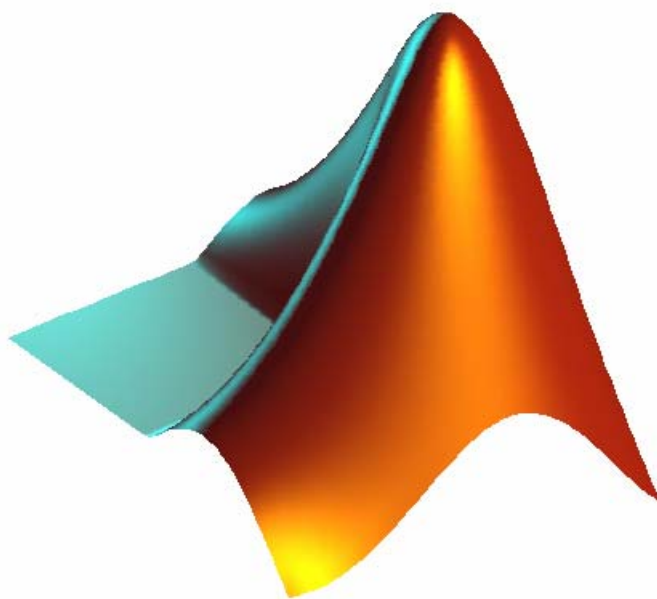


MATLAB Compiler チュートリアル



目次

はじめに	i
1 MATLAB Compiler 概要	1
1.1 MATLAB Compiler とは	1
1.2 特徴	2
1.3 使用できる M-ファイル	2
1.4 作成可能なアプリケーション	2
1.5 アプリケーション動作環境	3
1.6 アプリケーション動作の仕組み	3
1.7 アプリケーション開発フロー	4
2 環境設定	5
2.1 サポートコンパイラの確認	5
2.2 コンパイラの設定	5
3 実行ファイル作成	7
3.1 実行ファイル作成概要	7
3.2 実行ファイル動作のための設定	7
3.3 例題 1 文字列表示	7
3.4 例題 2 行列計算	9
3.5 例題 3 プロット表示	11
3.6 例題 4 MATLAB GUI	13
3.7 コマンドプロンプトの非表示設定	16
4 ライブラリ作成	18
4.1 ライブラリ作成概要	18
4.2 ライブラリ動作のための設定	18
4.3 ライブラリの呼び出し	18
4.4 例題 1 C 共有ライブラリ	21
4.5 例題 2 C++共有ライブラリ	25
4.6 Visual C/C++からの共有ライブラリ利用	28
5 アプリケーションの配布	32
5.1 配布可能な環境	32
5.2 配布ファイル	32
5.3 配布環境での作業	32
付録 MATLAB ヘルプ・情報リソース	33

はじめに

本チュートリアルは MATLAB Compiler を用いたアプリケーション開発を行いたい方を対象に、MATLAB Compiler の使用方法およびアプリケーションの作成・配布方法について説明しています。MATLAB Compiler 使用時の簡易マニュアルとしてお役立ていただければ幸いです。

なお、本チュートリアルは Windows 環境にインストールされた MATLAB Compiler 4.4 (R2006a) をベースに作成されています。他プラットフォームおよび他バージョンの MATLAB Compiler をご利用の場合、本チュートリアルを適用できない可能性がありますのでご注意ください。

■ Unix/Linux/Mac 環境について

本チュートリアルは使用 OS として Windows のみを対象としています。Unix/Linux/Mac 環境における MATLAB Compiler に関する操作は、Windows のそれと以下の点で異なります。

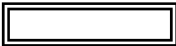

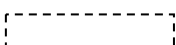
- アプリケーション動作のための環境設定
- MATLAB Component Runtime のインストール方法

Unix/Linux/Mac 環境における操作の詳細につきましては、「MATLAB Compiler User's Guide」マニュアルを参照してください。

■ 使用製品

製品名	バージョン番号	リリース番号
MATLAB	7.2.0	R2006a
MATLAB Compiler	4.4	R2006a

■ 枠線

枠線	内容
	MATLAB コマンドウィンドウ
	コマンドプロンプト
	テキストファイル (M-ファイル・C/C++ プログラム)

■ 略語

略語	内容
<matlabroot>	MATLAB インストールフォルダ

MATLAB インストールフォルダは、MATLAB コマンドウィンドウに次のコマンドを入力すれば確認することができます。

```
>> matlabroot
```

■ 例題ファイルの実行方法

example.zip を適当なフォルダに解凍後、MATLAB カレントディレクトリを example フォルダにしてから実行してください。本チュートリアルでは、C ドライブ直下に example.zip を解凍し、MATLAB カレントディレクトリが C:\example になっている状態を想定しています。

1 MATLAB Compiler 概要

1.1 MATLAB Compiler とは

MATLAB Compiler は、MATLAB 言語プログラムファイル（M-ファイル）からそれと等価な動作をするスタンドアロンアプリケーションを作成する機能を提供します。作成したアプリケーションは MATLAB Component Library と呼ばれるランタイムライブラリをインストールすれば、スタンドアロン環境（MATLAB がインストールされていない環境）上で動作します。MATLAB Compiler を利用することにより、MATLAB 言語による簡易プログラミングや豊富な数学・グラフィックスライブラリ・オプション製品の利用など、MATLAB のメリットを活用したアプリケーション開発が可能となります。

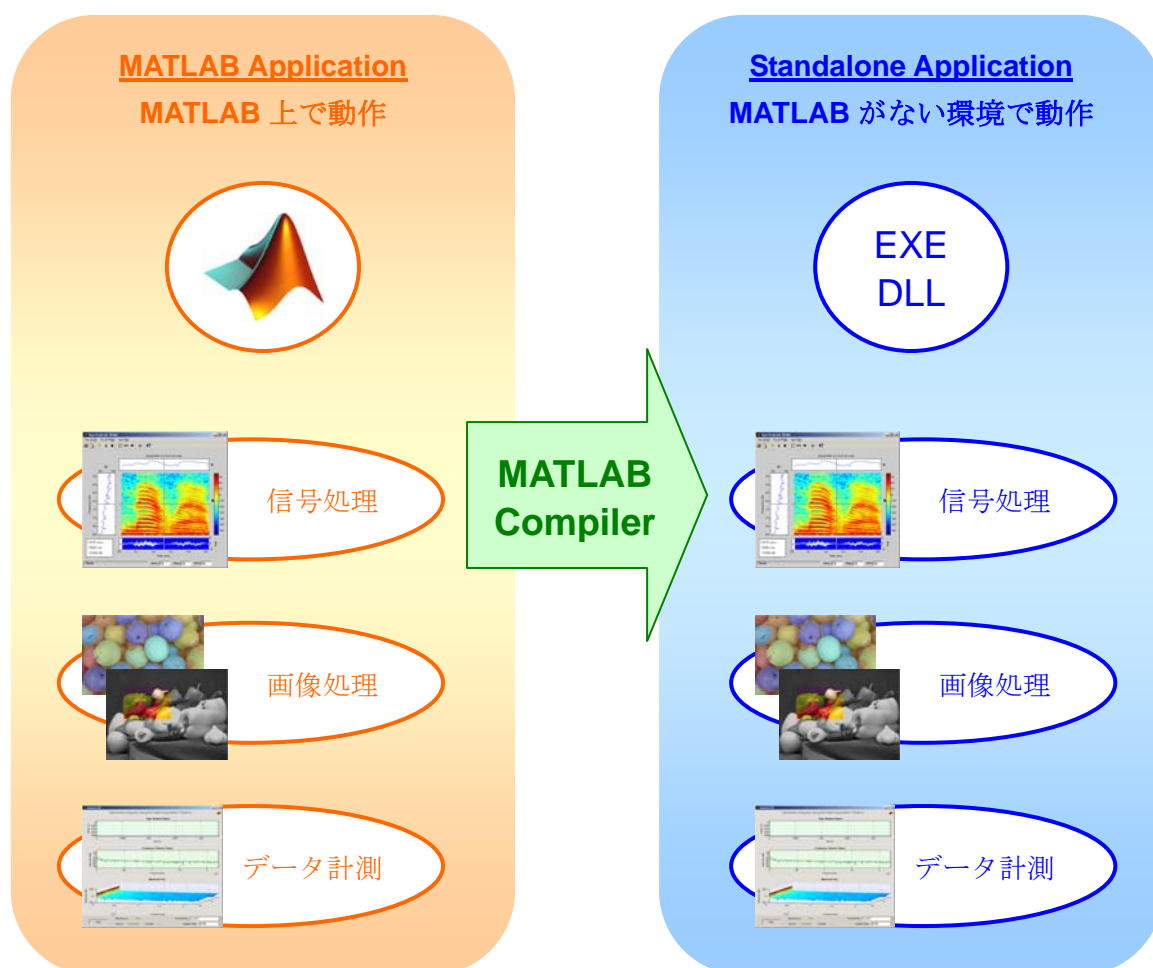


図 1-1 MATLAB Compiler 概念図

1.2 特徴

MATLAB Compiler の主な特徴は次の通りです。

- M-ファイルからそれと等価な動作をするスタンドアロンアプリケーション（実行ファイルおよびライブラリ）を作成。
- ほぼ全ての MATLAB 言語・データタイプ・関数をサポート。
- 各種 Toolbox 関数をサポート(*)。
- スタンドアロンアプリケーションの動作速度は MATLAB とほぼ同じ。
- スタンドアロンアプリケーションの配布には 2 次料金が不要（製品料金のみで複数ユーザに配布可能）。
- MATLAB Builder 製品を追加することにより、各製品・環境向けのソフトウェアコンポーネントを作成可能。

