第7章 MATLAB のプログラミング II

7.1 実験目的

前章では Command Window から直ちに実行できる Matlab コマンドや複数のコマンドを記述した m-ファイルの作成を述べた。本章では、Script M-ファイルだけでなく、Function M-ファイルの作成方法や、言語構成、またはプログラミングの動作原理などの更なる基本的事項について習得することを目的としている。

7.2 実験内容

M-ファイル種類は①Script M-ファイル、②Function M-ファイルの 2 種類の形式に分けられる。これらのファイルの具体的な作成方法は後述するが、まず両者の特徴を次表にまとめる。

	Script M-ファイル	Function M-ファイル
記述と実行の特徴	実行したいコマンド・関数を順に記述するのみ。	function で始まる宣言文が必要である。
	入力引数および出力引数がない。	入力引数のみ、または入出力両方の引数の定義が必要である。
	一連のコマンド・関数をまとめて処理することができる。	定義したい関数を作成し後に実行することができる。
	コメント： 実際のプログラミングでは、一般的にスクリプトM-ファイルでプログラムを最初に作成し、アルゴリズムの検証等を済ませたらそのプログラムをファンクションM-ファイルに移行させることが多い。	
Workspace の使用	ファイル内の変数は、全てグローバル変数として Workspace に一括して管理され、同一の変数名が使用された場合にはその内容も変化してしまう。	ファイル内の変数は、ローカル変数として Workspace に管理され、各独立した関数中で同一の変数名が使用されている場合でも、それらの内容は互いに干渉されない。
有効性	何度も繰り返して試行させるような一連の処理の記述に向いている。	MATLAB 言語を拡張するのに極めて有効である。

7.2.1 スクリプト M-ファイルのプログラミング

スクリプト M-ファイル自体の作り方は前述したように非常に簡単である。つまり、実行したいコマンド､関数を順に並べて一つのスクリプト M-ファイルの中に記述することである。しかし実際に、どの関数をどのようにして使用したら良いのかというのは Matlab の関数を事前にある程度熟知しておく必要がある。本節では、まず、script M-ファイルによく使われているコマンド・関数の一覧を示し、特にプログラミング言語でよく使用される制御文について詳しく説明する。その後、応用例題を通して Matlab プログラムの動作原理について説明する。

1. M-ファイルの常用コマンド･関数

M-ファイルは一連のコマンド・関数で構成される。以下良く用いられるコマンド・関数を種類別に纏めたものを次表に示す。

コマンド・関数の種類	よく使用されるコマンド・関数の名とその説明
一般的なコマンド	clear - ワークスペース変数を削除する。 load - ファイルから保存された変数を読み込む。 save - ワークスペース変数をファイルに保存する。
数値演算子、 比較演算子、 論理演算子	数値演算子は**例題 4_7** を参照すること 比較演算子: eq - 等しい == ne - 等しくない ~= lt - より小さい < gt - より大きい > le - より小さいまたは等しい <= ge - より大きいまたは等しい >= 論理演算子: and - 要素ごとの論理積 & or - 要素ごとの論理和 | not - 否定 ~
制御コマンド	制御文は**例題 5_2** を参照すること
基本行列と行列操作	**第 5 章**を参照すること。
初等数学関数	三角関数： sin - 正弦値 sinh - 双曲線正弦値

	asin - 逆正弦値　asinh - 逆双曲線正弦値 指数関数： exp - 指数関数　log - 自然対数 log10 - 常用対数(底が 10)　sqrt - 行列の平方根 複素数： abs - 絶対値　angle - 位相角 imag - 複素数の虚部　real - 複素数の実部 丸めと剰余： round - 最も近い整数への丸め rem - 除算の剰余　sign - 符号関数
特殊数学関数	gamma - ガンマ関数
行列関数 (線形数値代数)	行列解析： norm - 行列やベクトルのノルム　rank - 行列のランク det - 行列式　trace - 対角要素の和 線形方程式： inv - 逆行列　lu - LU 分解　qr - 直交三角分解 固有値と特異値： eig - 固有値と固有ベクトル　svd - 特異値分解
データ解析と フーリエ変換に 用いられる関数	基本演算： max - 配列の最大要素　min - 配列の最小要素 mean - 配列の平均値　median - 配列の中央値 std - 標準偏差　var - 分散 sort - 要素の昇順に並べ替え hist - ヒストグラム sum - 配列要素の和　prod - 配列要素の積 相関： corrcoef - 相関係数　cov - 共分散行列 フィルタリングとコンボリューション（畳み込み演算）： filter - 1 次元デジタルフィルタリング conv - コンボリューションと多項式の乗算 deconv - デコンボリューションと多項式の除算 detrend - 線形トレンド除去 フーリエ変換： fft - 1 次元高速フーリエ変換 ifft - 1 次元逆高速フーリエ変換

オーディオサポート関数	オーディオハードウェアデバイス： sound - ベクトルを音声として再生 soundsc - 音声としてのオートスケーリングと再生 オーディオファイルのインポート/エクスポート： wavread - (".wav")サウンドファイルの読み込み wavwrite - (".wav")サウンドファイルの書き出し
補間と多項式	データ補間： interp1 - 1次元補間(table lookup) 多項式： roots - 多項式の根 polyval - 多項式の計算
関数を引数とする関数と ODE ソルバ	数値積分(求積法) ： quad - 低次の数値積分 quadl - 高次の数値積分 常微分方程式の解： ode45 - ノンスティッフな微分方程式の中次の解法 1次元偏微分方程式ソルバ例： pdepe - 双曲線-楕円 PDEs の初期境界値問題の解法
グラフィックス	**第 6 章**を参照すること。
キャラクタと文字列	文字列、数値の変換： num2str - 数値を文字列に変換 str2num - 文字行列を数値に変換
ファイルの入出力	ファイルのオープンとクローズ： fopen - オープン fclose - クローズ バイナリファイルの入出力例： fread - ファイルからバイナリデータを読み込む。 fwrite - バイナリデータをファイルに書き出す。
時間と日付	現在の日付と時間： now - 日付番号での現在の日付と時間 date - 日付文字列での現在の日付 時間関数： tic - ストップウオッチタイマの開始 toc - ストップウオッチタイマの停止 pause - 秒単位での待機

