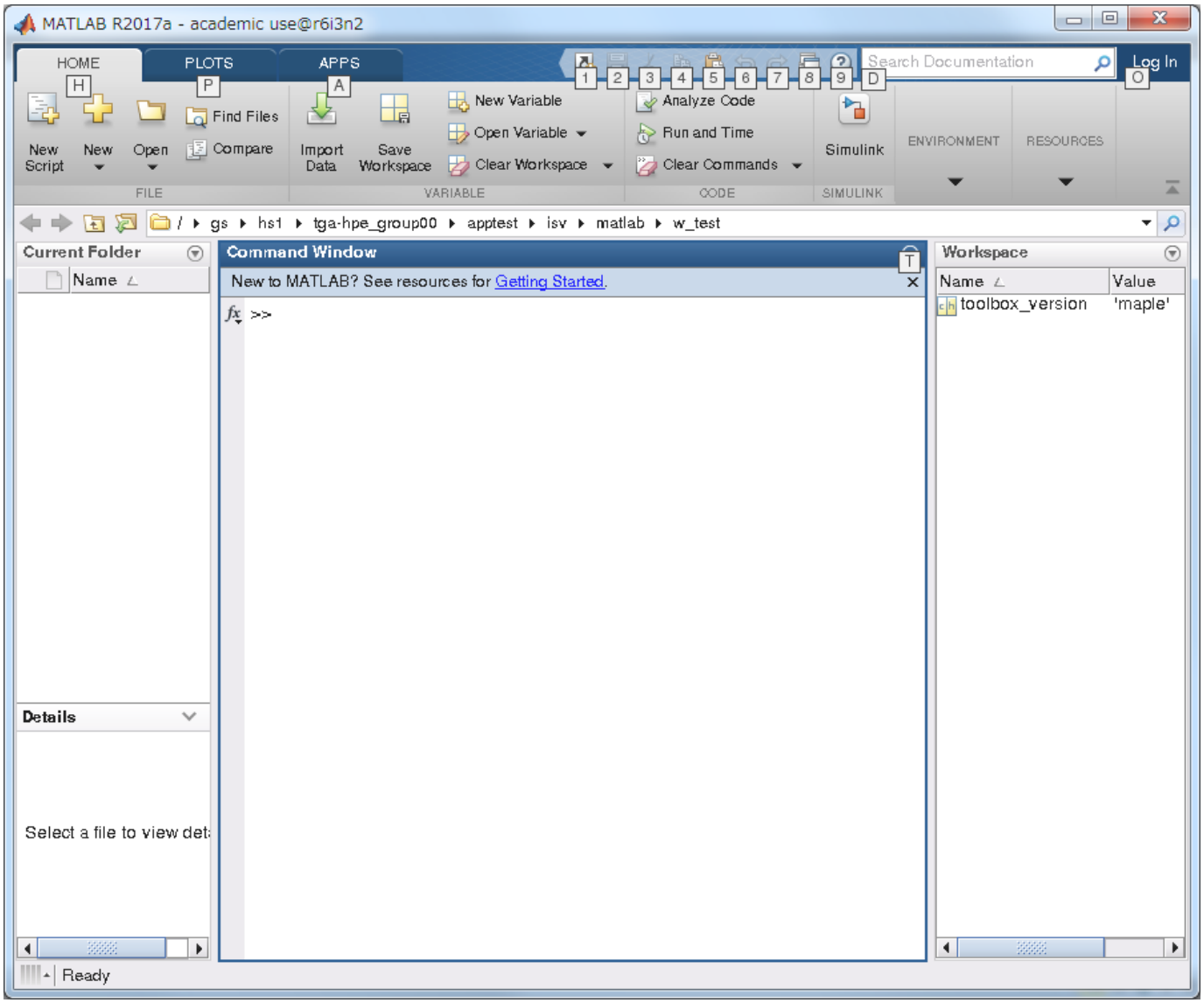
```
$ lmutil lmstat -S MLM -c 27014@lice0:27014@remote:27014@t3ldap1
```

3. MATLABの基本的な使用方法

3.1. オペレーション機能

3.1.1. デスクトップ環境

MATLABを起動すると、各種機能を持つウィンドウがCommand Windowと共に表示されます。
これらのウィンドウの統合環境をデスクトップ環境と呼んでおり、 このデスクトップ環境はいろいろなウィンドウの組み合わせで表示することができます。



3.1.1.1. Command Window

MATLABの基本的な作業ウィンドウです。 ここで各種コマンド・関数・プログラムを実行します。 Command Window上の「>>」記号に続けて、A=2+3と入力して下さい。 この入力で「2+3」という右辺の数式が実行され、 その結果が左辺の変数Aに代入されます。 MATLABでは「=」は数学的等価関係ではなく、 右辺の計算結果を左辺に代入する操作を表します。 つまり、数学的には成立しえない「A=A+2」などの式が成立します。

```
>> A=2+3

A =
```

```
     5

>>
```

3.1.1.2. Workspace

MATLAB上で定義された変数の一覧を表示するウィンドウです。 Workspaceとは、変数に対してMATLABが自動的に割り当てるメモリ領域のことをいいます。 また、変数を右クリックで選択すると変数のプロットや各種の編集を行うことができます。

3.1.1.3. Current Directory

カレントフォルダのファイルを表示します。 上部のボタンを押すことで作業フォルダの移動や新規フォルダの作成を行うことができます。 また、ファイルを右クリックで選択してファイルの表示や実行を行うことができます。 このウィンドウはフォルダのエクスプローラウィンドウに相当します。

3.1.1.4. Command History

今まで実行してきたMATLABコマンドの履歴を表示します。 履歴の各行を左クリックすると、そのコマンドを実行します。

3.1.1.5. Startボタン

MATLABのオプションツールで提供されている各種ツールやデスクトップツールのメニュー集です。 ここからツールごとのデモやヘルプ、MATLABの設定画面などを参照することができます。

3.1.2. 詳細設定

3.1.2.1. デスクトップ環境の詳細設定

「START」ボタンから「Preferences」を選択すると、設定画面を開くことができます。 なお、再起動後も設定した内容と同じ状態でMATLABが起動します。

3.1.2.2. ウィンドウの表示位置の変更方法

ウィンドウを左クリックした状態でドラッグすることで、任意の位置にはめ込むことができます。 「Desktop」メニューから「Desktop Layout」→「Default」を選択することで、デフォルト位置に戻すことができます。

3.2. 変数の基本的なデータ操作

3.2.1. 入力によるデータ定義

直接データを入力して変数を定義する場合、下記の規則に従う必要があります。

- 各要素は、ブランク、タブ、カンマで区切ります
- 要素全体は大括弧[]で囲みます
- 各行はセミコロン;、またはキャリッジリターンで区切ります
- ステートメントの最後にセミコロン;を付けると結果を表示しません
- 虚数単位は小文字のiまたはjを使用します
- ステートメントの最後にピリオドを3つ以上付けると、次の行への継続となります
- 文字データを定義する場合は、要素全体をシングルコート'で囲みます
- データを持たない変数は、空配列[]として定義します

(例)スカラ(1×1行列)変数A

```
>> A=1
```

```
A =

     1
```

(例)結果の非表示

```
>> A=1;
>>
```

(例)1行3列の行ベクトル変数C1

```
>> C1=[1 2 3]

C1 =

     1     2     3
```

(例)3行1列の列ベクトル変数C2

```
>> C2=[1;2;3]

C2 =

     1
     2
     3
```

(例)2行3列の実数行列変数D

```
>> D=[11,12,13;14,15,16]

D =

    11    12    13
    14    15    16
>> D=[11,12,13
14,15,16]

D =

    11    12    13
    14    15    16
```

(例)1行3列の複素行列変数E

```
>> E=[1+i,2+3i,5-2i]

E =

   1.0000 + 1.0000i   2.0000 + 3.0000i   5.0000 - 2.0000i
```

(例)1行6列の文字列変数F

```
>> F='MATLAB'

F =

MATLAB
```

(例)空変数G

```
>> G=[]

G =

     []
```

3.2.2. 関数による定義

規則的な要素をもつ大きなデータを定義する場合、前項で述べた要素を1つずつ入力していく方法はかなり非効率的です。 その代わりに、下表に示されている行列作成関数と演算子を使うことで、比較的簡単に大きなサイズのデータを作成できます。

zeros	ゼロ行列
rand	一様分布する乱数
ones	全要素が1の行列
randn	正規分布する乱数
eye	単位行列
linespace	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
:	等間隔ベクトル

(例)2行3列のゼロ行列m1

```
>> m1=zeros(2,3)

m1 =

     0     0     0
     0     0     0
```

(例)1行4列の一様分布する乱数ベクトルm2

```
>> m2=rand(1,4)

m2 =

    0.8147    0.9058    0.1270    0.9134
```

例)1行10列の等間隔ベクトルm3
等間隔ベクトルはコロン:を使って、「初期値　増分値　最終値」というフォーマットで定義します

```
>> m3=1:3:30

m3 =

     1     4     7    10    13    16    19    22    25    28
```

(例)1行10列の等間隔ベクトルm4
増分値が1の場合は増分値を省略可能(フォーマットは初期値　最終値となります)

```
>> m4=1:10

m4 =

     1     2     3     4     5     6     7     8     9    10
```

3.2.3. データの配列操作

配列操作に関して、次の2行3列の実数行列変数Mを例とします。

```
>> M=[1,2,3;4,5,6]

M =

     1     2     3
     4     5     6
```

3.2.3.1. 配列要素の取り出し

配列要素を取り出すには、変数名の後ろに()付きで行・列番号を指定します。
(例)変数Mの2行3列目の要素

```
>> a1=M(2,3)

a1 =

     6
```

(例)変数Mの2行目の1,2,3列の要素

```
>> a2=M(2,[1,2,3])

a2 =

     4     5     6
```

(例)変数Mの1行目の全ての列要素

```
>> a3=M(1,:)

a3 =

     1     2     3
```

3.2.3.2. 配列要素の置き換え

変数要素の置き換えは 変数名(i,j)=N ここでi,jは変数の行・列番号、Nは置き換える値。

(例)変数Mの2行2列目を1に置換

```
>> M(2,2)=1

M =

     1     2     3
     4     1     6
```

(例)変数Mの1列目を全て5に置き換え

```
>> M(:,1)=5
M =
     5     2     3
     5     1     6
```

3.2.3.3. 配列要素の結合

通常のデータ定義のように、大括弧[]を使用して配列同士を結合できます。

(例)変数a1とa2を横に結合

```
>> a12=[a1,a2]

a12 =

     6     4     5     6
```

(例)変数a3とa2を縦に結合

```
>> a32=[a3;a2]

a32 =

     1     2     3
     4     5     6
```

3.2.3.4. 配列操作関数と演算子

配列の大きさを調べたり、形状を変更したりするための関数が複数用意されています。

size	配列の大きさ
fliplr	行列の左右反転
length	ベクトルの長さ
flipud	行列の上下反転
reshape	行列のサイズ変更
rot90	行列の90°回転
’	共役転置
.’	転置

3.2.4. データ定義の注意

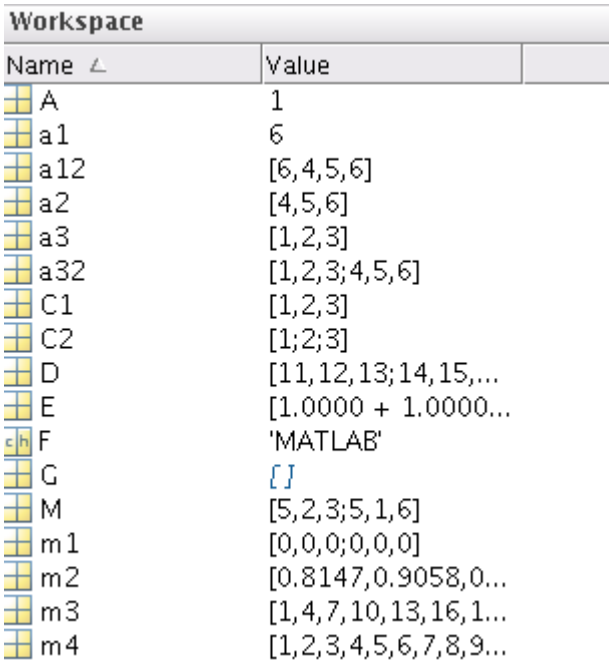
3.2.4.1. 変数名の制限及び注意点

1. 大文字・小文字を区別します
2. 変数名の文字数制限は63文字です
3. 数字および演算子で始まる変数名は使用できません
4. 日本語文字列を変数名に使用することはできません
5. 同じ変数名でデータを定義すると値が上書きされます
6. 変数名を指定せずにデータを定義すると、テンポラリ変数ansとして定義されます
7. 関数・コマンドと同じ変数名を使用しないで下さい
8. 予約変数と同じ変数名を使用しないで下さい