(*) 下記 URL にサポート Toolbox および非サポート Toolbox の一覧表が掲載されています。

- MATLAB Compiler サポート Toolbox
http://www.mathworks.com/products/compiler/compiler_support.html
- MATLAB Compiler 非サポート Toolbox
http://www.mathworks.com/products/ineligible_programs/

1.3 使用できる M-ファイル

MATLAB Compiler を用いてスタンドアロンアプリケーションを作成できる M-ファイルはファンクション M-ファイル（function 行から始まる M-ファイル）のみとなっています。スクリプト M-ファイルを使用する場合は、先頭行に function 行を追加してファンクション化する必要があります。

1.4 作成可能なアプリケーション

MATLAB Compiler で作成可能なスタンドアロンアプリケーションは次の通りです。

- 実行ファイル（EXE）
- ダイナミックリンクライブラリ（DLL）
- 特定製品・環境向けのソフトウェアコンポーネント（MATLAB Builder 製品が必要）

1.5 アプリケーション動作環境

スタンドアロンアプリケーションを作成した PC 上では、追加作業無しでアプリケーションが動作します。その他の PC では、MCRInstaller.exe を用いて MATLAB Component Runtime をインストールすることにより、スタンドアロンアプリケーションが動作するようになります。

1.6 アプリケーション動作の仕組み

MATLAB Compiler で作成したスタンドアロンアプリケーションは、主に次のファイルから構成されています。

- スタンドアロンアプリケーション本体 (EXE/DLL)
- Component Technology File (拡張子: ctf)

Component Technology File (CTF)とは、アプリケーション動作に必要な MATLAB 関連ファイル (M-ファイル/MEX-ファイル等) を圧縮したファイルです。ただし、M-ファイルはアルゴリズム保護のため、CTF 作成の際に暗号化されます。CTF はアプリケーション初回実行時に“アプリ名_mcr”フォルダに解凍され、MATLAB のサブセットである MATLAB Component Runtime (MCR)上で動作します。これに対して、アプリケーション本体は CTF を呼び出すインターフェースとして機能します。イメージとしては、「スタンドアロンアプリケーションが MATLAB の代わりに MCR 上で CTF (≒M-ファイル) を動作させている」とお考えください。下図はこのアプリケーション動作の様子を示した概念図です。

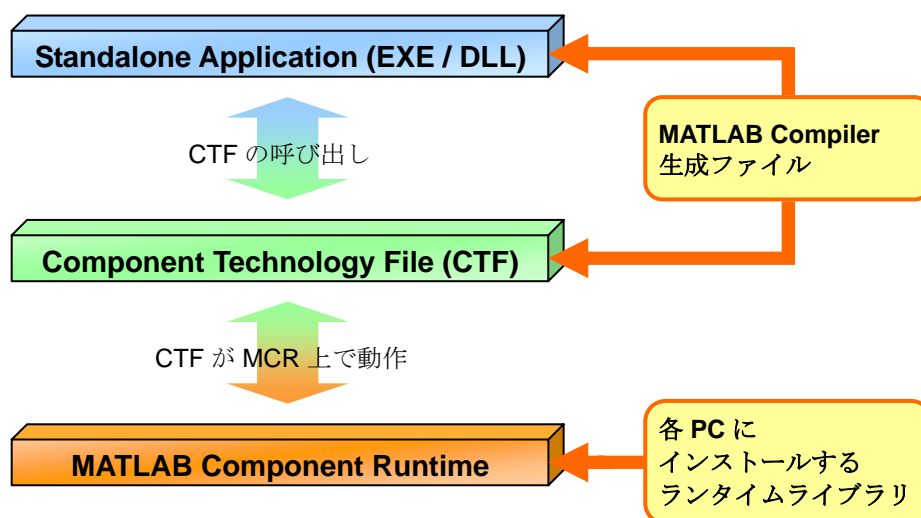


図 1-2 MATLAB Compiler 作成アプリケーション動作の仕組み

1.7 アプリケーション開発フロー

MATLAB Compiler を用いたアプリケーション開発の主なフローは下図の通りです。なお、ライブラリを作成した場合は、動作検証時にライブラリを呼び出すメインプログラムの検証も行う必要があります。

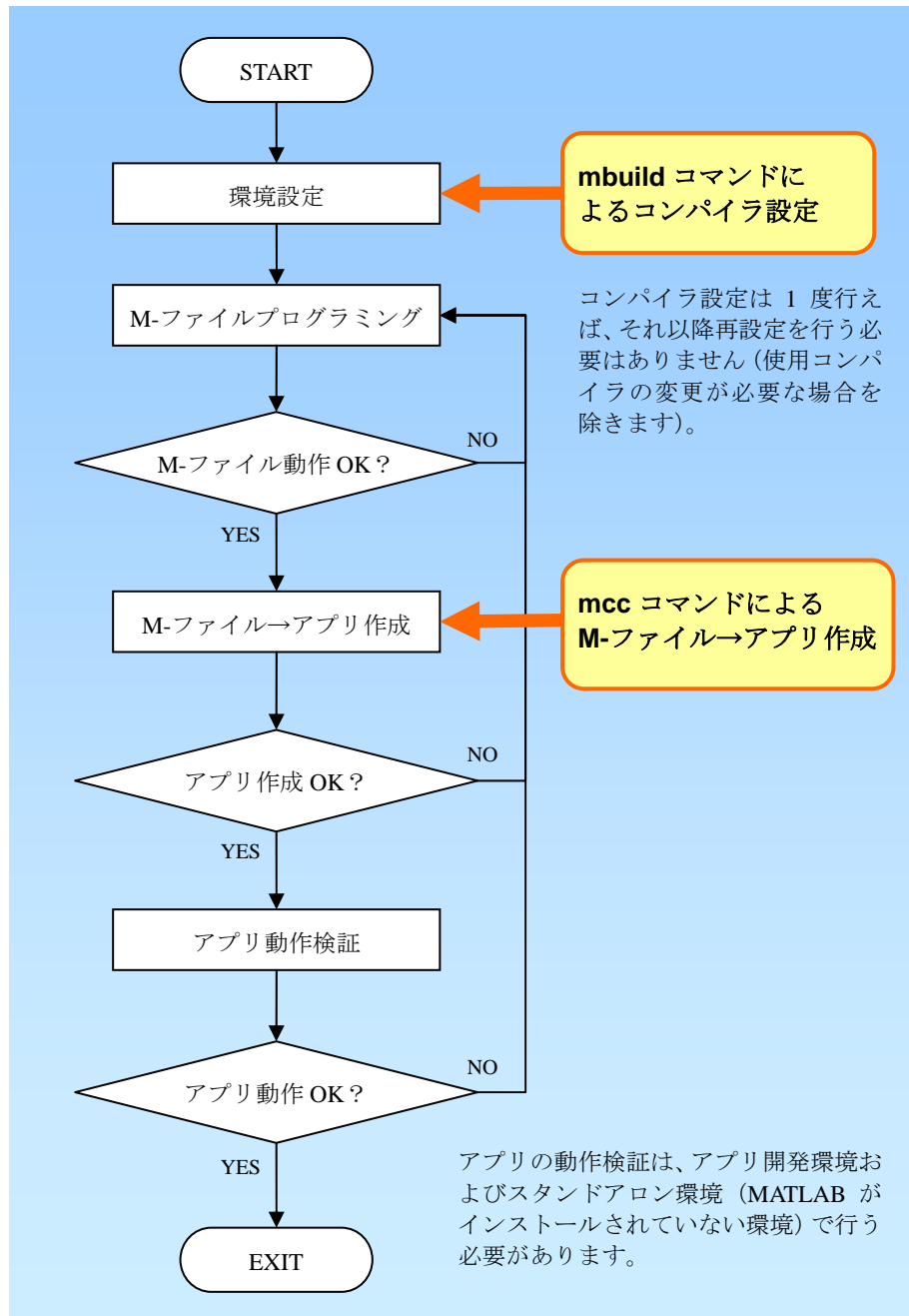


図 1-3 MATLAB Compiler を用いたアプリケーション開発フロー

2 環境設定

2.1 サポートコンパイラの確認

MATLAB Compiler がサポートするコンパイラは、下記 URL に掲載されています。

- MATLAB Compiler サポートコンパイラ
<http://www.mathworks.com/support/tech-notes/1600/1601.html>

MATLAB Compiler 使用前に、サポートコンパイラがインストールされているかどうかご確認ください。本チュートリアルでは、一部例題を除き Windows 版 MATLAB 7.2.0 (R2006a) に付属する C コンパイラ (Lcc C 2.4.1) を使用します。

2.2 コンパイラの設定

M-ファイルからアプリケーションを作成するためには、まず mbuild コマンドを用いてコンパイラの設定を行う必要があります。mbuild コマンドによるコンパイラの設定方法は次の通りです。

- ① MATLAB コマンドウィンドウに次のコマンドを入力します。

```
>> mbuild -setup
```

- ② 次のメッセージが表示されたら、<y>と入力して Enter キーを押します。

```
Please choose your compiler for building standalone MATLAB
applications:

Would you like mbuild to locate installed compilers [y]/n?
```

- ③ コンパイラの一覧が表示されますので、使用したいコンパイラの左横にある数字を入力してから Enter キーを押します。ここでは、Lcc C 2.4.1 を選択した例を示します。

```
Select a compiler:
[1] Lcc C version 2.4.1 in C:\PROGRAM FILES\MATLAB\R2006A\sys\lcc
[2] Microsoft Visual C/C++ version 7.1 in C:\Program Files\Microsoft
Visual Studio .NET 2003
[3] Microsoft Visual C/C++ version 6.0 in C:\Program Files\Microsoft
Visual Studio

[0] None

Compiler: 1
```

- ④ 次のメッセージが表示されたら、<y>と入力して Enter キーを押します。

```
Please verify your choices:

Compiler: Lcc C 2.4.1
Location: C:\PROGRAM FILES\MATLAB\R2006A\sys\lcc

Are these correct?([y]/n):
```

今回の MATLAB 起動以降も同じコンパイラ設定が反映されますので、上記の設定作業は一度だけ実行していただければ OK です。なお、使用コンパイラを変更したい場合は、再度同じ手順を実行してください。

3 実行ファイル作成

3.1 実行ファイル作成概要

ファンクション M-ファイルで作成されたアプリケーションをスタンドアロン環境上で動作させたい場合は、その M-ファイルから実行ファイル（EXE）を作成します。

実行ファイルを作成するコマンドは次の通りです。

```
>> mcc -m M-ファイル名.m
```

上記コマンドでカレントディレクトリに実行ファイル “M-ファイル名.exe” および “M-ファイル名.ctf” が生成されます。なお、mcc コマンドを使用する際、対象となる M-ファイルが MATLAB カレントディレクトリ、または MATLAB サーチパスに登録されたフォルダに存在している必要があります。

3.2 実行ファイル動作のための設定

MATLAB Compiler で作成した実行ファイルを動作させるためには、CTF ファイルが下記フォルダのどちらかに存在する必要があります。

- 実行ファイルと同じフォルダ
- 環境変数 Path に登録されているフォルダ

3.3 例題 1 文字列表示

myhello.m は文字列 “Hello World” をコマンドウィンドウに表示するファンクション M-ファイルです。

```
myhello.m
1 function myhello
2 % 文字列“Hello World”を表示するファンクション M-ファイル
3
4 disp('Hello World'); % “Hello World”の表示
```

MATLAB 上での myhello.m の実行結果は次の通りです。

```
>> myhello  
Hello World
```

myhello.m から実行ファイルを作成するために、次のコマンドを入力します。

```
>> mcc -m myhello.m
```

実行ファイルの作成に成功すると、カレントディレクトリに myhello.exe および myhello.ctf が生成されます。次に、myhello.exe の動作確認を行うためにコマンドプロンプトを起動します。コマンドプロンプトの起動方法は次の通りです。

- Windows 2000 の場合
[スタート] ボタン → [プログラム] → [アクセサリ] → [コマンドプロンプト]
- Windows XP の場合
[スタート] ボタン → [すべてのプログラム] → [アクセサリ] → [コマンドプロンプト]

コマンドプロンプトを起動することができたら、次に cd コマンドを用いて myhello.exe が存在するフォルダに移動します。例えば、myhello.exe が存在するフォルダが C:\example の場合、コマンドプロンプトに下記コマンドを入力します。

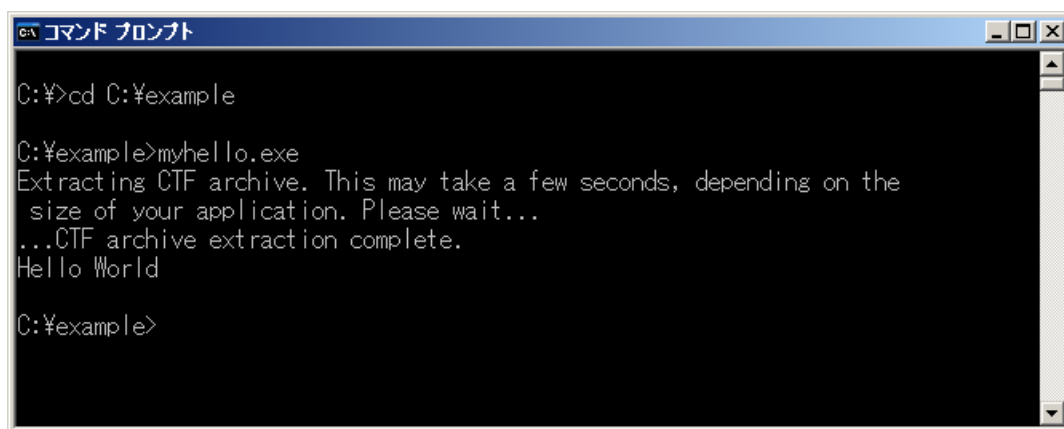
```
cd C:\example
```

フォルダの移動を終えたら、下記コマンドを入力して myhello.exe を実行します。

```
myhello.exe
```

myhello.exe を実行すると、myhello.ctf の解凍メッセージが表示された後、コマンドプロンプトに文字列 “Hello World” が表示されます。myhello.exe が存在するフォルダをエクスプローラで開いてみると、myhello.ctf を解凍したフォルダ（myhello_mcr フォルダ）が新たに作成されていることを確認できます。次の図は、myhello.exe を実行した際のコマンドプ

コンソールをキャプチャした画像です。



```
C:\example>cd C:\example

C:\example>myhello.exe
Extracting CTF archive. This may take a few seconds, depending on the
size of your application. Please wait...
...CTF archive extraction complete.
Hello World

C:\example>
```

図 3-1 myhello.exe の実行結果

なお、CTF の解凍はアプリ初回実行時のみですので、myhello.exe を再度実行すると今度は解凍メッセージが表示されずに文字列 “Hello World” が表示されます。

3.4 例題 2 行列計算

mymagic.m は n 行 n 列の魔方陣行列の各列の和を求めるファンクション M-ファイルです。魔方陣行列のサイズ n は入力引数として与えられます。なお、魔方陣行列とは、縦・横・斜めの要素の総和が全て等しい行列のことを指します。

```
mymagic.m
1  function mymagic(n)
2      % n 行 n 列の魔方陣行列の各列の和を求めるファンクション M-ファイル
3
4      % 入力引数 n が文字データの場合、それを数値データに変換
5      if ischar(n)
6          n = str2num(n);
7      end
8
9      M = magic(n)      % n 行 n 列の魔方陣行列の作成
10     S = sum(M)         % 各列の和の計算
```

コマンドプロンプトから渡される入力引数は、最終的には全て文字データとして扱われ

るため、mymagic.m では 5～7 行目で引数 n を数値データに変換する処理を行っています。

MATLAB 上での mymagic.m の実行結果は次の通りです。入力値が数値データ (4) と文字データ ('4') のどちらであっても、同じ結果が得られます。

```
>> mymagic(4)
M =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1

S =
    34    34    34    34

>> mymagic('4')
M =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1

S =
    34    34    34    34
```

mymagic.m から実行ファイルを作成するために、次のコマンドを入力します。

```
>> mcc -m mymagic.m
```

実行ファイルの作成に成功すると、カレントディレクトリに mymagic.exe および mymagic.ctf が生成されます。コマンドプロンプトを開いて、mymagic.exe の動作確認を行ってください。次の図は、mymagic.exe を実行した際のコマンドプロンプトをキャプチャした画像です。

```
コマンド プロンプト
C:\example>mymagic.exe 4
Extracting CTF archive. This may take a few seconds, depending on the
size of your application. Please wait...
...CTF archive extraction complete.
M =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
S =
    34    34    34    34

C:\example>
```

図 3-2 mymagic.exe の実行結果

3.5 例題 3 プロット表示

myplot.m は正弦波を 2 次元プロットするファンクション M-ファイルです。

```
myplot.m
1 function myplot
2 % 正弦波を 2 次元プロットするファンクション M-ファイル
3
4 t = 0 : pi / 20 : 4 * pi; % 時間データの作成
5 y = sin(t); % 正弦波データの作成
6
7 plot(t, y) % 正弦波のプロット
8 axis tight % 座標軸をデータ範囲に合わせる
9 grid on % 目盛りの追加
```

MATLAB 上で myplot.m を実行すると、次の Figure ウィンドウが表示されます。

```
>> myplot
```

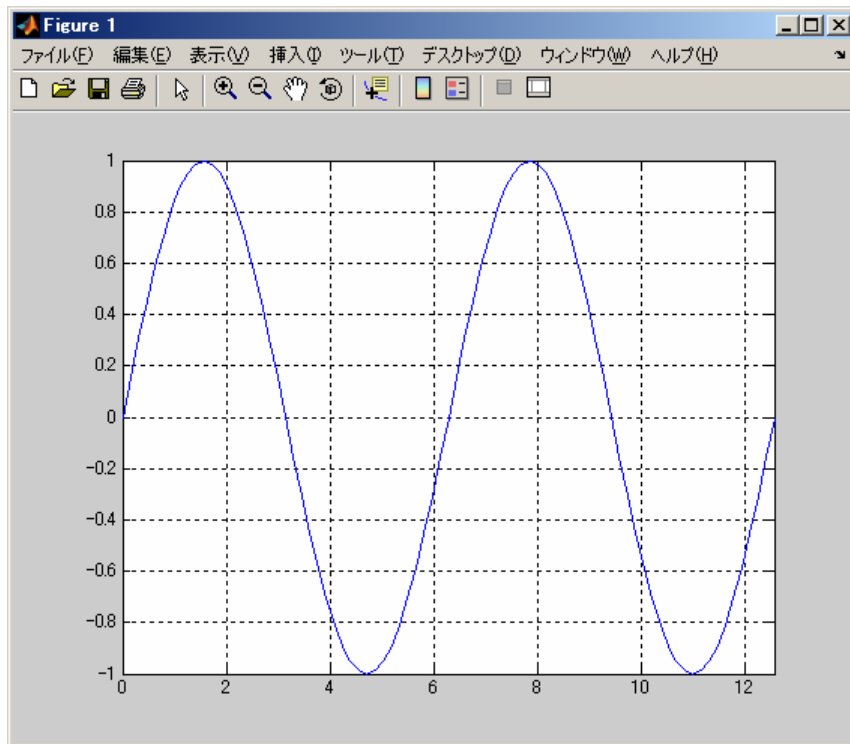



図 3-3 myplot.m の実行結果

myplot.m から実行ファイルを作成するために、次のコマンドを入力します。

```
>> mcc -m myplot.m
```

実行ファイルの作成に成功すると、カレントディレクトリに myplot.exe および myplot.ctf が生成されます。コマンドプロンプトを開いて、myplot.exe の動作確認を行ってください。図 3-3 と同じ Figure ウィンドウが表示されます。