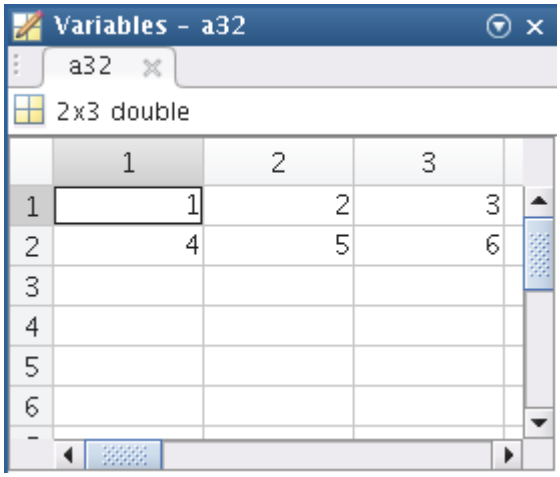
   (例)虚数単位i,j、円周率pi、無限大inf

3.2.5. 配列エディタの機能

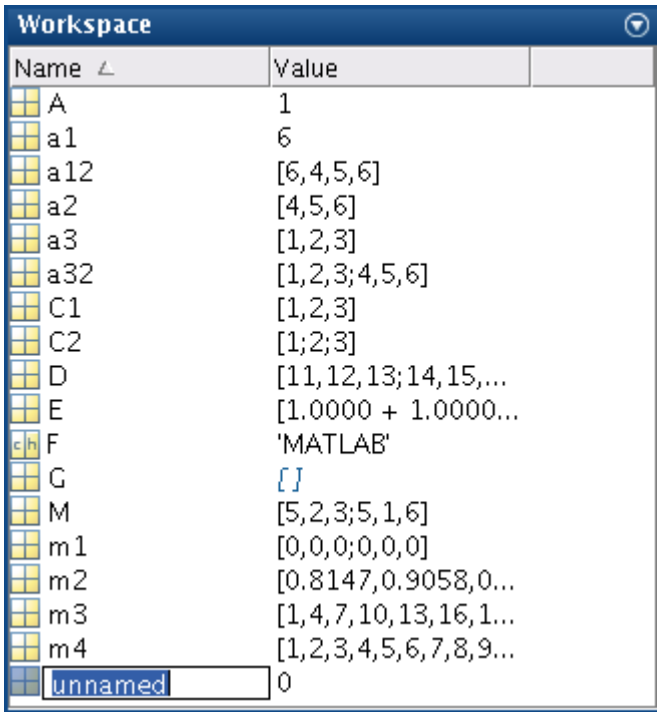
Workspace ウィンドウで変数をダブルクリックすると配列エディタが起動し、 変数の編集を行うことができます。



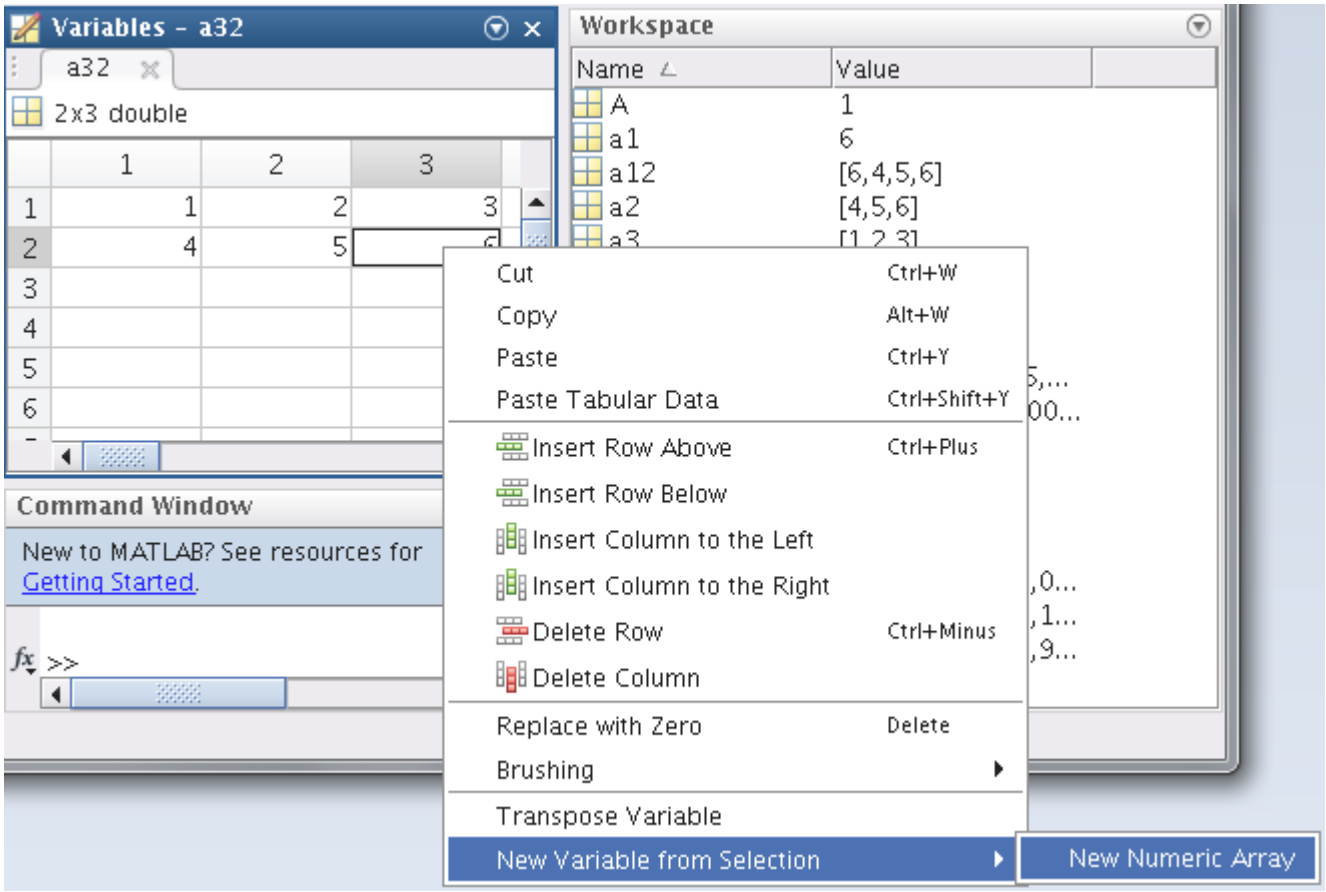
配列エディタが起動します。



はじめから変数を定義する場合、 Workspaceウィンドウの「New」ボタンをクリックすると、 「Unnamed」という変数名がWorkspaceに作成されます。 この「unnamed」は作成直後、ハイライト表示されますので、 変数名を変更することができます。 このとき、「unnamed」は0の要素を持った1行1列の変数(スカラ値)となりますので、 これを、配列エディタを開いて編集します。



既に存在する変数からデータを切り出して定義する場合、 配列エディタ上で要素の一部を選択し、 右クリック⇒「Create Variable from Selection」を選択します。 すると、Workspaceに「a321」という変数が作成されます。



3.3. ヘルプ機能

3.3.1. 関数・コマンド名が分かっている場合

関数やコマンド名が既に分かっている場合、その機能・使用法について調べる方法は大きく分けて3つあります。 ここでは単位行列を作成するeye関数を例として説明します。

3.3.1.1. helpコマンド

次のように入力すると、各関数のヘルプテキストがコマンドウィンドウ上に表示されます。

```
>>help 関数名
```

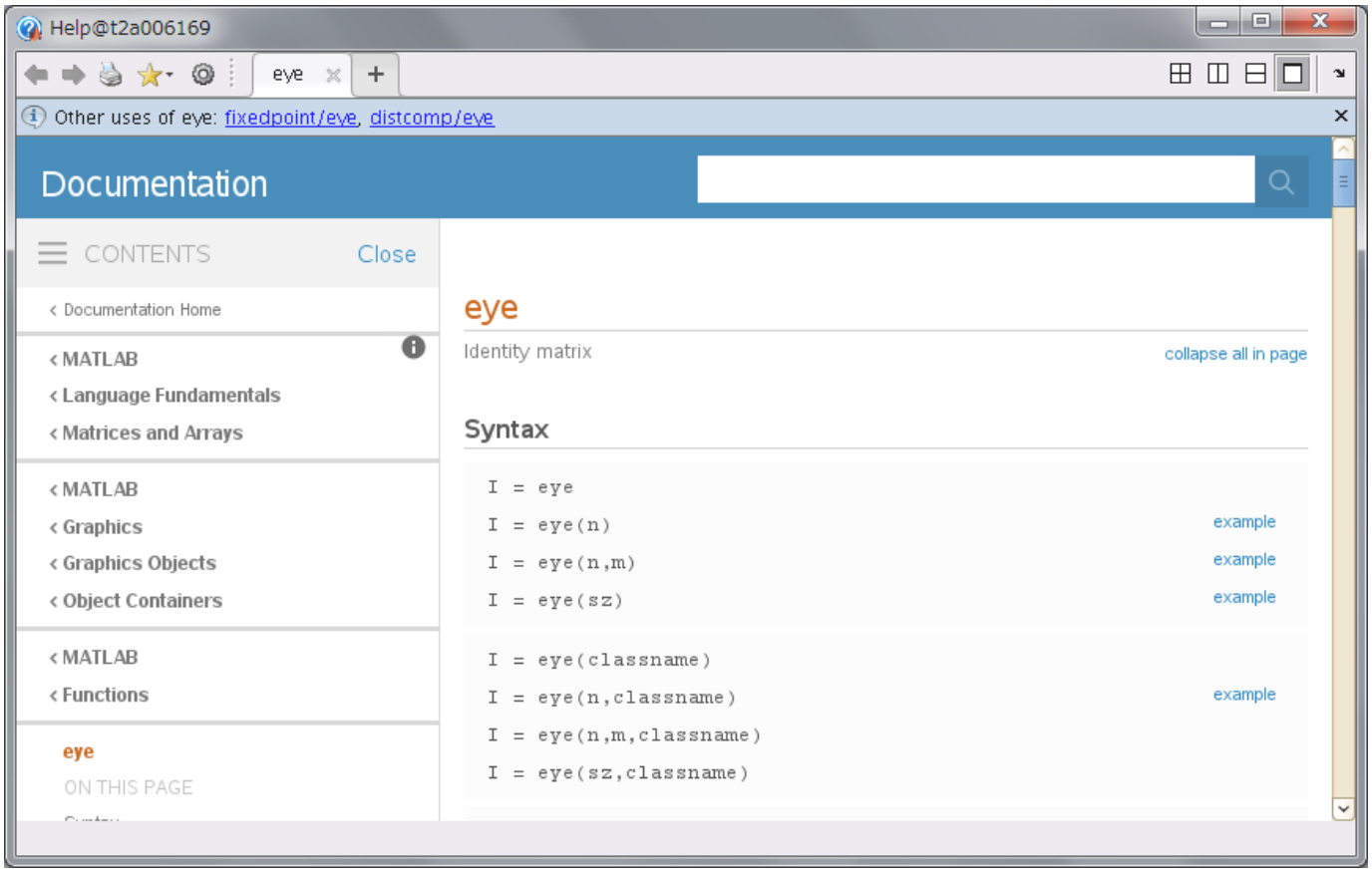
(例)help eye

```
>> help eye
EYE Identity matrix.
   EYE(N) is the N-by-N identity matrix.

   EYE(M,N) or EYE([M,N]) is an M-by-N matrix with 1's on
   the diagonal and zeros elsewhere.

   EYE(SIZE(A)) is the same size as A.
```