3.6 例題 4 MATLAB GUI

MATLAB には、GUIDE と呼ばれる GUI 作成機能が用意されています。GUIDE を利用することにより、MATLAB 上で動作する各種 GUI ツールを作成することができます。GUIDE で作成される GUI プログラムはファンクション M-ファイルですので、これを `mcc` コマンドの対象とすることにより、MATLAB 上と同じ動作をする GUI アプリケーションを作成することができます。

本節では、GUIDE の使用方法および GUI のプログラミング方法の詳細は説明しません。その代わりに、GUIDE のデモ GUI を用いて、MATLAB Compiler による GUI アプリケーションの作成方法を説明します。GUIDE のデモ GUI の起動・作成方法は次の通りです。

- ① MATLAB コマンドウィンドウに下記コマンドを入力して、GUIDE クイックスタートを起動します。

```
>> guide
```

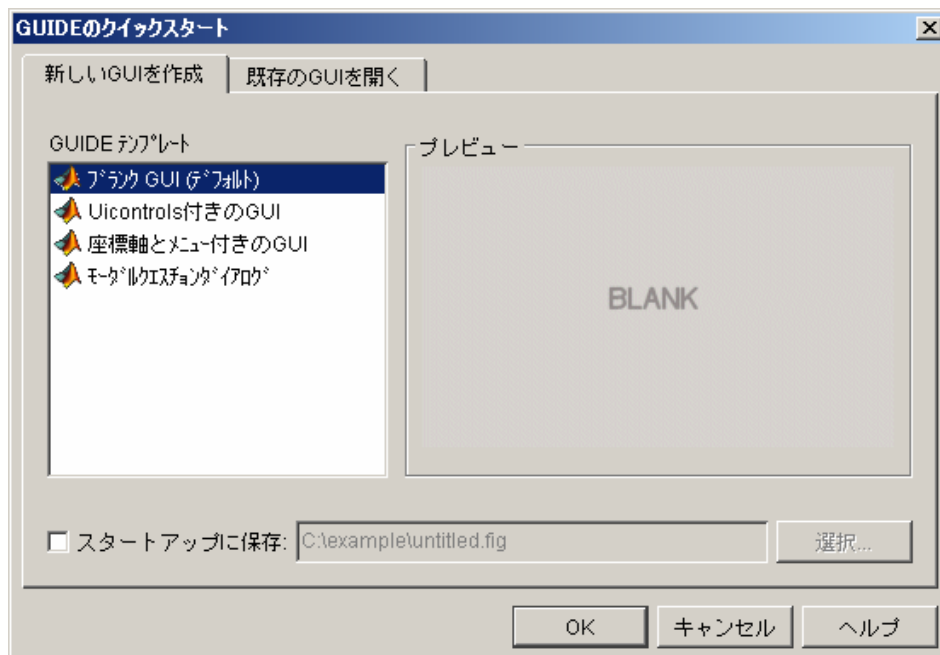


図 3-4 GUIDE クイックスタート

- ② [GUIDE テンプレート] から「座標軸とメニュー付きの GUI」を選択し、[スタートアップに保存] にチェックを入れます。

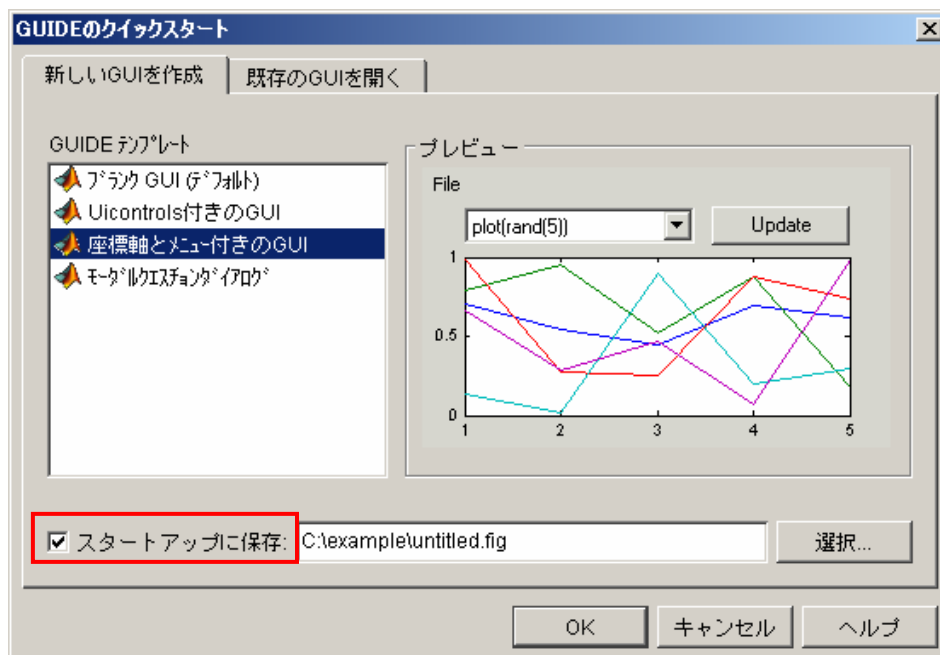


図 3-5 デモ GUI の選択

- ③ [選択] ボタンを押し、GUI の保存名を入力します。ここでは“mygui.fig”というファイル名で保存します。ファイル名の入力を終えたら、[保存] ボタンを押してください。

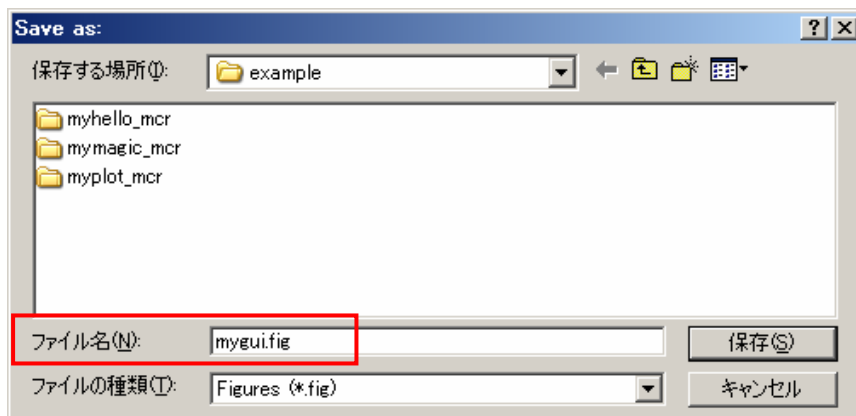


図 3-6 GUI 保存名の設定

- ④ [保存] ボタンを押して GUIDE クイックスタートに戻ったら、[OK] ボタンを押します。以上の操作で、カレントディレクトリに次のファイルが生成されます。

mygui.m : GUI の動作を記述するファンクション M-ファイル
 mygui.fig : GUI のレイアウトを保存した Figure ファイル

上記ファイルが生成されると、テキストエディタ／GUIDE にそれぞれのファイル（mygui.m／mygui.fig）が表示されます。

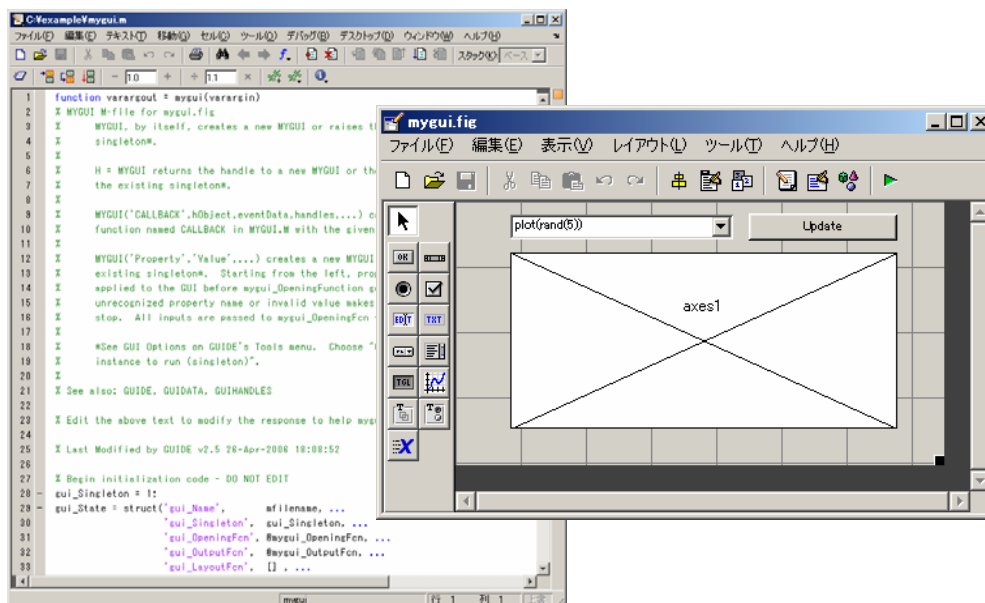


図 3-7 mygui.m と mygui.fig

- ⑤ 両ファイルを閉じてから、mygui の動作確認を行います。MATLAB GUI を起動するには、GUI 名（今の場合 mygui）をコマンドウィンドウに入力します。

```
>> mygui
```

mygui が起動したら、ポップアップメニューからプロットタイプを選択し、[Update] ボタンを押してください。プロットタイプによってグラフ表示が変化します。

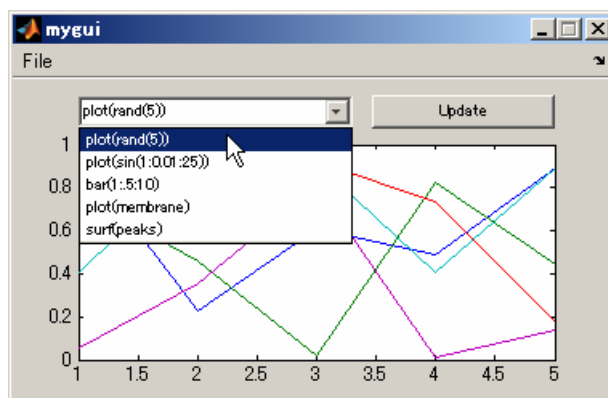


図 3-8 mygui の動作確認

mygui の動作確認が終了したら、次に mygui.m から実行ファイルを作成します。M-ファイルから実行ファイルを作成する方法は、例題 1 ～ 3 と同じ要領です。

```
>> mcc -m mygui.m
```

実行ファイルの作成に成功すると、カレントディレクトリに mygui.exe および mygui.ctf が生成されます。コマンドプロンプトを開いて、mygui.exe の動作確認を行ってください。MATLAB 上の mygui と同じ動作を示します。

3.7 コマンドプロンプトの非表示設定

MATLAB Compiler で作成した実行ファイルをアイコンのダブルクリックにより起動すると、最初にコマンドプロンプトが開きます。例えば、下図は mygui.exe アイコンをダブルクリックした際の画面をキャプチャしたものです。

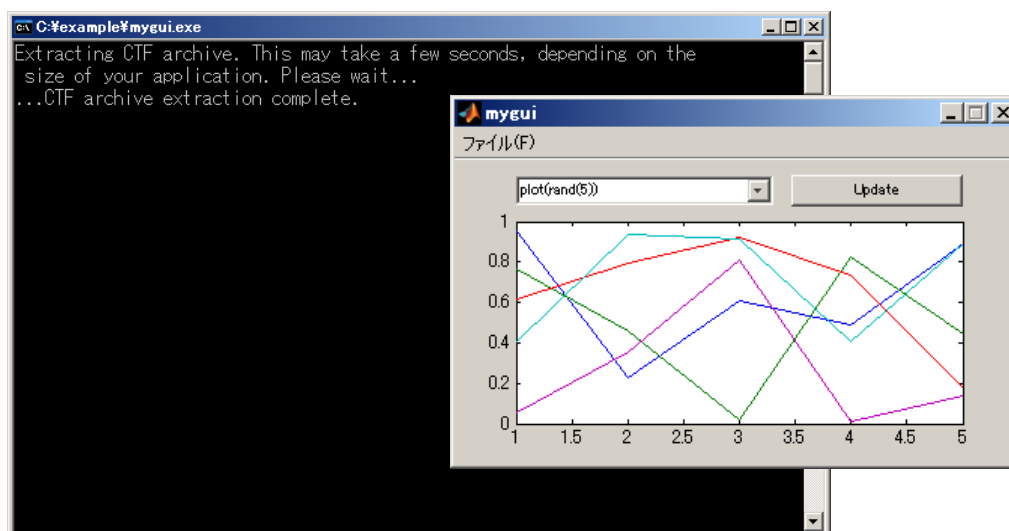


図 3-9 mygui.exe のダブルクリックによる起動

実行ファイルダブルクリック時にコマンドプロンプトを表示しないようにするためには、以下の手順でアプリケーションの作成を行います。

- ① MATLAB コマンドウィンドウに次のコマンドを入力して、MATLAB Compiler のコンパイラ設定ファイル (compopts.bat) をカレントディレクトリにコピーします。

```
>> copyfile(fullfile(prefdir, 'compopts.bat'))
```

- ② コピーした `compopts.bat` を下記コマンドで開きます。

```
>> edit compopts.bat
```

- ③ `compopts.bat` 内で変数 `LINKFLAGS` を設定している行 (`set LINKFLAGS= ...`) があります。この行でいったん改行して、次の行に以下の記述を追加します。なお、記述すべき内容は使用するコンパイラにより異なります。

■ Lcc C

```
set LINKFLAGS=%LINKFLAGS% -subsystem windows
```

■ Microsoft Visual C/C++

```
set LINKFLAGS=%LINKFLAGS% /SUBSYSTEM:WINDOWS /ENTRY:mainCRTStartup
```

■ Borland

```
set LINKFLAGS=%LINKFLAGS% -aa
```

- ④ M-ファイルからアプリケーションを作成します。

```
>> mcc -m mygui.m
```

- ⑤ カレントディレクトリにある `compopts.bat` を削除します。

```
>> delete compopts.bat
```

カレントディレクトリに `compopts.bat` が存在する場合、**MATLAB Compiler** はそれを参照してアプリケーション作成を行います。上記手順でコマンドプロンプトに文字列や計算結果を表示するアプリケーションを作成した場合、予期せぬエラーが発生する可能性がありますので、必要に応じて `compopts.bat` の編集・削除を行ってください。

4 ライブラリ作成

4.1 ライブラリ作成概要

ファンクション M-ファイルで記述されたアルゴリズムをユーザ独自のメインプログラムから呼び出してスタンドアロン環境上で動作させたい場合は、M-ファイルをライブラリ化して、メインプログラムから同ライブラリを呼び出します。

C ライブラリを作成するコマンドは次の通りです。

```
>> mcc -B csharedlib:ライブラリ名 myfun1.m myfun2.m
```

C++ライブラリを作成するコマンドは次の通りです。

```
>> mcc -B cpplib:ライブラリ名 myfun1.m myfun2.m
```

上記コマンドでカレントディレクトリに“ライブラリ名.dll”、“ライブラリ名.lib”、“ライブラリ名.h”、“ライブラリ名.ctf” が生成されます。複数の M-ファイルをライブラリに登録したい場合は、半角スペース区切りで M-ファイル名を追加します。

4.2 ライブラリ動作のための設定

MATLAB Compiler で作成したライブラリを動作させるためには、ライブラリおよび CTF ファイルが下記フォルダのどちらかに存在する必要があります。

- ライブラリを呼び出すアプリケーションと同じフォルダ
- 環境変数 Path に登録されているフォルダ

4.3 ライブラリの呼び出し

MATLAB Compiler で作成したライブラリを呼び出すメインプログラム（C/C++言語）のスケルトンコードは次の通りです。なお、このコードではライブラリ名が“mylib”である場合を想定しています。

```

/* ① ライブラリ用ヘッダのインクルード */
#include "mylib.h"

/* ② アプリケーションの初期化処理 */
if(!mclInitializeApplication(NULL, 0)) {
    アプリケーション初期化失敗時処理
}

/* ③ ライブラリの初期化処理 */
if (!mylibInitialize()) {
    ライブラリ初期化失敗時処理
}

/* ④ ライブラリ関数の使用 */
ライブラリ入力用データの作成
ライブラリ関数の呼び出し

/* ⑤ Figure ウィンドウを表示する場合、mclWaitForFiguresToDie 関数を実行する */
mclWaitForFiguresToDie(NULL);

/* ⑥ ライブラリの終了処理 */
mylibTerminate();

/* ⑦ アプリケーションの終了処理 */
mclTerminateApplication();

```

- ① ヘッダファイル“ライブラリ名.h”をインクルードします。
- ② MATLAB Compiler 作成ライブラリを使用する前に、mclInitializeApplication 関数を必ず実行する必要があります。
- ③ ライブラリを初期化するために、<ライブラリ名>Initialize 関数を実行します（この例の場合、<ライブラリ名>= mylib）。
- ④ ライブラリ関数を呼び出します。ライブラリ入力用データを作成する際には、C 言語と MATLAB で配列データのインデックス優先順序が異なることに注意してください（C 言語：行優先、MATLAB：列優先）。

- ⑤ ライブラリ内関数が **Figure** ウィンドウを表示する場合、メインプログラム終了前に **mclWaitForFiguresToDie** 関数を実行する必要があります。
- ⑥ ライブラリを終了するために、<ライブラリ名>**Terminate** 関数を実行します。
- ⑦ 最後に **mclTerminateApplication** 関数を実行します。

②と⑦および③と⑥は対の処理となっています。

■ 補足

ライブラリの呼び出し処理を C++のクラスとして実装する場合は、コンストラクタに初期化処理、デストラクタに終了処理を記述してください。次のコードはライブラリ呼び出し用クラスのスケルトンコードです。

```
// コンストラクタ
CMyLib::CMyLib()
{
    if (!mclInitializeApplication(NULL, 0)) {
        アプリケーション初期化失敗時処理
    }
    if (!mylibInitialize()) {
        ライブラリ初期化失敗時処理
    }
}

// デストラクタ
CMyLib::~~CMyLib()
{
    mylibTerminate();
    mclTerminateApplication();
}

// ライブラリ関数の呼び出し
CMyLib::CallMyLib()
{
    ライブラリ関数の呼び出し
}
```

4.4 例題 1 C 共有ライブラリ

myadd.m は 2 変数の和を求めるファンクション M-ファイル、mybar3.m は 3 次元バープロットを表示するファンクション M-ファイルです。

myadd.m

```
1 function y = myadd(x1, x2)
2 % 2 変数の和を求めるファンクション M-file
3
4 y = x1 + x2;
```

mybar3.m

```
1 function mybar3(x)
2 % 3 次元バープロットを表示するファンクション M-file
3
4 bar3(x) % 3 次元バープロット
5 colormap summer % カラーマップ変更
```

MATLAB 上での両 M-ファイルの実行例および結果は次の通りです。

```
>> x1 = [1 2 3; 4 5 6; 7 8 9];
>> x2 = [0 1 2; 0 1 2; 0 1 2];
>> y = myadd(x1, x2);
>> mybar3(y)
```