3.3.1.2. docコマンド

次のように入力すると、ヘルプブラウザに各関数のリファレンスが表示されます。 ヘルプテキストよりも詳細な情報が欲しいときに使用します。

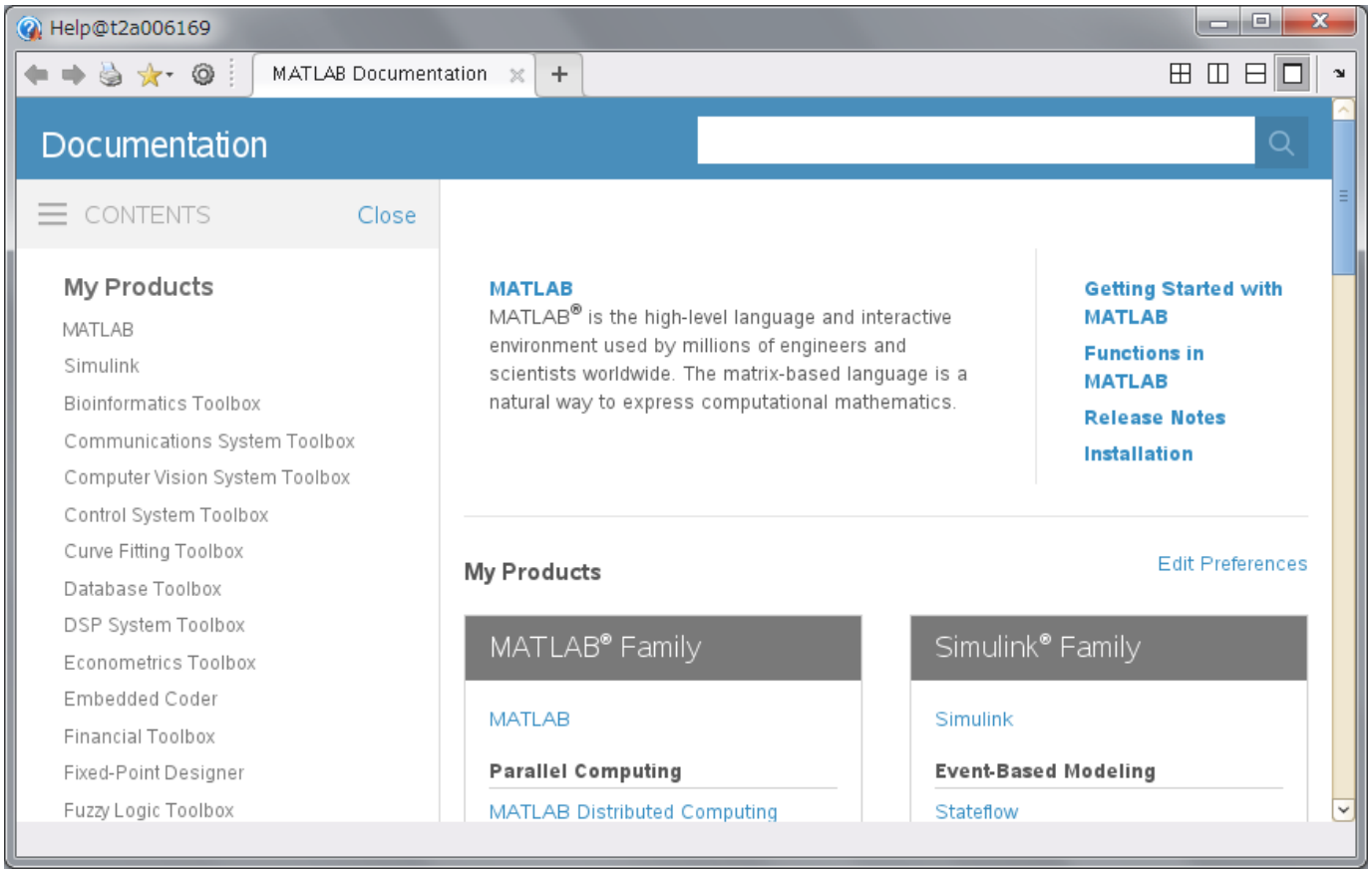
```
>> doc 関数名
```



3.3.1.3. ヘルプブラウザ

ヘルプブラウザは下記コマンドで起動します。

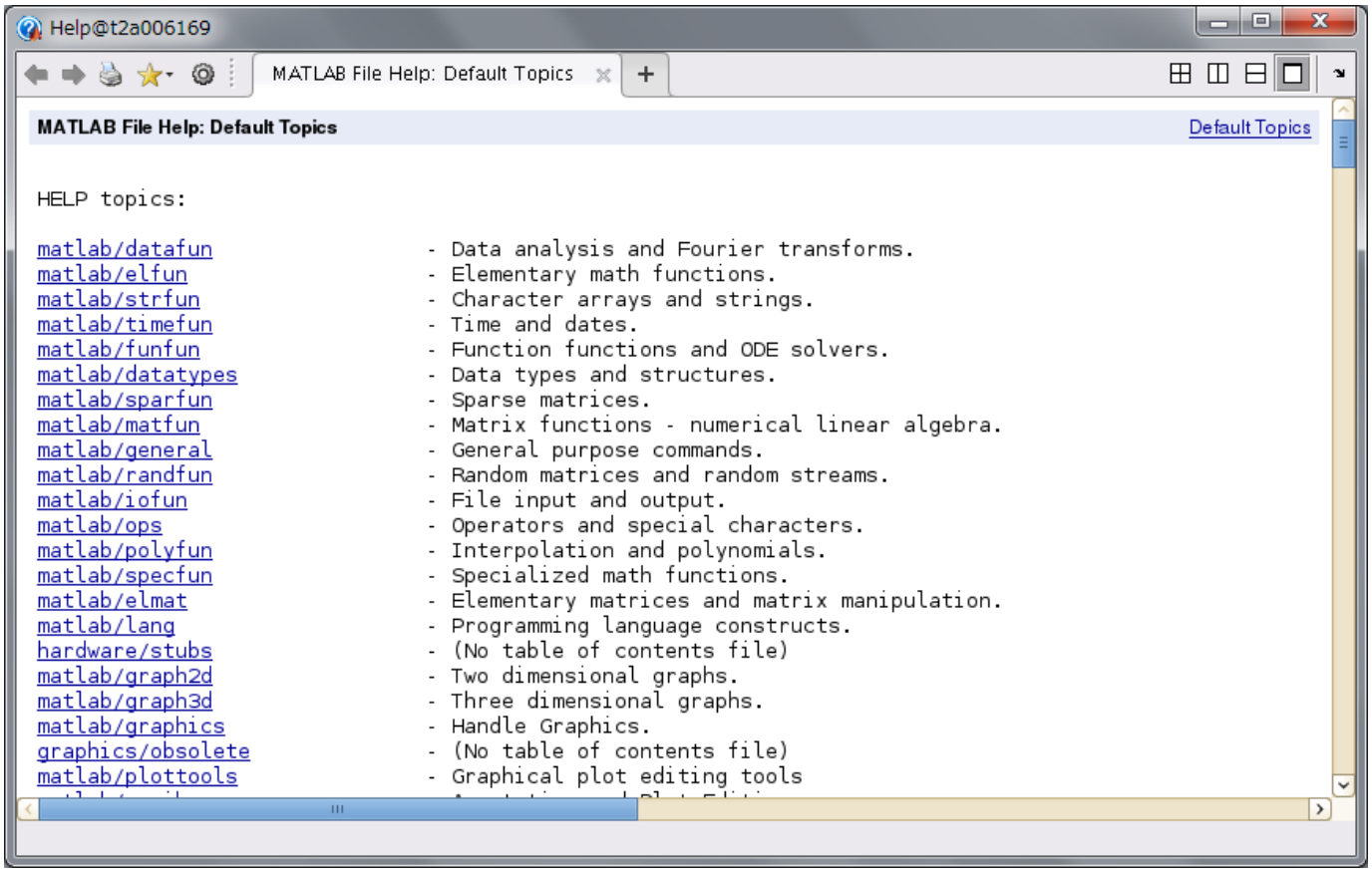
```
>> helpbrowser
```



また、デスクトップウィンドウの「Help」メニューから「MATLAB Help」を選択するか、 「?」アイコンをクリックすることで起動させることができます。 ヘルプブラウザの「Search」タブを選択し、 「Search for」フィールドに検索したい関数名を入力して「Go」ボタンを実行します。

(2)関数・コマンド名が分からない場合

関数やコマンド名は分からないが、目的の機能を持つ関数・コマンドが存在するかどうか調べたい場合には、 helpwinコマンドを利用します。 Helpwinコマンドを実行すると、 各ツールの機能別関数リストがヘルプブラウザに表示されます。



3.3.2. 数値演算

3.3.2.1. 四則演算

MATLABでは、スカラ演算だけでなく行列演算(線形代数則)の演算子も用意します。

+	A + B	行列の加算
-	A - B	行列の減算
*	AB	行列の乗算
^	AB	行列のべき乗
/	AB-1	行列の除算(右割り)
	A-1B	行列の除算(左割り)
.*	A(i,j)*B(i,j)	要素単位の乗算
.^	A(i,j)B(i,j)	要素単位のべき乗
./	A(i,j)/B(i,j)	要素単位の除算
.	B(i,j)/A(i,j)	要素単位の除算

ピリオド「.」の有無により、演算子のスカラ演算と行列演算を区別しています。 加算と減算についてはどちらの演算とも同じ結果になりますので、「.+」「.-」は用意されていません。

(例)各演算子の計算結果の確認

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

```
A =

     1     2
     3     4
>> B=[5,6;7,8]

B =

     5     6
     7     8
>> A+B

ans =

     6     8
    10    12
>> A*B

ans =

    19    22
    43    50
>> A.*B

ans =

     5    12
    21    32
```

3.3.2.2. 数学関数

入力した変数に指定した配列の全要素に対して、計算を行います。 代表的な数学関数を以下に示します。

Sin	正弦値
conj	共役複素数
Exp	指数
real	複素数の実部
log10	常用対数
imag	複素数の虚部
sqrt	平方根
rem	除算の剰余
abs	絶対値
prod	配列の要素の積

(例)行列データに対する余弦値の計算

```
>> x1=0:pi/4:pi;
>> X=[x1;2*x1]

X =

         0    0.7854    1.5708    2.3562    3.1416
         0    1.5708    3.1416    4.7124    6.2832

>> Y=cos(X)

Y =

    1.0000    0.7071    0.0000   -0.7071   -1.0000
    1.0000    0.0000   -1.0000   -0.0000    1.0000
```

3.3.2.3. 行列関数

行列関数は数値演算のコアルーチンを担っている非常に重要な関数です。 代表的な行列関数を以下に示します。

inv	逆行列
norm	行・ベクトルのノルム
det	行列式
null	行列のNULL空間
rank	行列のランク
eig	固有値と固有ベクトル

(例)連立方程式の解法

```
3x+4y=6
2x+5y=8
>> A=[3,4;2,5]

A =

     3     4
     2     5

>> b=[6;8]

b =

     6
     8

>> x=inv(A)*b

x =

   -0.2857
    1.7143

>> x=A\b

x =

   -0.2857
    1.7143
```

処理速度や計算精度の観点から考えると、逆行列を求めてから計算するよりも バックスラッシュ演算子「」で処理した方が有効です。

3.3.2.4. 解析関数

MATLABでは様々な解析関数が用意されています。 ここでは代表的なデータ解析関数を取り上げます。

max	最大値
gradient	勾配
min	最小値
corrcoef	相関係数
mean	平均値
cov	共分散行列
std	標準偏差
interp1	1次元補間
roots	多項式の根
conv	畳み込み
polyfit	多項式近似
fft	高速フーリエ変換
polyval	多項式の計算
fft2	2次元高速フーリエ変換

(例)多項式の根、計算 MATLABでは多項式の係数を係数ベクトルで表現しています。
多項式の解を下の例では求めています。

```
>> coef=[1,5,4]

coef =

     1     5     4

>> R=roots(coef)

R =

    -4
    -1

>> V=polyval(coef,R)

V =

     0
     0

>>
```

(例)多項式の畳み込み

```
>> c1=[1,2,3]

c1 =

     1     2     3

>> c2=[4,5]

c2 =

     4     5

>> r=conv(c1,c2)

r =

     4    13    22    15
```

(例)行列の縦方向の最大値、平均値、相関係数

```
>> M=[2,-10,5;6,13,4;3,5,9]

M =

     2   -10     5
     6    13     4
     3     5     9

>> max(M)

ans =

     6    13     9

>> mean(M)

ans =

    3.6667    2.6667    6.0000

>> corrcoef(M)

ans =

    1.0000    0.8983   -0.4539
    0.8983    1.0000   -0.0162
   -0.4539   -0.0162    1.0000
```

3.3.3. ファイルデータの入出力

ここでは、外部ファイルからデータを読み込んで定義する方法、 及び定義したデータをファイルに保存する方法について取り上げます。

3.3.3.1. ファイルからの入力

下記に示す各種フォーマットのデータを読み込むことができます。

テキストファイル(.dat,.txt,.csv)	数値・文字を含むテキストフォーマット
スプレッドシート形式ファイル(.xls,.wk1)	Excelフォーマット、Lotus123フォーマット
オーディオファイル(.wav,.au)	Windows WAVEフォーマット、Sun Microsystemsフォーマット
オーディオビジュアルファイル(.avi)	AVIオーディオビジュアルフォーマット
イメージファイル(.jpg,.tif,.bmp,.png,.hdf,.pcx,.xwd,.gif)	JPEG,TIFF,BMP,PNG,HDF,PCX,XWD,GIFフォーマット
MAT-ファイル(.mat)	MATLAB固有バイナリフォーマット
その他バイナリファイル(.bin)	ビット解釈やマシンフォーマットの指定されたバイナリフォーマット

MATLABにデータを読み込む方法は、以下の2通りがあります。

・インポートウィザードを使う

・読み込みコマンドを使う

インポートウィザードを使う
インポートウィザードはMATLABにデータを取り込む際に、その読み込みフォーマットを設定するGUIツールです。
上記ファイルフォーマットのほとんどを読み込むことができますが、 ここでは例として以下のテキストファイルを読み込みます。

data1.txt

```
0.000 , 2.000
0.001 , 4.000
0.002 , 6.000
0.003 , 8.000
```

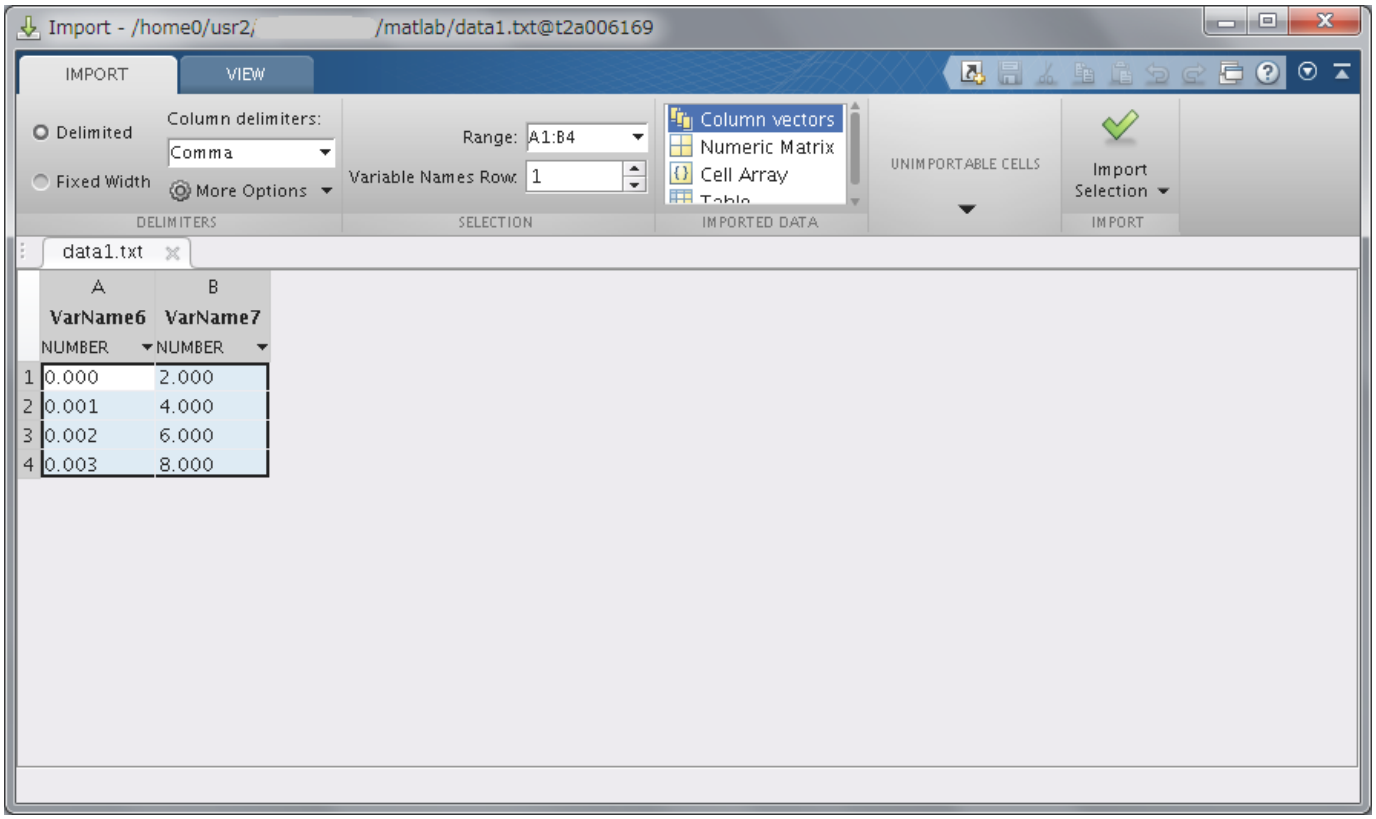
data2.txt

```
// Header //

DATE 2011/02/01
FORMAT ASCII
INTVL 7.85E-2 sec

time disp

0.000  0.000
0.010  3.565
0.020  7.890
0.030 11.345
0.040 15.010
0.050 23.780
```