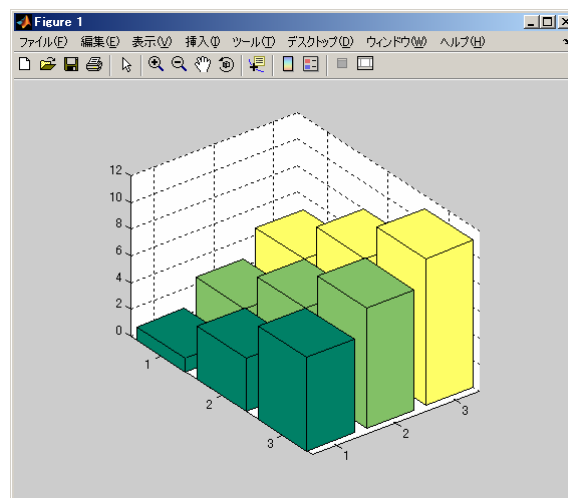


図 4-1 myadd および mybar3 の実行結果

myadd.m および mybar3.m を含む C 共有ライブラリ（ライブラリ名：mylib）を作成するために、次のコマンドを入力します。

```
>> mcc -B csharedlib:mylib myadd.m mybar3.m
```

ライブラリの作成に成功すると、カレントディレクトリに mylib.dll、mylib.lib、mylib.h、mylib.ctf が生成されます。ここで、ライブラリ関数がどのように実装されたかを確認するために、mylib.h を開きます。

```
mylib.h
-----
extern LIB_mylib_C_API
bool MW_CALL_CONV mlxMyadd(int nlhs, mxArray *plhs[],
                           int nrhs, mxArray *prhs[]);

extern LIB_mylib_C_API
bool MW_CALL_CONV mlxMybar3(int nlhs, mxArray *plhs[],
                           int nrhs, mxArray *prhs[]);

extern LIB_mylib_C_API bool MW_CALL_CONV mlfMyadd(int nargout, mxArray** y
                                                , mxArray* x1, mxArray* x2);

extern LIB_mylib_C_API bool MW_CALL_CONV mlfMybar3(mxArray* x);
```

mylib.h を見ると、myadd.m および mybar3.m が mlf または mlx という接頭語が付いた関数として実装されていることが分かります。mlf／mlx 関数は入出力形式が異なるだけで、機能的には同じ関数となります。両関数の入出力形式の特徴は次の通りです。

- mlf<M-ファイル名>(出力数, 出力引数ポインタリスト, 入力引数リスト)
ファンクション M-file に出力引数が無い場合、mlfMybar3 関数のように出力数と出力引数ポインタリストが省略された形になります。
- mlx<M-ファイル名>(出力数, 出力引数ポインタ配列, 入力数, 入力引数ポインタ配列)
この入出力形式は MEX-ファイル（MATLAB から実行できる C プログラム）と同じです。使用方法是 MEX-ファイルと同じですので、その詳細については「External Interfaces」マニュアルの MEX-ファイルの章を参照してください。

本チュートリアルでは `mlf` 関数の方を使用します。なお、`mlf/mlx` 関数の引数のデータタイプとして使われている `mxArray` は、MATLAB データを表現するためのデータタイプです。`mxArray` データの取り扱い方法については、「External Interfaces」マニュアルの MEX-ファイルの章を参照してください。

下記の `call_mylib.c` は、`mylib.dll` を呼び出すメインプログラムのサンプルです。

call_mylib.c

```
1  /*=====
2  * C 共有ライブラリを呼び出すサンプル C プログラム : call_mylib.c
3  *=====*/
4
5  #include <stdio.h>
6  #include "mylib.h"
7
8  /* 3×3 行列の転置を求める関数のプロトタイプ宣言 */
9  void transpose(double x[3][3], double xt[3][3]);
10
11 int main()
12 {
13     /* 入力用 mxArray データの宣言 */
14     mxArray *x1, *x2;
15
16     /* 出力用 mxArray データの宣言 */
17     mxArray *y = NULL;
18
19     /* 行列データ定義および転置用データの宣言 */
20     double data1[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
21     double data2[3][3] = {{0, 1, 2}, {0, 1, 2}, {0, 1, 2}};
22     double data1t[3][3];
23     double data2t[3][3];
24
25     /* アプリケーションの初期化処理 */
26     if(!mclInitializeApplication(NULL, 0)) {
27         fprintf(stderr, "アプリケーションを初期化できません¥n");
28         exit(1);
29     }
30
31     /* ライブラリの初期化処理 */
32     if (!mylibInitialize()) {
33         fprintf(stderr, "ライブラリを初期化できません¥n");
34         exit(1);
35     }
36
37     /* MATLAB データ順序は列優先のため、行列データを転置する */
38     transpose(data1, data1t);
39     transpose(data2, data2t);
40
41     /* mxArray データ領域を 3×3 実数行列として作成 */
42     x1 = mxCreateDoubleMatrix(3, 3, mxREAL);
43     x2 = mxCreateDoubleMatrix(3, 3, mxREAL);
44 }
```

```

45      /* mxArray 実部への行列データの割り当て */
46      memcpy(mxGetPr(x1), data1t, 9 * sizeof(double));
47      memcpy(mxGetPr(x2), data2t, 9 * sizeof(double));
48
49      /* ライブラリ関数の呼び出し */
50      mlfMyadd(1, &y, x1, x2);
51      mlfMybar3(y);
52
53      /* Figure ウィンドウを表示する場合、mclWaitForFiguresToDie 関数を実行する */
54      mclWaitForFiguresToDie(NULL);
55
56      /* ヒープ領域に作成された mxArray データを解放する */
57      mxDestroyArray(x1);
58      mxDestroyArray(x2);
59      mxDestroyArray(y);
60
61      /* ライブラリの終了処理 */
62      mylibTerminate();
63
64      /* アプリケーションの終了処理 */
65      mclTerminateApplication();
66
67      return 0;
68  }
69
70  /*=====
71   * 3×3 行列の転置を求める関数 : transpose
72   *=====*/
73  void transpose(double x[3][3], double xt[3][3])
74  {
75      int i, j;
76
77      for(i = 0; i < 3; i++) {
78          for(j = 0; j < 3; j++) {
79              xt[j][i] = x[i][j];
80          }
81      }
82  }

```

C 言語と MATLAB で配列データのインデックス優先順序が異なることを反映して、 3×3 の 2 次元行列を転置してから `mlfMyadd` 関数に入力している点にご注意ください。

`call_mylib.c` をビルドするには、`mbuild` コマンドを使用します。

```
>> mbuild call_mylib.c mylib.lib
```

`call_mylib.exe` を実行すると、図 4-1 と同じ結果が得られます。

4.5 例題 2 C++共有ライブラリ

本例題は「4.4 例題 1 C 共有ライブラリ」の C++バージョンです。Windows 版 MATLAB 7.2.0 (R2006a)に付属する C コンパイラ (Lcc C 2.4.1) では C++プログラムをビルドできないため、本例題を実行する前に使用コンパイラを C++対応コンパイラ (Microsoft Visual C/C++ 等) に変更してください。

まず、myadd.m および mybar3.m を含む C++共有ライブラリ (ライブラリ名 : mylib) を作成するために、次のコマンドを入力します。

```
>> mcc -B cpplib:mylib myadd.m mybar3.m
```

ライブラリの作成に成功すると、カレントディレクトリに mylib.dll、mylib.lib、mylib.h、mylib.ctf が生成されます。ここで、ライブラリ関数がどのように実装されたかを確認するために、mylib.h を開きます。

mylib.h

```
extern LIB_mylib_CPP_API void MW_CALL_CONV myadd(int nargout, mxArray& y
                                     , const mxArray& x1
                                     , const mxArray& x2);

extern LIB_mylib_CPP_API void MW_CALL_CONV mybar3(const mxArray& x);
```

C++共有ライブラリの場合、mlf 関数の代わりに上記のような<M-ファイル名>関数が実装されます。この関数の入出力形式は基本的には mlf 関数と同じですが、データタイプが mxArray から mxArray クラスに変更されている点が異なります。mxArray クラスは MATLAB データを mxArray よりも簡単に取り扱えるようにしたクラスです。mxArray クラスの詳細については、「MATLAB Compiler User's Guide」マニュアルの「C++ Utility Library Reference」を参照してください。

下記の call_mylib.cpp は、mylib.dll を呼び出すメインプログラムのサンプルです。

call_mylib.cpp

```
1  /*=====
2  * C++共有ライブラリを呼び出すサンプルC++プログラム：call_mylib.cpp
3  *=====*/
4
5  #include "mylib.h"
6
7  /* 3×3 行列の転置を求める関数のプロトタイプ宣言 */
8  void transpose(double x[3][3], double xt[3][3]);
9
10 int main()
11 {
12     /* アプリケーションの初期化処理 */
13     if (!mclInitializeApplication(NULL, 0)) {
14         std::cerr << "アプリケーションを初期化できません" << std::endl;
15         exit(1);
16     }
17
18     /* ライブラリの初期化処理 */
19     if (!mylibInitialize()) {
20         std::cerr << "ライブラリを初期化できません" << std::endl;
21         exit(1);
22     }
23
24     try {
25         /* 入力用 mxArray オブジェクト (3×3 実数行列) の作成 */
26         mxArray x1(3, 3, mxDOUBLE_CLASS, mxREAL);
27         mxArray x2(3, 3, mxDOUBLE_CLASS, mxREAL);
28
29         /* 出力用 mxArray オブジェクトの作成 */
30         mxArray y;
31
32         /* 行列データ定義および転置用データの宣言 */
33         double data1[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
34         double data2[3][3] = {{0, 1, 2}, {0, 1, 2}, {0, 1, 2}};
35         double data1t[3][3];
36         double data2t[3][3];
37
38         /* MATLAB データ順序は列優先のため、行列データを転置する */
39         transpose(data1, data1t);
40         transpose(data2, data2t);
41
42         /* 入力用 mxArray オブジェクトへの行列データの割り当て */
43         x1.SetData(data1t[0], 9);
44         x2.SetData(data2t[0], 9);
45
46         /* ライブラリ関数の呼び出し */
47         myadd(1, y, x1, x2);
48         mybar3(y);
49
50         /* Figure ウィンドウを表示する場合、mclWaitForFiguresToDie 関数を実行す
           る */
51         mclWaitForFiguresToDie(NULL);
52     }
```

```

53     catch (const mxArrayException& e) {
54         std::cerr << e.what() << std::endl;
55     }
56     catch (...) {
57         std::cerr << "予期せぬエラー" << std::endl;
58     }
59
60     /* ライブラリの終了処理 */
61     mylibTerminate();
62
63     /* アプリケーションの終了処理 */
64     mclTerminateApplication();
65
66     return 0;
67 }
68
69 /*=====
70  * 3×3 行列の転置を求める関数 : transpose
71  *=====*/
72 void transpose(double x[3][3], double xt[3][3])
73 {
74     int i, j;
75
76     for (i = 0; i < 3; i++) {
77         for (j = 0; j < 3; j++) {
78             xt[j][i] = x[i][j];
79         }
80     }
81 }