主な手順は以下の通りです。

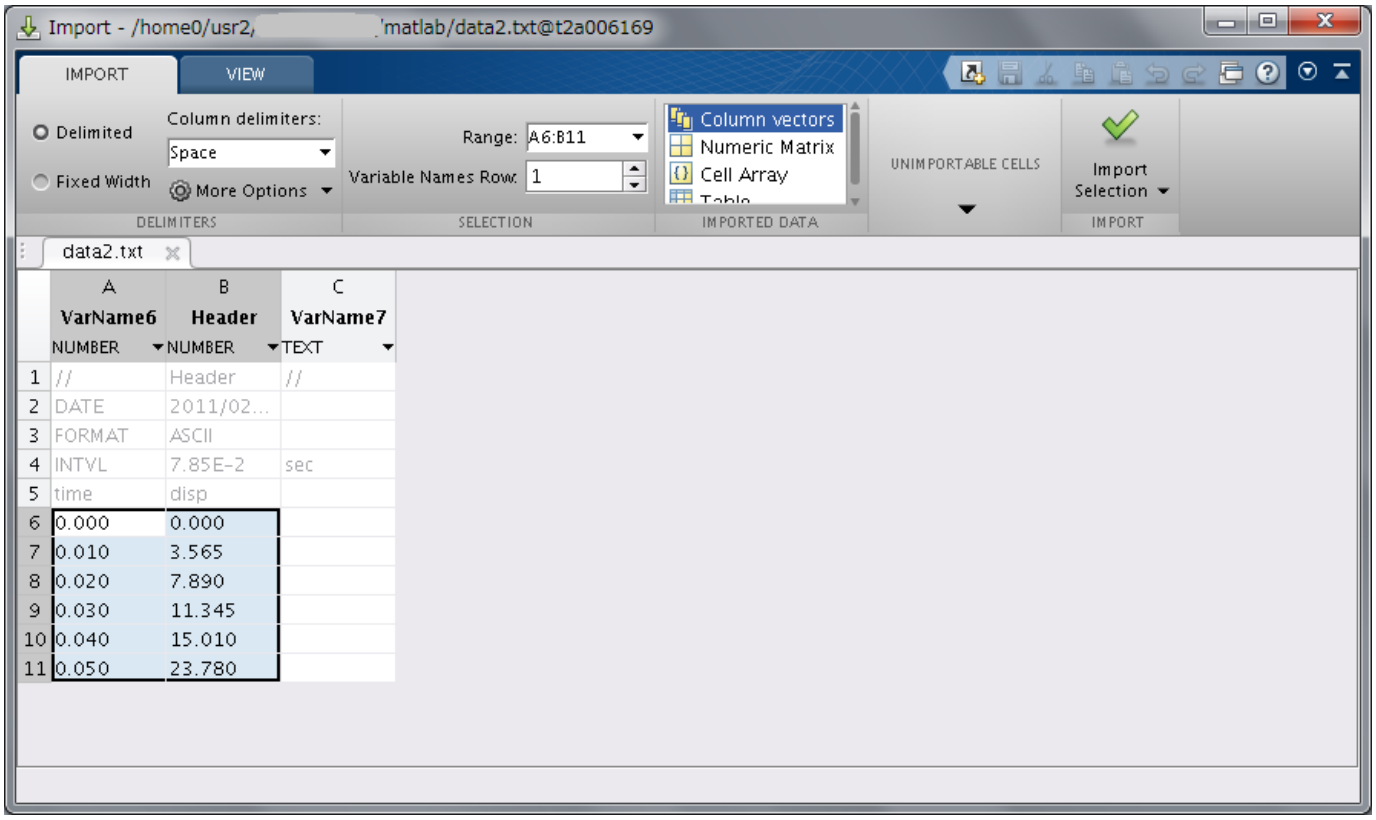
1. 「Import Data」を選択
2. 読み込むファイルを選択し、OKを押す
3. Import Wizardウィンドウの「Next >」ボタンを押す
4. インポートする変数にチェックする
5. 「Finish」ボタンを押す
6. data1.txtの場合

Select Column Separator では、カンマ区切りなので Delimited で「Comma」を選択。



*data2.txtの場合

Select Column Separator では、スペース区切りなのでセパレータに「Space」を選択。 Number of text head lines では、「7」とする。



・読み込みコマンドを使う

読み込みコマンドを使うことで、前節で述べたフォーマットのファイルを全て読み込むことができます。 MATLABのデータインポート関数は大きく分けて2種類あり、それぞれのデータフォーマットに応じて使い分けます。

1. 標準インポート関数
2. 低水準インポート関数
3. 標準インポート関数

各ファイルフォーマットに対応したインポート関数が用意されています。 代表的な標準インポート関数を以下に示します。

load	MAT-ファイル及びブランク区切りのファイル
dlmread	任意の区切り文字で区切られたファイル
textread	フォーマット付き数値・文字を含むファイル
xlsread	Excelスプレッドシートファイル
urlread	URLのファイル
imread	画像ファイル
wavread	WAVEサウンドファイル
aviread	AVIファイル

・低水準インポート関数

標準インポート関数が対応していない複雑なフォーマットの場合は、 低水準インポート関数を使います。 代表的な低水準インポート関数を以下に示します。

fopen	ファイルを開く
fclose	ファイルを閉じる
fgetl	1行読み込み(終端子無し)
fseek	ファイルポインタの設定
frewind	ファイルポインタを先頭に移動
fscanf	フォーマット指定のテキストデータの読み込み
fread	バイナリデータの読み込み
textscan	フォーマット指定のテキストデータの読み込み(大きなデータ)

3.3.3.2. ファイルへの出力

下記のファイルフォーマットへ保存することができます。

テキストファイル(.dat,.txt,.csv)	数値・文字を含むテキストフォーマット
スプレッドシート形式ファイル(.xls)	Excelフォーマット
オーディオファイル(.wav,.au)	Windows WAVEフォーマット、Sun Microsystemsフォーマット
オーディオビジュアルファイル(.avi)	AVIオーディオビジュアルフォーマット
イメージファイル(.jpg,.tif,.bmp,.png,.hdf,.pcx,.xwd)	JPEG,TIFF,BMP,PNG,HDF,PCX,XWDフォーマット
MAT-ファイル(.mat)	MATLAB固有バイナリフォーマット
その他バイナリファイル(.bin)	ビット解釈やマシンフォーマットの指定されたバイナリフォーマット

基本的にはコマンド入力によりデータをファイルに保存します。 ただし、ファイルフォーマットによってはメニュー等から保存することができます。

・エクスポート関数を使う

・Workspace機能を使う(MAT-ファイルのみ)

代表的な標準エクスポート関数

save	MAT-ファイル及びブランク区切りのファイル
csvwrite	カンマ区切りで区切られたファイル(csv形式)
dlmwrite	任意の区切り文字で区切られたファイル
xlswrite	Excelスプレッドシートファイル
urlwrite	URLのファイル
imwrite	画像ファイル
wavwrite	WAVEサウンドファイル
avifile	AVIファイル

Workspace機能(MAT-ファイルでの保存のみ)

Workspaceウィンドウに表示されている変数は下記の手順でMAT-ファイルに保存できます。

1. Workspaceの変数""をクリック
2. Shiftキーを押しながら変数""をクリック
3. 選択範囲を右クリックし、コンテキストメニューから「別名で保存」を選択
4. 「MAT-ファイルに保存」ウィンドウで保存するファイル名を指定(拡張子は.mat)

4. グラフィックス

4.1. 2次元グラフィックス

代表的な2次元グラフィックス関数には、以下のものがあります。

plot	線形プロット
contour	コンタープロット
semilogx	X片対数プロット
quiver	矢印プロット
semilogy	Y片対数プロット
stream2	ストリームプロット
loglog	両対数プロット
image	イメージの表示
plotyy	左右両軸プロット
imagesc	イメージの表示(SC)

2次元グラフィックスの代表的なplot関数の書式は以下になります。

```
plot(x1,y1,’Color LineStyle Marker’,x2,y2,’ Color LineStyle Marker’,…)
```

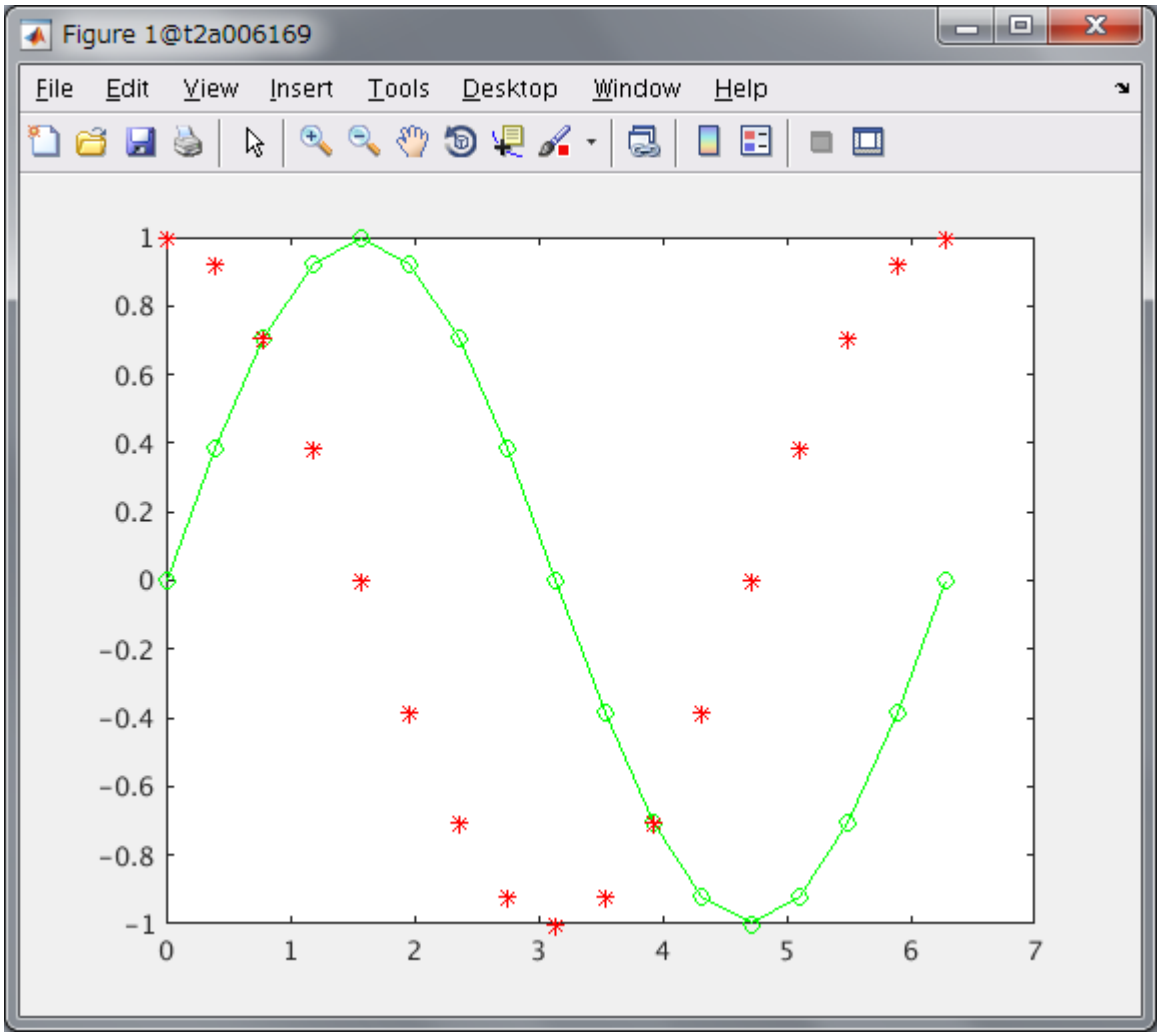
(x1,y1),(x2,y2)はそれぞれ表示するデータの組み合わせを表します。 また、’Color LineStyle Marker’は描画するラインのオプションのプロパティを表し、 それぞれ線の色、線種、マーカーを指定します。 線 のプロパティの詳細については、「doc linespec」コマンドで確認して下さい。

(例)Sinカーブ、Cosカーブのプロット

```
>> x=0:pi/8:2*pi;
>> y1=sin(x);y2=cos(x);
>> plot(x,y1,'g-o',x,y2,'r*')
```