```

call_mylib.cpp では、mxArray メソッドおよびライブラリ関数呼び出し時のエラーをキャッチするために、mwException クラスを使用しています。mwException クラスの詳細については、「MATLAB Compiler User's Guide」マニュアルの「C++ Utility Library Reference」を参照してください。

call_mylib.cpp をビルドするには、mbuild コマンドを使用します。

```
>> mbuild call_mylib.cpp mylib.lib
```

call_mylib.exe を実行すると、図 4-1 と同じ結果が得られます。

4.6 Visual C/C++からの共有ライブラリ利用

先の例題では `mbuild` コマンドを用いてメインプログラムのビルドを行いました。デバッグ作業等を考慮すると、コンパイラの統合開発環境 (IDE) を使用した方が効率的な開発を行う事ができます。本節では、IDE 利用の例として、Microsoft Visual Studio .NET 2003 を用いて「4.5 例題 2 C++共有ライブラリ」の `call_mylib.cpp` をコンソールアプリケーションとしてビルドする手順について説明します。なお、`mylib.dll` は既に作成済みであるとします。

- ① Visual Studio .NET 2003 を起動し、[ファイル] メニュー→ [新規作成] → [プロジェクト] を選択します。
- ② [プロジェクトの種類] ツリーから [Visual C++プロジェクト] → [Win32] を選び、[テンプレート] から「Win32 コンソールプロジェクト」を選択します。[プロジェクト名] は `call_mylib` とします。

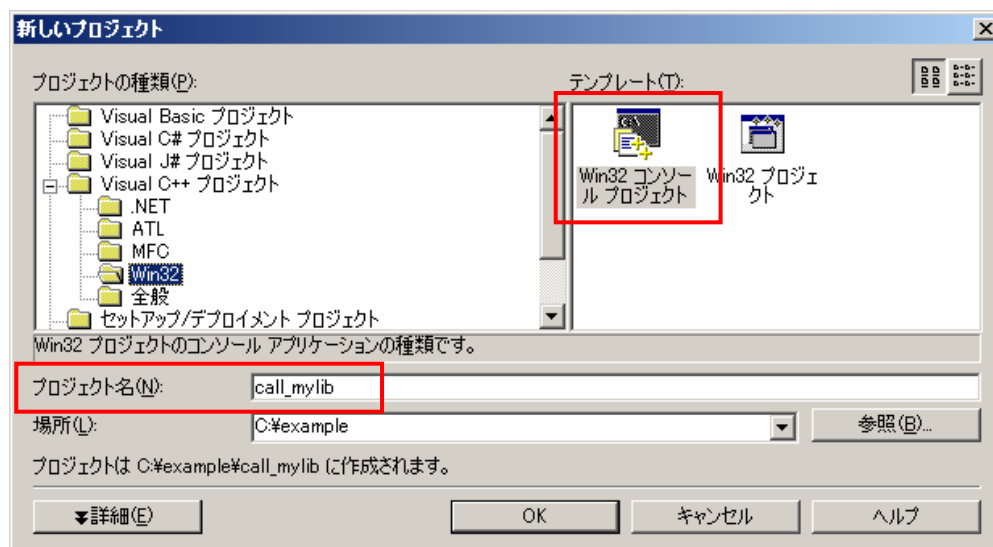


図 4-2 プロジェクト設定(1)

- ③ 「Win32 アプリケーションウィザード」の [アプリケーションの設定] で [空のプロジェクト] にチェックを入れてから、[完了] ボタンを押します。

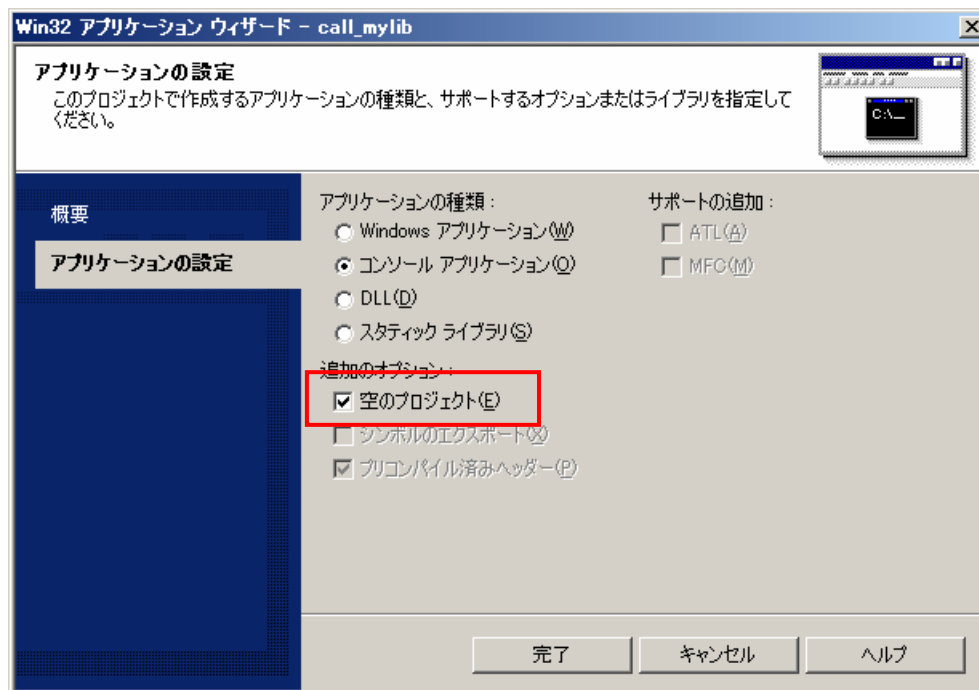


図 4-3 プロジェクト設定(2)

- ④ [プロジェクト] メニュー→ [既存項目の追加] で `call_mylib.cpp` を選択して [開く] ボタンを押します。
- ⑤ [プロジェクト] メニュー→ [プロパティ] → [C/C++] → [全般] → [追加のインクルードディレクトリ] を選択し、「...」ボタンをクリックします。「追加のインクルードディレクトリ」ウィンドウが表示されたら、`mylib.h` が存在するフォルダパスと `<matlabroot>%extern%include` を追加します。

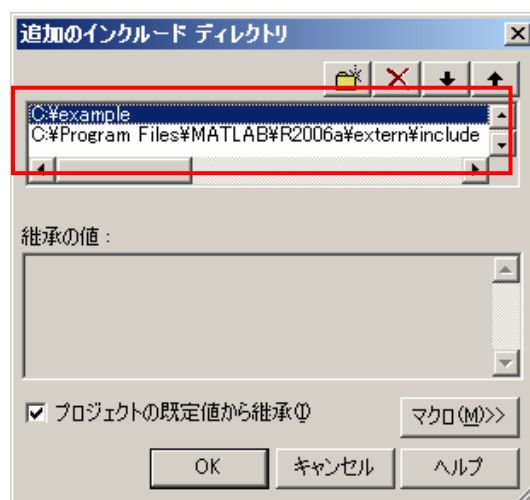


図 4-4 プロジェクト設定(3)

- ⑥ [プロジェクト] メニュー→ [プロパティ] → [C/C++] → [コード生成] → [ランタイムライブラリ] を「マルチスレッド DLL (/MD)」に設定します。

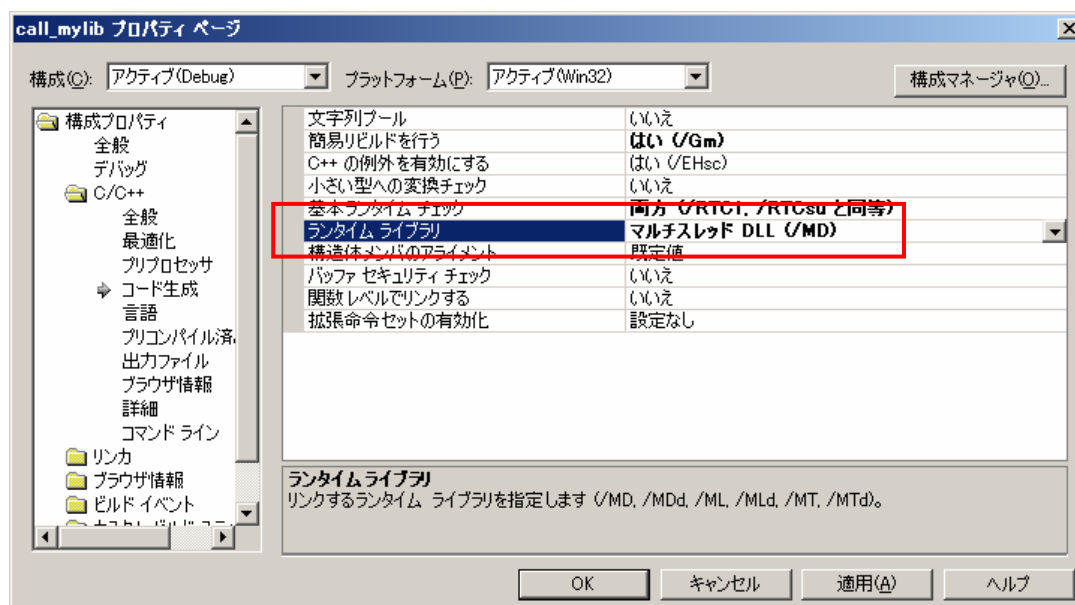


図 4-5 プロジェクト設定(4)

- ⑦ [プロジェクト] メニュー→ [プロパティ] → [リンカ] → [全般] → [追加のライブラリディレクトリ] を選択し、「...」ボタンをクリックします。「追加のライブラリディレクトリ」ウィンドウが表示されたら、mylib.lib が存在するフォルダパスと mclmcrtr.lib が存在するフォルダパス (<matlabroot>%extern%lib%win32%microsoft) を追加します。

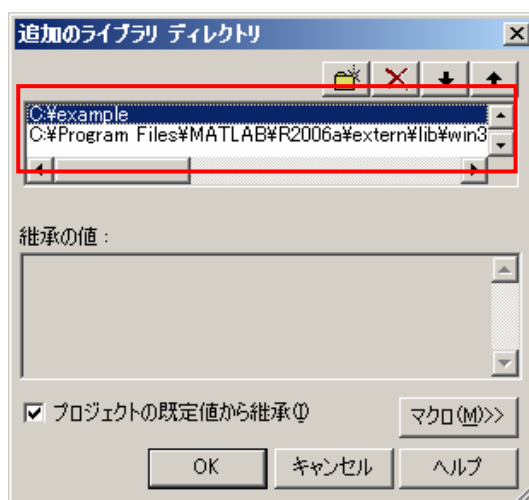


図 4-6 プロジェクト設定(5)

- ⑧ [プロジェクト] メニュー→ [プロパティ] → [リンカ] → [入力] → [追加の依存ファイル] を選択し、「...」 ボタンをクリックします。「追加の依存ファイル」ウィンドウが表示されたら、mylib.lib と mclmcrtr.lib を追加します。

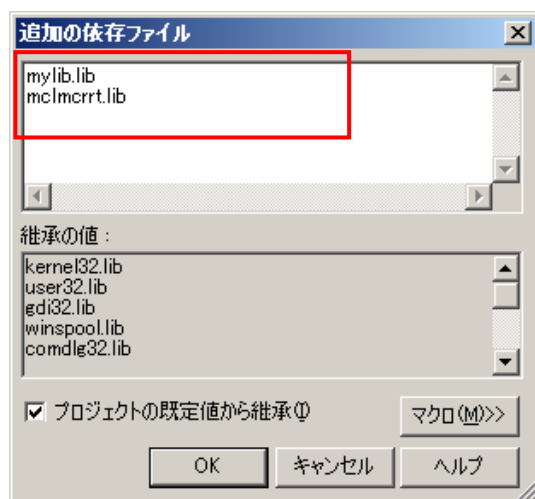


図 4-7 プロジェクト設定(6)

- ⑨ [ビルド] メニュー→ [ソリューションのビルド] を押し、call_mylib.exe を作成します。
- ⑩ call_mylib.exe が存在するフォルダに mylib.dll と mylib.ctf をコピー後、call_mylib.exe をダブルクリックして動作確認を行います。なお、Visual Studio. NET 2003 の [デバッグ] メニュー→ [開始] でエラーが出る場合は、mylib.ctf が存在するフォルダパスを環境変数 Path に追加してから動作確認を行ってください。

5 アプリケーションの配布

5.1 配布可能な環境

MATLAB Compiler で作成したスタンドアロンアプリケーションの動作可能環境は、アプリケーション作成に用いた MATLAB の動作対応環境と同じとなります。例えば、Windows 版 MATLAB 7.2 (R2006a) の場合、Windows XP / Windows 2000 / Windows Server 2003 が対応 OS となっていますので、Windows XP 上で作成したアプリを上記 OS 上で動作させることができます。なお、MATLAB の動作環境の詳細については、弊社ホームページの製品情報を参照してください。

5.2 配布ファイル

MATLAB Compiler で作成したスタンドアロンアプリケーションの動作に必要なファイルは以下の通りです。

- 実行ファイル（アプリ名.exe）またはライブラリ（アプリ名.dll）
- CTF ファイル（アプリ名.ctf）
- FIG ファイル（GUI を作成した場合）
- ライブラリを呼び出すメインプログラム（ライブラリを作成した場合）
- MCRInstaller.exe（<matlabroot>\¥toolbox¥compiler¥deploy¥win32 に用意されています）

上記ファイルを配布環境にコピーしてください。

5.3 配布環境での作業

MATLAB Compiler で作成したスタンドアロンアプリケーションの動作に必要な作業は以下の通りです。

- Administrator 権限を持つユーザで Windows にログインして、MCRInstaller.exe を実行します。これにより、スタンドアロンアプリケーションの動作に必要な MATLAB Component Runtime がインストールされます。
- 「3.2 実行ファイル動作のための設定」および「4.2 ライブラリ動作のための設定」に記載されている設定を行います。

付録 MATLAB ヘルプ・情報リソース


■ 関数・コマンドの使用方法を調べる

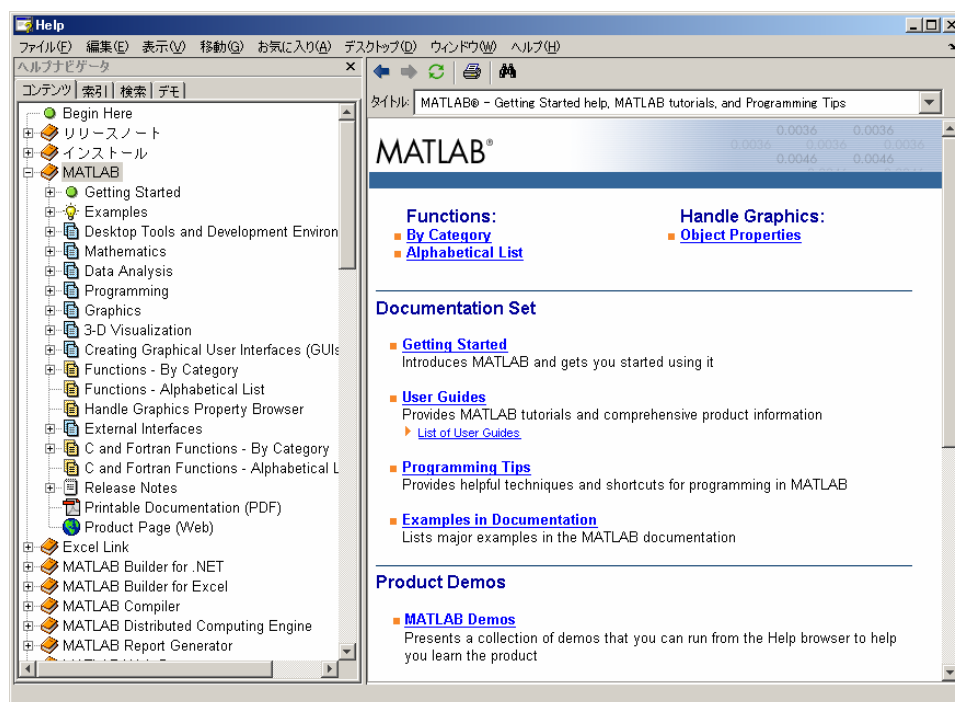
```
>> help 関数・コマンド名  
  
or  
  
>> doc 関数・コマンド名
```

■ 関数・コマンドの一覧リストを表示する

```
>> helpwin
```

■ MATLAB ヘルプブラウザ

MATLAB の [ヘルプ] メニュー→ [MATLAB ヘルプ] を選択するか、 アイコンをクリックすることによりヘルプブラウザを開くことができます。ヘルプブラウザからは MATLAB マニュアルやデモを参照することができます。



■ 開発元ソリューション検索ページ

http://www.mathworks.com/search/advanced_search.html

MATLAB に関する最新の情報を検索することができます。

The screenshot shows the MathWorks Advanced Search page. The browser window title is "The MathWorks - Search - Microsoft Internet Explorer". The address bar shows "http://www.mathworks.com/search/advanced_search.html". The page header includes the MathWorks logo and navigation links: Home, Select Country, Contact Us, Store, and a search bar. Below the header is a navigation menu with links: Products & Services, Industries, Academia, Support, User Community, and Company. The main content area is titled "Advanced Search" and contains a "Find results" section with four search criteria: "using all of the words", "using the exact phrase", "with at least one of the words", and "without the words". Each criterion has a text input field and a "Results Per Page" dropdown set to "10". A "search tips" link is also present. Below the search criteria is a section titled "To narrow your search, select one or more categories." with a subtext "By default, the entire Web site is searched." This section contains a grid of checkboxes for various categories: Products & Services (Applications, Training, Consulting, Third-Party Products), Industries, Academia, Entire Support Site (Solutions and Technical Notes, R2006a Documentation, Bug Reports, MATLAB & Simulink Based Books, Inactive Solutions, R13SP2 Documentation, Function List for all Products), User Community (File Exchange, MATLAB Newsgroup, Link Exchange, MATLAB Contest), and Company (Careers, Events, Newsletters, Pressroom). A "Search" button is located at the bottom of the category selection area. The footer includes copyright information "© 1994-2006 The MathWorks, Inc." and links to Site Help, Patents, Trademarks, Privacy Policy, and Preventing Piracy.