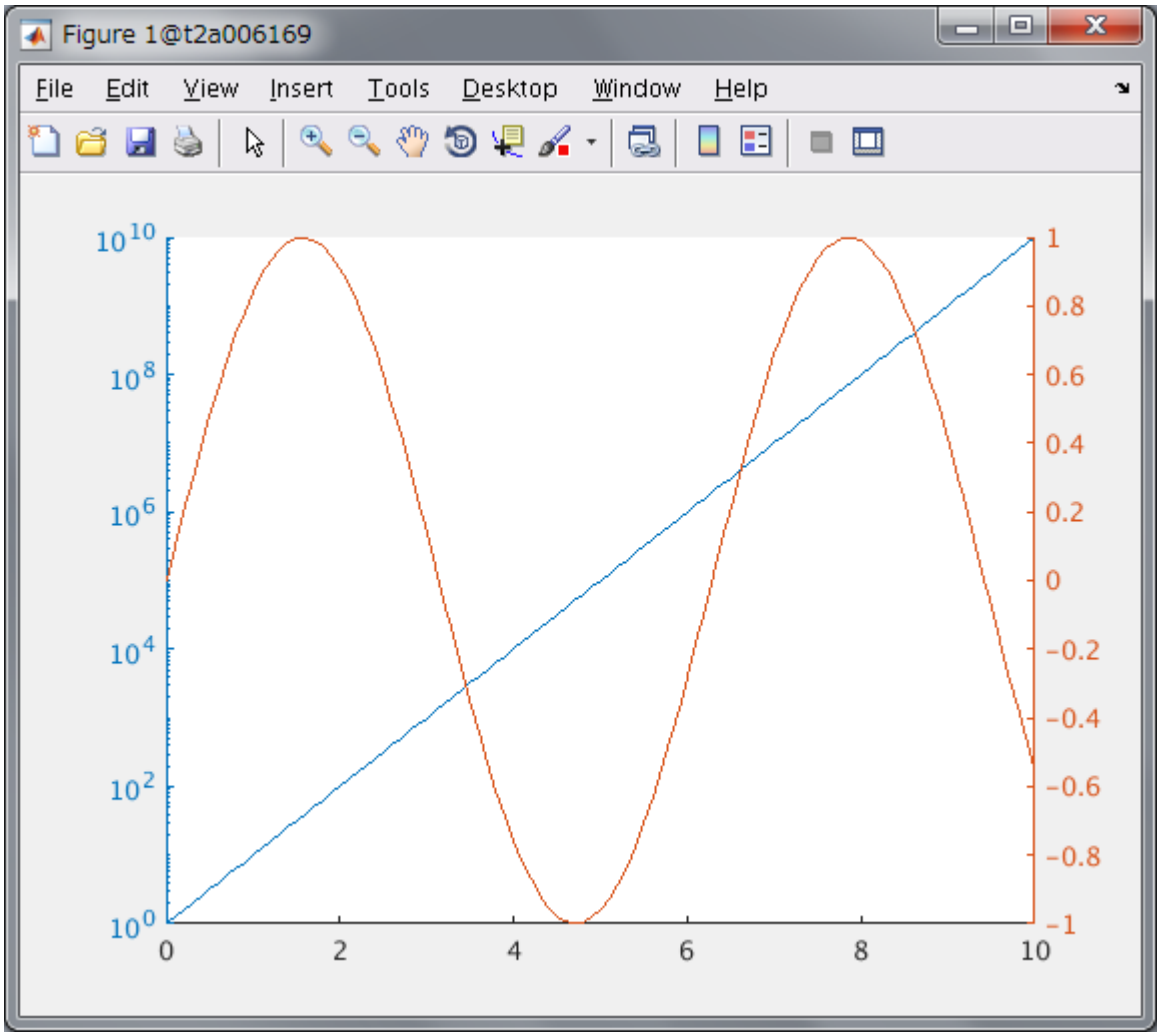
※グラフ線プロパティの説明

	(x,y1)	**(x,y2)**
カラー	緑(g)	赤(r)
ライン	実線(-)	なし
マーカー	丸(o)	アスタリスク(*)



(例)左右両軸プロット

```
>> x=0:0.1:10;
>> y1=10.^x;
>> y2=sin(x);
>> plotyy(x,y1,x,y2,'semilogy','plot')
```



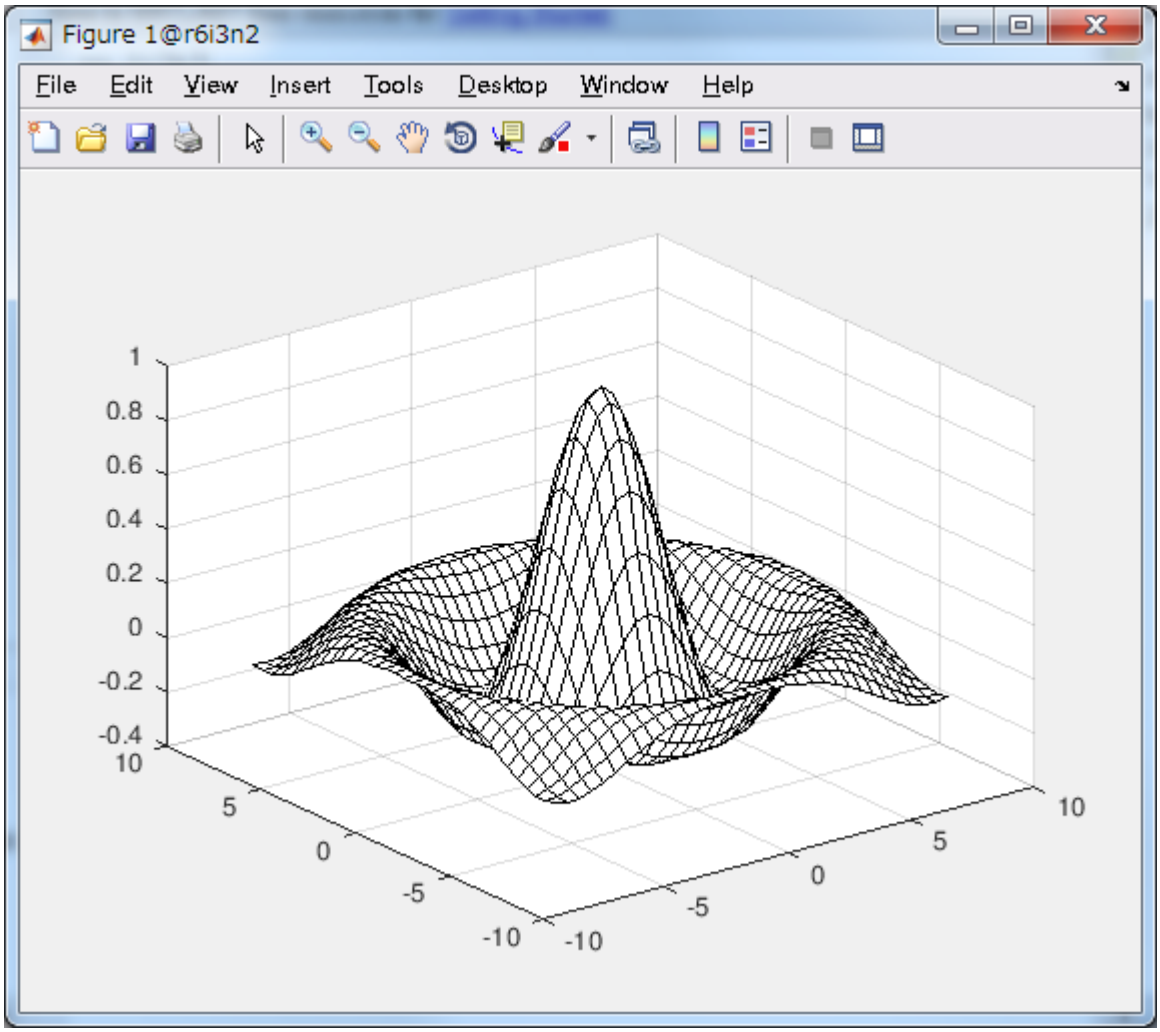
4.2. 3次元グラフィックス

代表的な3次元グラフィックス関数には、以下のものがあります。

plot3	3次元プロット
meshc	メッシュコンタープロット
mesh	メッシュプロット
caxis	カラー軸のスケーリング
surf	サーフィスプロット
colormap	カラーマップ
contour3	コンタープロット
colordef	背景色の設定

(例) 2次元sinc関数、sin(r)/r をx およびy 方向で実行しグラフ化します。 R は、行列の中心である原点からの距離です。 eps (小さな浮動小数点数を出力するMATLABコマンド)を加えると、 原点での0/0が中間で生じることを避けることができます。

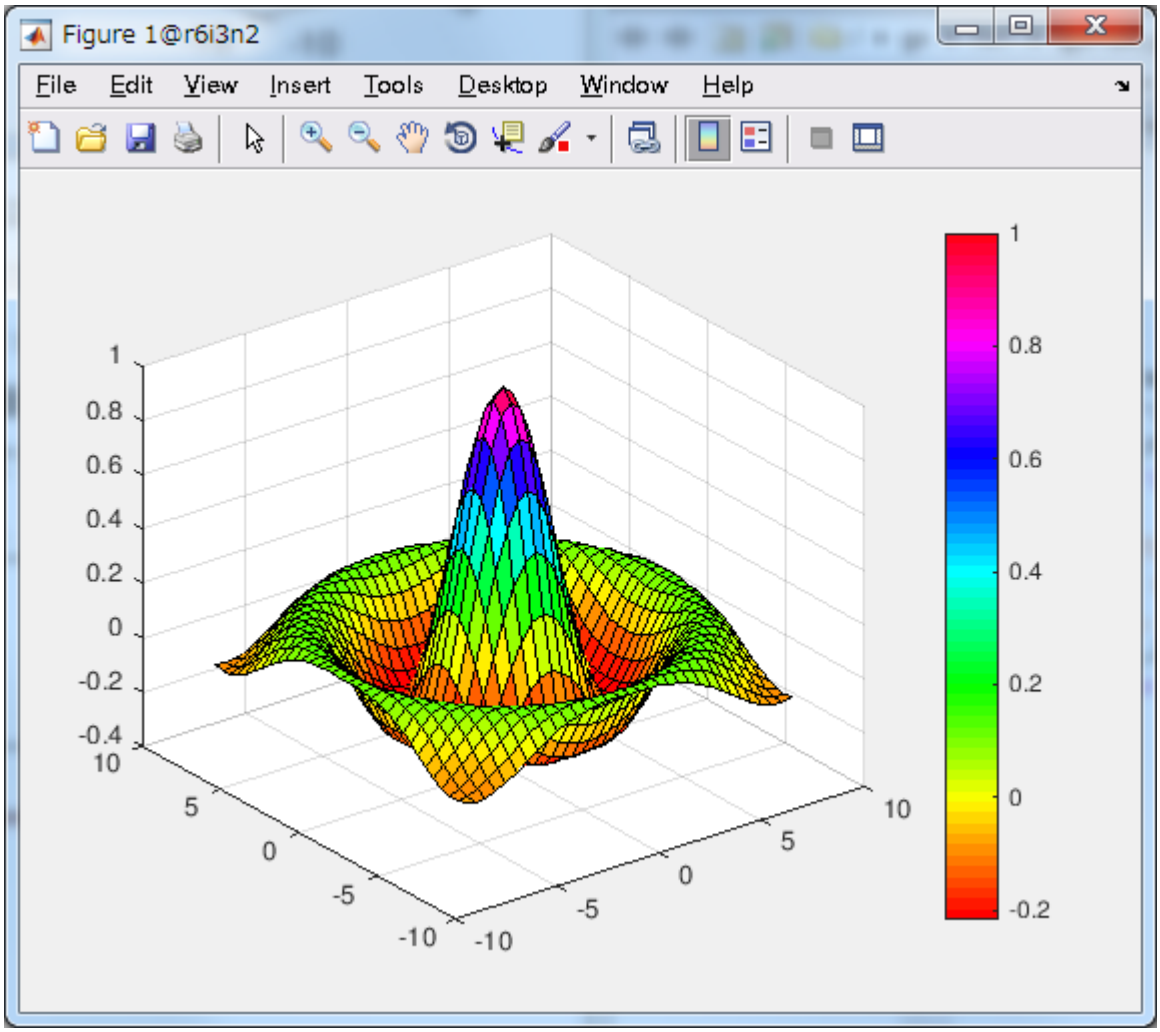
```
>> [X,Y] = meshgrid(-8:.5:8);
>> R = sqrt(X.^2 + Y.^2) + eps;
>> Z = sin(R)./R;
>> mesh(X,Y,Z,'EdgeColor','black')
```



デフォルトでは、 MATLABはカレントのカラーマップを使ってメッシュを色付けします。 しかしこの例題では、 EdgeColor surface プロパティを指定することによって、単色のメッシュを用います。

例　カラーサーフェスプロット サーフェスプロットは、長方形の面が色付けされることを除いて、 メッシュプロットに似ています。 面のカラーは、Z の値とカラーマップによって決定されます(colormap は、順番付けられたカラーのリストです)。 次のステートメントは、sinc 関数をサーフェスプロットとしてグラフ化し、 カラーマップを選択し、カラーバーを付加して、データのカラーへのマッピングを示します。

```
>> surf(X,Y,Z)
>> colormap hsv
>> colorbar
```



(例)透明なサーフェス サーフェスの表面は、可変の程度で透明にすることができます。 透明性(alpha値として参照されます)は、オブジェクト全体に対して指定されるか、あるいはカラーマップと同様に機能するalphamapに基づきます。

```
>> surf(X,Y,Z)
>> colormap hsv
>> alpha(.4)
```

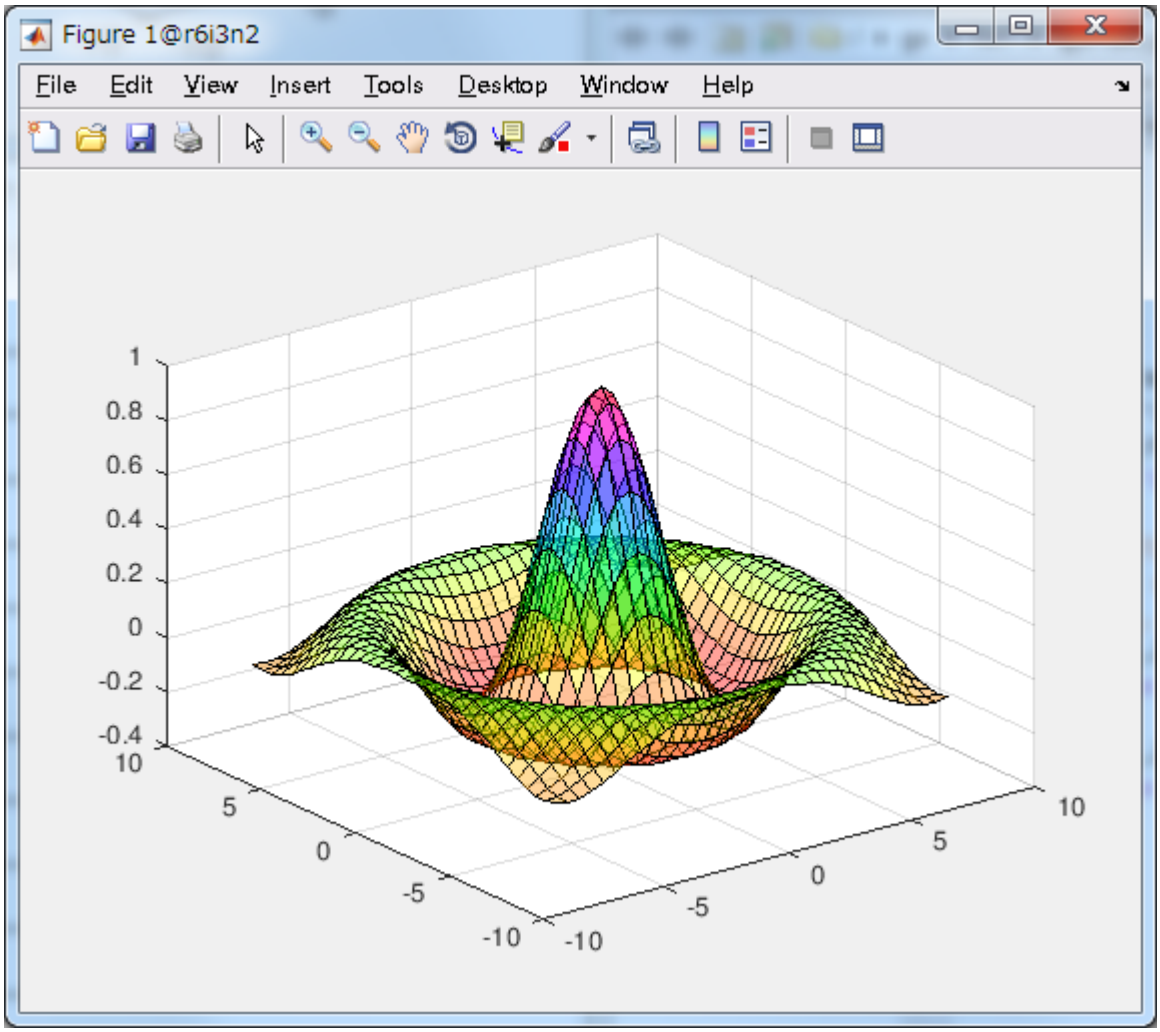


Figure 1@r6i3n2
File Edit View Insert Tools Desktop Window Help
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
10
5
0
-5
-10
-10
-5
0
5
10

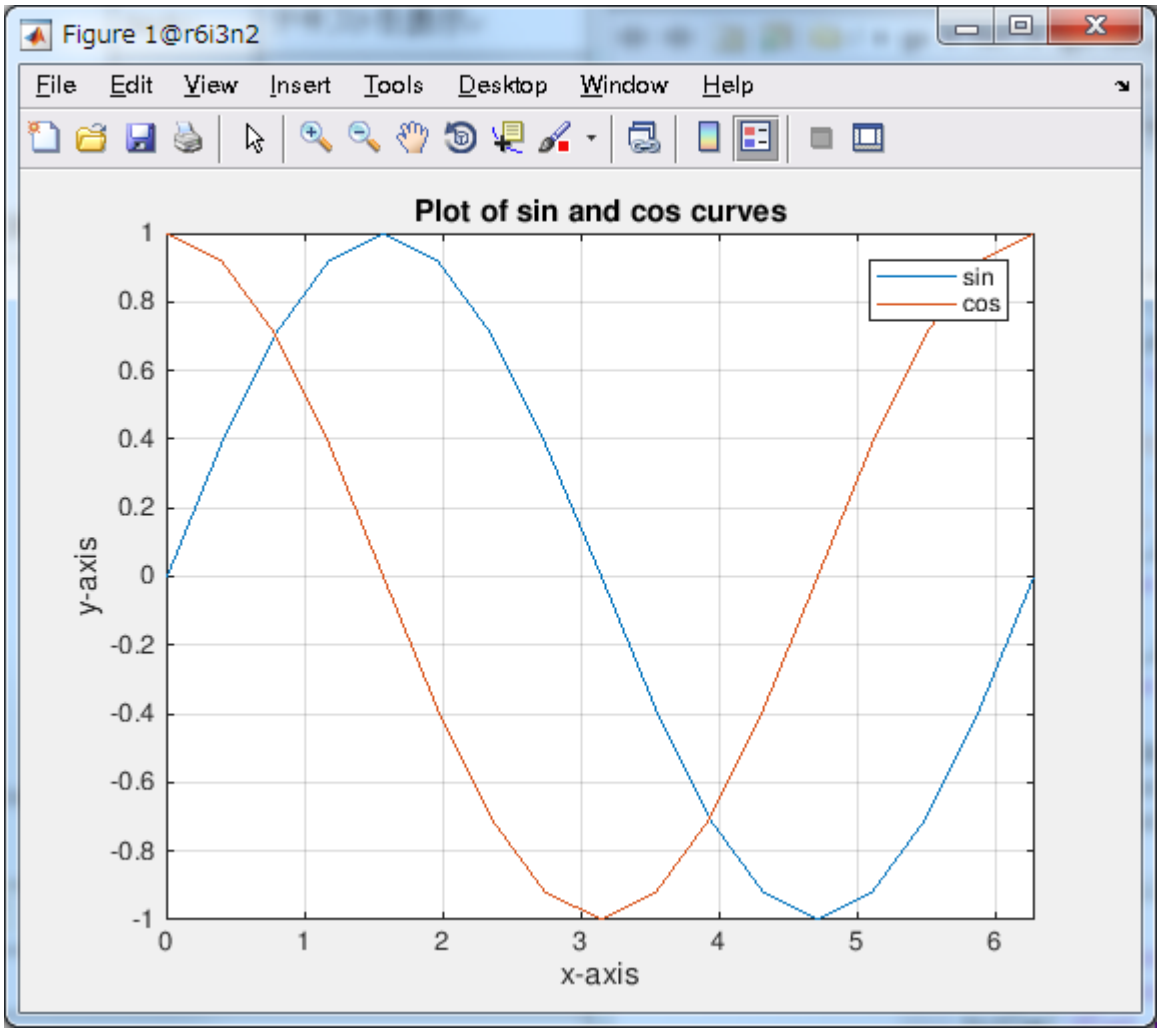
4.2.1. グラフの軸・注釈の設定

代表的な軸設定・注釈設定関数を以下に示します。

xlim	X軸範囲の変更
Xlabel	X軸ラベル
ylim	Y軸範囲の変更
ylabel	Y軸ラベル
zlim	Z軸範囲の変更
zlabel	Z軸ラベル
axis	軸範囲の変更
title	タイトル
grid	グリッド表示
legend	凡例
view	視点の変更
text	テキストを表示
colorbar	カラーバー
gtext	マウスを使ったテキスト表示

(例)Sinカーブ、Cosカーブの装飾付きプロット

```
>> x=[0:pi/8:2*pi];
>> y(:,1)=sin(x);
>> y(:,2)=cos(x);
>> plot(x,y)
>> xlim([0,2*pi])
>> grid
>> xlabel('x-axis')
>> ylabel('y-axis')
>> title('Plot of sin and cos curves')
>> legend('sin','cos')
```



4.2.2. グラフィックスの編集機能

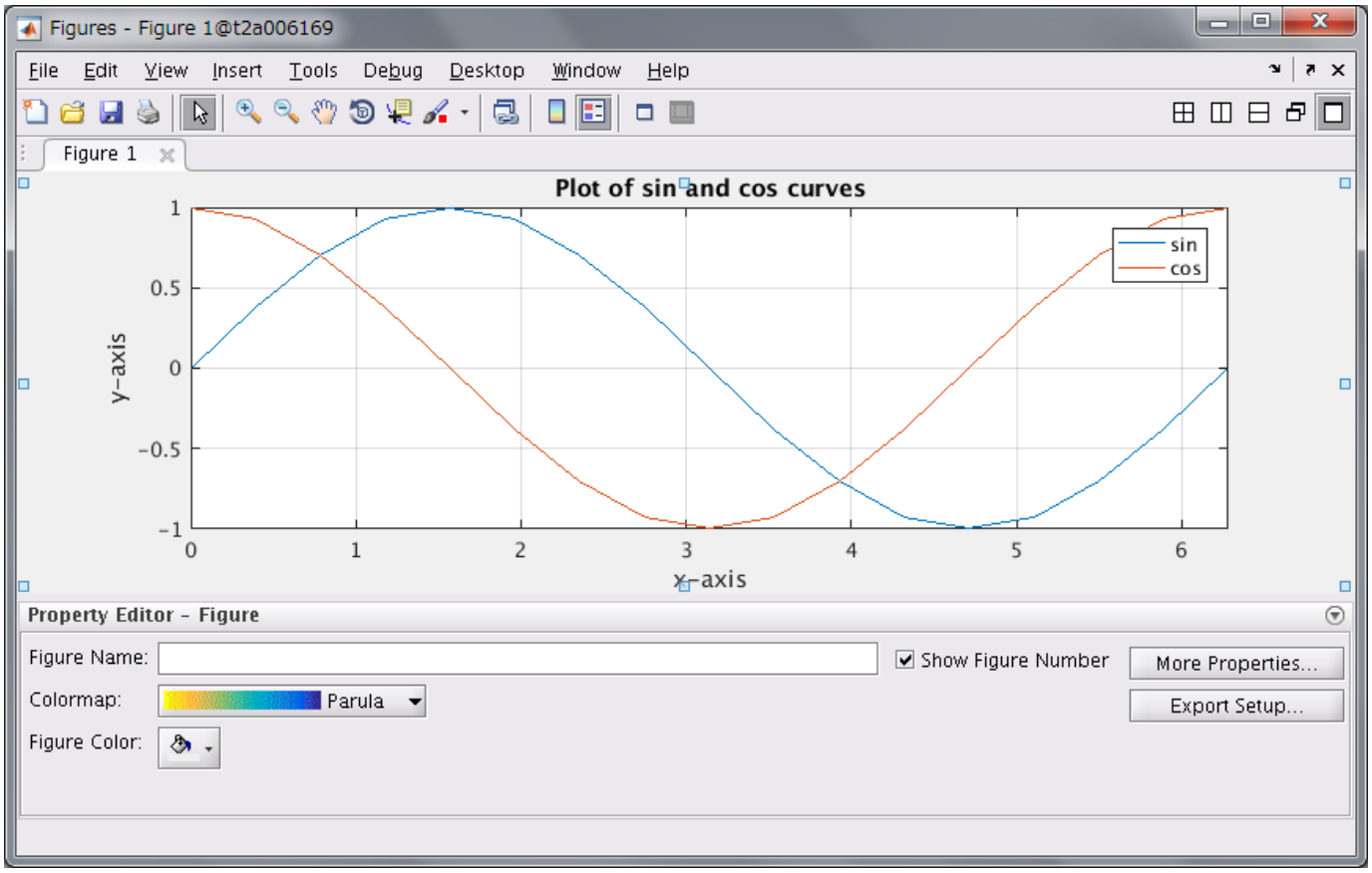
グラフィックスの編集を行う代表的な方法には次の2通りあります。

・プロパティエディタを利用する

・Plot Tool機能を利用する

プロパティエディタはMATLABのグラフィックス編集を行うGUIツールで、 基本的にマウス操作がメインになります。
これに対して、コマンドによる編集方法はキーボード入力がメインになります。

1. プロパティエディタの利用 プロパティエディタを起動するには、Figureウィンドウの「Edit」⇒「Figure Properties」を選択します。 Property Editorが起動します。変更したいプロパティを選択すると表示されるメニューがその都度変わります。
2. Plot Tool機能の利用


Figures - Figure 1@t2a006169
File Edit View Insert Tools Debug Desktop Window Help
Figure 1
Plot of sin and cos curves
sin
cos
y-axis
x-axis
1
0.5
0
-0.5
-1
0
1
2
3
4
5
6
Property Editor - Figure
Figure Name:
Show Figure Number
More Properties...
Colormap:
Parula
Export Setup...
Figure Color:

5. プログラミング

5.1. プログラミングの基本

これまでの処理では、単にコマンドや関数をコマンドウィンドウに直接入力して実行しました。 しかし、この方法では複数の処理をまとめて実行したいときや処理を行いたいときは不便です。 このような場合、M-ファイルと呼ばれるMATLABプログラムを作成します。 M-ファイルとは、コマンドや関数を実行したい順に記述したテキストファイル(拡張子　.m)です。 テキストエディタを使って、M-ファイルを作成すれば、他のMATLAB関数やコマンドと同じように利用することができます。 なお、MATLAB言語はインタプリタ型言語なので、M-ファイルの実行時にコンパイルやリンクという前処理は必要ありません。

5.1.1. プログラミングの基本的な流れ

1. テキストエディタを使ってM-ファイルを作成
2. コマンドウィンドウ、もしくは他のM-ファイルから作成したM-ファイルを実行

M-ファイルには下記の2種類の形式が存在します。

・スクリプトM-ファイル

・ファンクションM-ファイル

M-ファイルはテキストファイルなので、任意のテキストエディタを使用して編集することができます。 MATLABにはM-ファイルの編集に便利なエディタがありますので、これを使用することを推奨します。

5.1.2. M-ファイル編集エディタの起動方法

・「edit」コマンドを入力する

以下のコマンドを実行する

```
>> edit
```

・「File」メニューから選択する

「File」⇒「New」⇒「M-ファイル」を選択

5.2. スクリプトM-ファイル

スクリプトM-ファイルは以下のような機能をもっています。

・一連のコマンド・関数を連続的に処理することができる

スクリプトM-ファイルには特別な構文は必要ありません。 単純にテキストファイルの先頭行から順に処理内容を記述します。 実行は、次の(1)(4)で行います。

(1) エディタの起動

前節で述べたように起動します。

・「edit」コマンドを入力する

・「File」メニューから選択する

(2) スクリプトM-ファイルの作成

プログラムは以下のように記述します。 ここでは、例として、次の関数のグラフを作成する 処理プログラム「sample1.m」ファイルを作成します。

Y=0.1-0.3cos(x)+0.2cos(2x)
【sample1.m】

```
clear all
a=[0.1 -0.3 0.2];
x=-5:0.1:5;
y=a(1)+a(2)*cos(x)+a(3)*cos(2*x);
plot(x,y)
```

(3) スクリプトM-ファイルの保存

エディタの「File」メニューから「Save As」を選択し、「sample1.m」として保存します。
なお、ファイル名の保存には以下の制限があります。

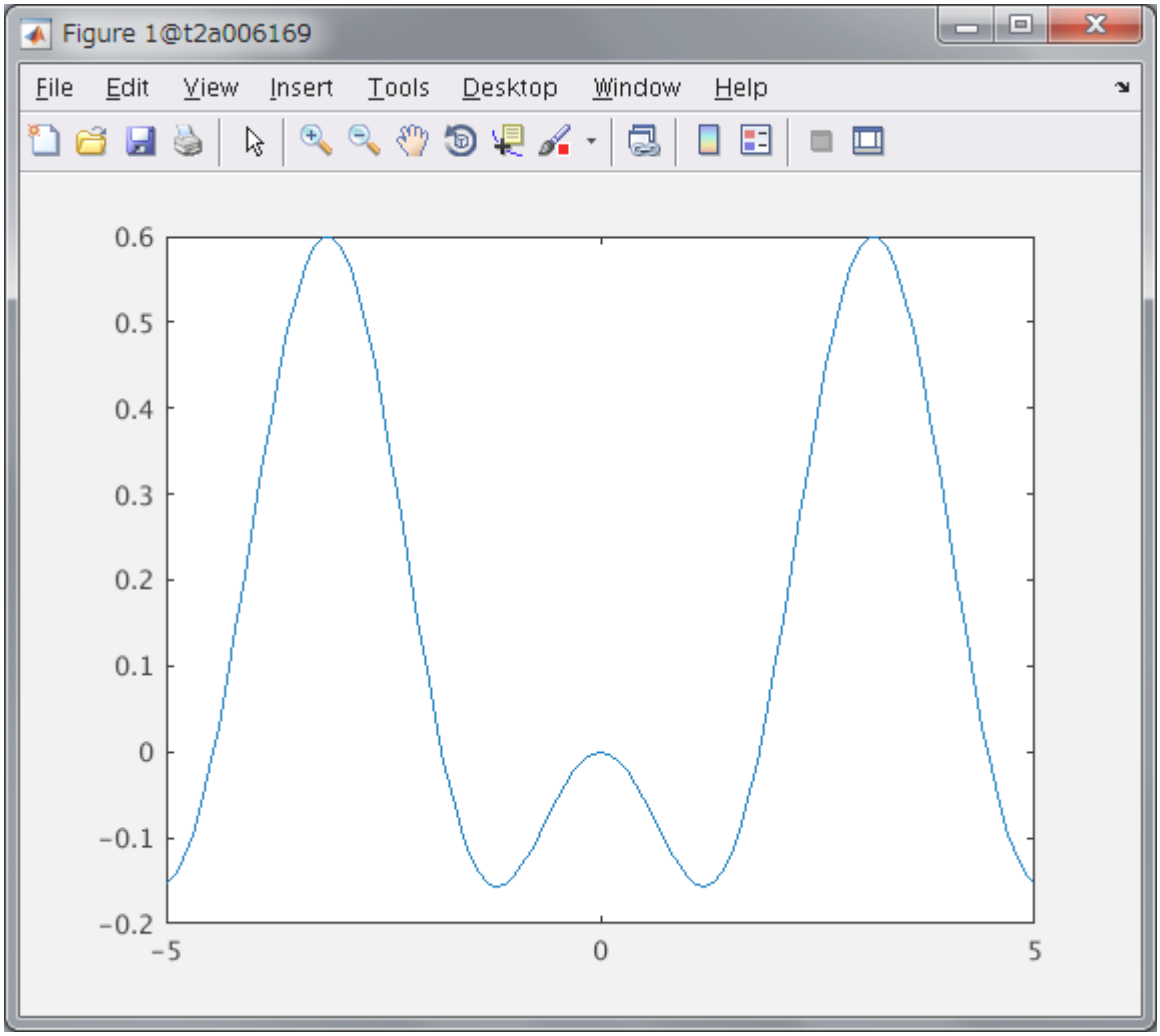
```
1:大文字・小文字は区別されます
2:ファイル名の文字制限は63文字です
3:数字及び演算子で始まるファイル名は使用できません
4:日本語文字列をファイル名に使用することはできません
5:関数・コマンド名と同じ名前にしないでください
6:予約変数と同じ名前にしないでください
```

(4) スクリプトM-ファイルの実行

スクリプトM-ファイルの実行はコマンドラインにファイル名を

```
>> sample1
```

と入力するか、エディタの「Run」ボタンを押す。 以下のグラフが表示されれば、成功です。



・コメントアウト

上記で作成したスクリプトM-ファイルにコメントを加えるには、 まず「%」を記述し、それ以降にコメント文を記述します。
「%」以降はコメントとみなし、MATLABは行の内容を無視し、実行しません。 ちなみに上記ファイルにコメント文を追記すると以下のようになります。

【sample1.m】

```
% ワークスペース内の全ての変数を消去する
clear all
% 係数aを定義する
a=[0.1 -0.3 0.2];
% -5から5において、0.1間隔でxを設定する
x=-5:0.1:5;
% y=a(1)+a(2)cos(x)+a(3)cos(2x)の計算
y=a(1)+a(2)*cos(x)+a(3)*cos(2*x);
% 結果をプロットする
plot(x,y)
```

5.3. ファンクションM-ファイル

ファンクションM-ファイルは以下のような機能をもっています。

・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することができる

(1) MATLAB関数

MATLABにおける関数は、数学における関数概念と同様に、入力と出力間の対応関係をして定義されています。 例えば、MATLABのsin関数について考えます。 y=sin(x)という式は、 xという変数を関数の入力値にとり、その値の正弦値を計算した結果を変数yに代入しています。 関数の入力値に用いる変数(この場合x)のことを入力変数、 関数の計算結果の出力先の変数(この場合y)のことを出力変数といいます。 MATLABではデフォルトで多くの関数が提供されていますが、 これに加えてユーザ定義の関数をプログラミングして使用することができます。 この関数機能をプログラミングしたM-ファイルのことをファンクションM-ファイルといいます。

(2) ファンクションM-ファイル構文

スクリプトM-ファイルには特別な構文は必要なく、 ファイル名もMATLABの変数名の規則を満たすものであれば、 自由な名前をつけることができました。 これに対して、ファンクションM-ファイルには次の2点の条件があります。

1.M-ファイルの1行目にfunction行を記述する(必須)

```
function [出力変数] = 関数名(入力変数)
```

2.関数名とM-ファイル名を同じにする(推奨)

関数名: sample_func ⇒ M-ファイル名　sample_func.m

また、MATLABでは、複数入力・複数出力の関数を作成することができます。 複数の場合は以下のように記述します。

```
function [y1,y2,y3,…] = sample(x1,x2,x3,…)
```

出力引数が1つの場合は、出力引数を大括弧[]で囲む必要はありません。

(3)ファンクションM-ファイルの作成と実行

例として、前節と同様に次の関数のグラフを作成する処理プログラムファイルを作成します。 今回は、スクリプトM-ファイルからファンクションM-ファイルを呼び出すような処理にします。

Y=0.1-0.3cos(x)+0.2cos(2x)
【sample2.m】

```
clear all
a=[0.1 -0.3 0.2];
x=-5:0.1:5;
y=func2(a,x);
```

【func2.m】

```
function y=func(a,x)
y=a(1)+a(2)*cos(x)+a(3)*cos(2*x);
plot(x,y)
```

実行方法は、sample2.mファイルとfunc2.mファイルを作成後、コマンド「sample2」を入力するか、 M-ファイルエディタでsample2.mファイルを開き、「Run」ボタンを押すかのどちらかです。 実行結果は前節と同様になります。

5.4. 制御構造

M-ファイルは原則として1行目から順に処理を実行すると前節まで述べました。 しかし、この処理を条件などにより変更できれば、より高度な処理を実現できます。 MATLABには、このようなプログラムを制御するための構文が用意されています。 ここでは、代表的な比較演算子、論理演算子、制御構文について説明します。

(1)比較演算子

比較演算子について、下表に示します。

==	eq	等しい
~=	ne	等しくない
<	lt	小さい
>	gt	大きい
<=	le	小さいか等しい
>=	ge	大きいか等しい

比較演算子は、後述する制御構文ifに付属する形で頻繁に用いられる。 比較演算子は2つの変数を比較し、その比較が正しい場合は1、そうでない場合は　を出力する。 例えば、a==bはaとbが等しいときに1を、そうでない場合には　を出力する。

```
>> a=4; b=4; c=(a==b)

c =

     1
```

(2)論理演算子

論理演算子について、下表に示します。

&	and	要素ごとの論理積
|	or	要素ごとの論理和
~	not	論理否定
xor		排他的論理和

(3)制御構文

制御構文には下表に示すものがあります。

if	条件分岐による処理選択
switch	多分岐選択処理
for	指定回数の繰り返し処理
while	不定回数の繰り返し処理
try/catch	例外処理(エラー処理)

それぞれの詳細について、説明します。

・if文

MATLABにおけるif文の構成は次のようになります。

```
if 条件1
  プログラムA
elseif 条件2
  プログラムB
else
  プログラムC
end
```

(例) if文サンプルプログラム

```
a=1;
if a<0
  b=1;
elseif a==0
  b=2;
elseif a<=2
  b=3;
else
  b=4;
end
```

このプログラムを実行するとbに3が代入される。 aが0以下のときはb=1、0のときはb=2、0より大きく 2以下のときはb=3、2より大きいときはb=4が代入される。

・switch文

switch文もif文と同様に条件分岐を実行するコマンドであり、構造は次のようになります。

```
switch a
  case m
    プログラムA
  case n
    プログラムB
  otherwise
    プログラムC
end
```

switchの直後には変数、または計算式が続きます。 上の例では変数aを指定している。 このaがcaseの直後に続く文と一致するとき、その後のプログラムを実行する。 上記例ではa==mのとき、プログラムAが実行され、a==nのとき、プログラムBが実行される。 どちらにも当てはまらない場合、otherwiseの後ろの文、つまりプログラムCが実行される。

(例) switch文サンプルプログラム

```
a=3;
 switch a
   case 1
     b=1;
   case 2
     b=2;
   case 3
     b=3;
   otherwise
     b=4;
 end
```

この場合は、aの値が3つ目のcase文に合致するのでbに3が代入される。

・for文

MATLABにおけるfor文はforとendに囲まれる部分を繰り返し実行する。

```
for [変数] = [ベクトル]
  % この部分が繰り返し実行される
end
```

(例) for文サンプルプログラム1(繰り返し回数 50回)

```
for n=1:50;

end
```

上記例では、n=1,2,3,...,50と変化しながらfor　end間のプログラムを実行します。
(例) for文サンプルプログラム2(繰り返し回数 11回)

```
for n=0:0.1:1;

end
```

上記例では、n=0,0.1,0.2,0.3,...,1と変化しながらfor　end間のプログラムを実行します。
(例) for文サンプルプログラム3(繰り返し回数 4回)

```
for n=[1 3 -1 4]

end
```

上記例では、n=1,3,-1,4と変化しながらfor　end間のプログラムを実行します。

・breakとcontinue

for文の繰り返し途中で計算を中止し、for文の外に抜け出すときはbreak文を用います。

(例) break文サンプルプログラム

```
a=0;
for n=1:100
  a=a+n;
  if a>100
    break
  end
end
```

上記例では、3行目でaにnが加算され、それが100より大きくなるとfor文を中断し、 次(8行目以降)へと進む。 for文の繰り返し中に、以降の計算をスキップし、 次の繰り返し計算に移るときはcontinue文を用います。

(例) continue文サンプルプログラム

```
a=0;
for n=1:100
  if rem(n,3)==0
    continue
  end
  a=a+n;
end
```

ここで用いているrem(a,b)はaをbで割った余りを出力します。 このプログラムはnが3のときは何もせず、次の繰り返しに進み、3の倍数でないときのみa=a+nを実行します。

・while文

for文では、繰り返し回数が明示されているのに対し、 while文はwhileの後ろに続く条件文を満たす間、繰り返し実行する。

```
while n<m
  % n<mが真である間、この部分が繰り返し実行される
end
```

上記例では、n<mが真(つまり1)の間はwhile内を繰り返し実行し、 繰り返す回数はそのwhile内のプログラムに依存します。
(例) while文サンプルプログラム1

```
n=1;
while n<=5
  disp('ここは5回実行される')
  n=n+1;
end
```

変数nを1から5まで変化させながら5回繰り返す。
(例) while文サンプルプログラム2

```
n=1;
while 1
  disp('ここは5回実行される')
  if n>=5
    break;
  end
  disp('ここは4回実行される')
  n=n+1;
end
```

whileの後ろの条件式を1に設定し(つまり、ここの条件は常に真なので、ここでwhile文が終わることはない)、 while内にあるif文で条件を満たしたときにbreak文でwhileから抜け出し繰り返しを中断する。

6. Parallel Computing Toolbox の利用

6.1. Parallel Computing Toolbox について

Parallel Computing Toolbox の主な機能は次の通りです。

- パラレル for ループ (parfor) によるマルチプロセッサでのタスク並列アルゴリズムの実行
- CUDA に対応した NVIDIA GPU のサポート
- ローカルのマルチコア デスクトップで 12 ワーカーまで起動可能
- 大規模データ セットの処理とデータ並列アルゴリズムに対応する分散配列および spmd (Single Program Multiple Data) 構文

複数のワーカーによる並列処理を行うことで計算時間が短縮するメリットがあります。 また、GPU計算がサポートされているため、GPU を使用した演算が可能です。
詳細な内容は、Mathworks 社のホームページや、MATLAB のヘルプ機能をご参照ください。

製品紹介のページ

ドキュメンテーション

6.2. 並列処理

ここでは、Parallel Computing Toolbox による並列処理の 基本的な利用方法を説明します。
次のような、sin カーブをプロットするコードについて考えます。

```
for i=1:1024
  A(i) = sin(i*2*pi/1024);
end
plot(A)
```

このコードを並列処理する方法を説明します。

並列処理を行うためには、ワーカーを起動しておく必要があります。 ここでワーカーとは、MATLAB セッションとは別に動作するMATLAB 計算エンジンのプロセスのことで、 ワーカーを使用する関数を用いることで各ワーカープロセスに処理を割り振ることができます。 ワーカーの起動には parpool 関数を使用します。

```
>> parpool('local', 4)
Starting parallel pool (parpool) using the 'local' profile ... connected to 4 workers.
```

第2引数の「4」は起動するワーカーの数で、最大「12」まで指定できます。
並列処理を行うようにコードの修正を行います。 違いは、「for」の代わりに「parfor」を用いることだけです。

```
parfor i=1:1024
  A(i) = sin(i*2*pi/1024);
end
plot(A)
```

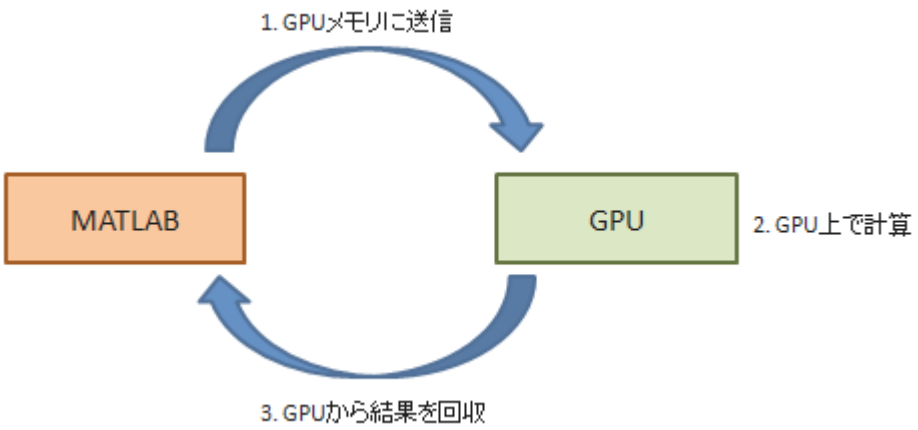
ワーカープロセスを終了する場合は、次のコマンドを実行します。

```
>> poolobj = gcp('nocreate');
>> delete(poolobj)
Parallel pool using the 'local' profile is shutting down.
```

6.3. GPU を使用した演算

MATLAB R2010bから「Parallel Computing Toolbox」のGPUコンピューティング対応されています。 GPU とのデータのやり取りを意識する必要があり 主な手順としては次のようになります。

1. GPUメモリに送信
2. GPU上で計算
3. GPUから結果を回収



GPU 演算の流れを実際の計算例を使って示します。

この例ではGPU のメモリ上にデータを送信する関数「GPUArray」と、 GPU 上の結果をメインメモリへ回収する関数「gather」を用いています。 また、fft2 関数は GPU 計算に対応しており使用例を示します。

```
>> N = 6;
>> M = magic(N)          ← 行列 M を作成

M =

    35     1     6    26    19    24
     3    32     7    21    23    25
    31     9     2    22    27    20
     8    28    33    17    10    15
    30     5    34    12    14    16
     4    36    29    13    18    11

>> G1 = gpuArray(M);    ← GPU メモリに送信

>> G2 = fft2(G1);       ← fft2 を GPU 上で実行

>> M1 = gather(G2)      ← 結果をメインメモリを回収

M1 =

   1.0e+02 *

   6.6600 + 0.0000i   0.0000 + 0.0000i   0.0000 - 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.7200 + 0.3118i  -0.2700 - 0.4677i  -0.3600 + 0.6235i   0.5400 - 0.0000i  -0.6300 - 0.4677i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.5400 + 0.3118i   0.0000 + 0.0000i   0.0000 - 0.2078i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.4500 + 1.0912i   1.3500 + 0.4677i   1.2600 + 0.0000i   1.3500 - 0.4677i   0.4500 - 1.0912i
   0.0000 + 0.0000i   0.0000 - 0.0000i   0.0000 + 0.2078i   0.0000 + 0.0000i   0.5400 - 0.3118i   0.0000 + 0.0000i
   0.0000 + 0.0000i  -0.6300 + 0.4677i   0.5400 + 0.0000i  -0.3600 - 0.6235i  -0.2700 + 0.4677i   0.7200 - 0.3118i

>> M2 = fft2(M)          ← CPU のみで計算した場合。GPU での計算結果と同じになることが確認できる。

M2 =

   1.0e+02 *

   6.6600 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 - 0.0000i   0.7200 + 0.3118i  -0.2700 - 0.4677i  -0.3600 + 0.6235i   0.5400 + 0.0000i  -0.6300 - 0.4677i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.5400 + 0.3118i   0.0000 + 0.0000i   0.0000 - 0.2078i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.4500 + 1.0912i   1.3500 + 0.4677i   1.2600 + 0.0000i   1.3500 - 0.4677i   0.4500 - 1.0912i
   0.0000 - 0.0000i   0.0000 + 0.0000i   0.0000 + 0.2078i   0.0000 + 0.0000i   0.5400 - 0.3118i   0.0000 + 0.0000i
   0.0000 + 0.0000i  -0.6300 + 0.4677i   0.5400 + 0.0000i  -0.3600 - 0.6235i  -0.2700 + 0.4677i   0.7200 - 0.3118i
```

なお、GPU 対応している関数の一覧を得るには次のコマンドを実行します。

```
>> methods('gpuArray')

Methods for class gpuArray:

abs                 csch                im2int16            lsqr                sec
accumarray          ctranspose          im2single           lt                  secd
acos                cummax              im2uint16           lu                  sech
     : (以下略)
```

個々の関数のヘルプを参照するには次のコマンドを実行します。

```
>> help gpuArray/functionname
```

mtimes関数の場合は次のようになります。

```
>> help gpuArray/mtimes
 * Matrix multiply for gpuArray
    C = A * B
    C = MTIMES(A,B)

    64-bit integers are not supported.

    Example:

        N = 1000;
        A = gpuArray.rand(N)
        B = gpuArray.rand(N)
        C = A * B

    See also MTIMES, GPUARRAY.
```

6.4. 複数ノードの使用

Warning

matlabのプラグインの仕様上、複数ノードでの実行はf_nodeでしか動作しません。

qrshまたはiqrshで計算ノードを確保後、

```
. /apps/t3/sles12sp2/uge/latest/default/common/settings.sh
```

を実行してからmatlabを起動して下さい。
matlab内で計算用のジョブをさらに発行するため、matlabを起動する際に確保する資源は`-l s_core=1`などの小さいもので構いません。

複数ノードを使用して並列計算を行うには以下の設定を行う必要があります。

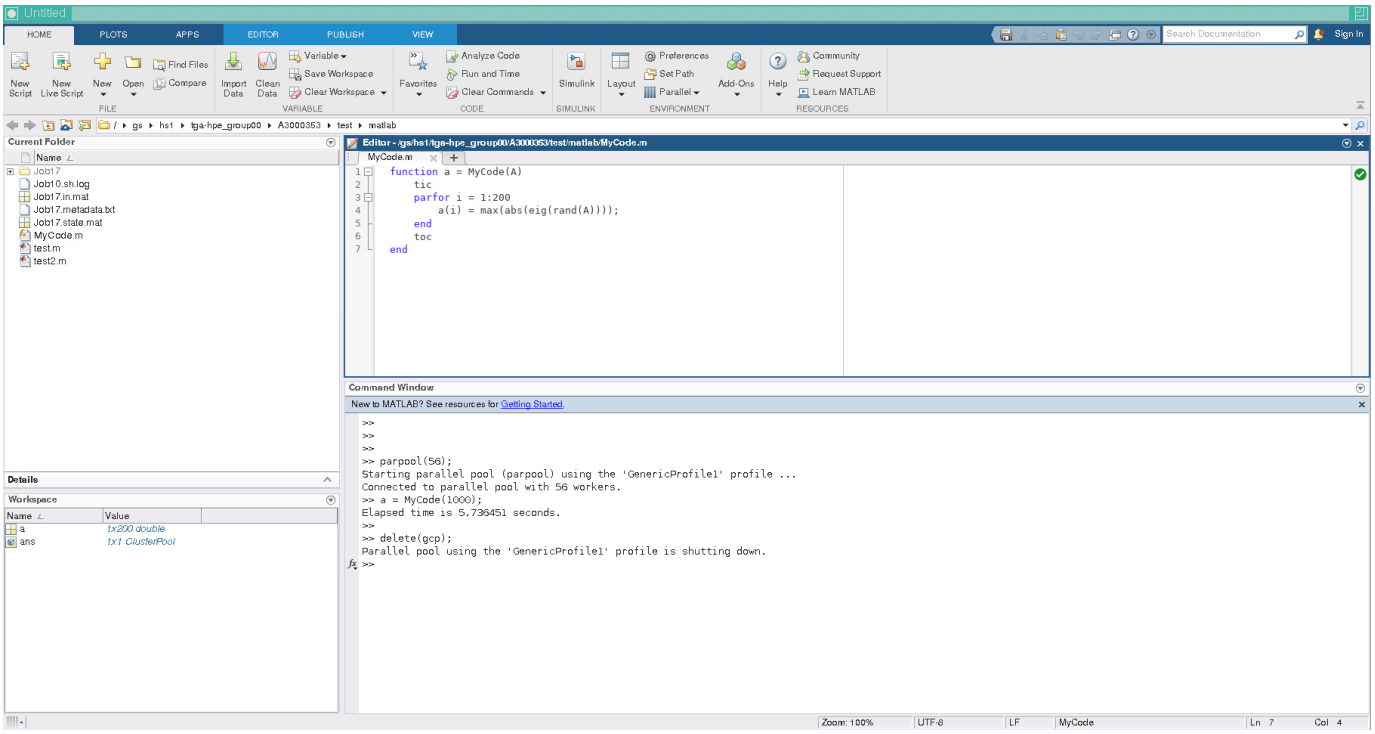
- Home -> Parallel -> Create and Manage Clustersを選択
- Add Cluster Profile -> Genericを選択
- Editを選択し、SCHEDULER PLUGINの設定でPluginScriptLocationに`/apps/t3/sles12sp2/isv/matlab/plugins/matlab-parallel-gridengine-plugin`を入力
- AdditionalPropertiesのNameに`AdditionalSubmitArgs`、Valueに`-g <グループ名> -l h_rt=<時間> -l f_node=<ノード数>`を入力
- Validateを選択し、Numbers of workers to useにノード数x28(例:f_node=2の場合は56)を入力し実行、設定したノード数で実行できるか確認。全てPassedになっていれば正しく設定されている
- Validateをパスしたら、Parallel -> Select Parallel Environment -> CLUSTERから先ほど作成したプロファイルを指定

```
parpool(N)
```

でジョブが投入され、ジョブが流れると

```
Starting parallel pool (parpool) using the 'GenericProfile1' profile ...
Connected to parallel pool with N workers.
```

と表示され、複数ノードでの実行が可能となります。



改訂履歴

改定日付	内容
2024/01/16	「6.4. 複数ノードの使用」の修正
2024/01/12	「6.4. 複数ノードの使用」を追加
2020/04/30	X転送に関する記載を修正
2019/08/19	mkdocs版作成
2017/03/14	初版