これらのコマンドや関数に関する詳細は、`>> helpwin` で参照してください。

2. 制御文

一般的にプログラミング作業においては、同じような計算パターンが重複しないように、反復や分岐制御文などを用いたりしてプログラミングの効率化を図る場合が多い。MATLABでも反復、分岐、条件と言った制御文が用意されている。また MATLAB の制御文の文法はC 言語等とほぼ同じであるので容易に理解できるであろう。しかし、MATLAB はインタープリタ言語であるため、普通コンパイラ言語と違い、プログラム中に反復文を使い過ぎてしまうと演算速度がかなり落ちてしまう。このような理由から、MATLAB では制御文の使用を極力控え、出来るだけ MATLAB に備えられている行列演算の特性をうまく利用してプログラミングを行うことが望ましい。

例題 7_1：典型的な制御文を次表のように示す。また、制御文ごとに作成したプログラム例題を実行してみよう。

制御文の書式	プログラムの例題： 次の制御文を用いて整数 1～100 までの奇数和と偶数和、総和をそれぞれ求めよ。

1．指定回数の反復文 ：
for … end
for … for …… end … end
 	
書式例：
for 繰返し回数
実行文
End
	%ファイル名：forloop.m
	clear all
	odd=0;even=0;
	for i=1:2:100
	odd=odd+i; even=even+(i+1);
	end
	total=odd+even;

2．不定回数の反復文:
while … end
while … while …… end … end
 	
書式例：
while 繰返し終了条件
実行文
End
	%ファイル名：whileloop.m
	clear all
	odd=0;even=0;i=0;j=0;I=100;
	while i<=I;
	odd=odd+j; j=i+1;
	even=even+i; i=i+2;
	end
	total=odd+even;

3．条件分岐文: if … end if … else … end	%ファイル名：forif.m clear all odd=0;even=0;

if…elseif…else…end
 	
書式例：
if 判別条件１
実行文１
elseif 判別条件２
実行文２
else 判別条件３
実行文３
End
	for i=1:100;
	if rem(i,2) ~=0
	odd=odd+i;
	else
	even=even+i;
	end
	end
	total=odd+even;

4. 分岐文:
switch … case …… end
switch … case … othercase … end
 	
書式例：
switch 分岐点（数値、数式）
case 分岐路１
実行文１
case 分岐路２
実行文２
othercase
実行文３
end
	%ファイル名：switchcase.m
	clear all
	odd=0;even=0;
	for i=1:100;
	switch rem(i,2)
	case 1
	odd=odd+i;
	case 0
	even=even+i;
	end
	end
	total=odd+even;

例題7_2：第6章にある例題6_5（p.6-8）の計算プログラムに対して、制御文を利用したものに書き直してみよう。

プログラム　Samp7_2.m

```
% The generalized Gaussian distributions
% filename: Samp7_2.m
% Jianting Cao, Feb. 11, 2003

% Workspace 変数をすべて削除
clear all

% 係数 lambda, x, alpha, p の定義
```

```
lambda = 1/sqrt(2); x = -5:0.1:5; alpha = [1 2 3];p=[];

% ループ計算
for i=1:3;
    p(:,i)=alpha(i)*lambda/(2*gamma(1/alpha(i)))*exp(-abs(lambda*x).^alpha(i));
end;

% プロットと修飾
plot(x,p(:,1),'k*-',x, p(:,2),'ro-',x, p(:,3),'b.-');
title('The generalized Gaussian distributions ','FontSize',14);
xlabel('x','FontSize',16);ylabel('p(x)','FontSize',16);
legend('super-Gaussian', 'Gaussian', 'sub-Gaussian');
axis([-5 5 0 0.5]);grid;
```

7.2.2 ファクション M-ファイルのプログラミング

前述したように、MATLAB ではファンクション M-ファイルを使うことで任意の関数の作成ができ、MATLAB 言語そのものさえも拡張することができる。具体的に Function M-ファイルの記述は以下のようになる。

1) 第一行目の Function 定義部の一般書式は

function [出力引数] = 関数名(入力引数)

であるが、出力の引数がない場合の形式は、

function 関数名(入力引数)

となる。この定義部を実行すると、入力引数が Local Workspace へ引き渡される。

2) 第二行目以降のコメント部は

定義した関数に関する情報(目的、入力と出力の引数の使用法など)を記述しておけば、後に help 関数名でその使用法をチェックすることができる。なお、コメント文は記号<%>で始まる。

3) Function 本体：

普通 Local Workspace で計算を行い、最後に関数の計算した値をあらかじめ出力引数に割り当ててから Workspace に戻り、後の操作に用いられるようにする。

例題7_3：平均値と分散を一度に計算する関数を作成してみよう。

プログラム名： my_mv.m

```
Function [mean,var]=my_mv(x)
%  ベクトルの平均値,分散を計算する関数
```

```
%  x  - 行，または列ベクトル
%  mean  -  平均値  （スカラ）
%  var -    分散     （スカラ）
% 埼工大__電子工学科__専門実験__第 5 章; Feb. 15, 2003; by 曹  建庭

[m,n] = size(x);
if (~((m==1) | (n==1)) | (m==1 & n==1))  % ベクトルの判別
   error('Input must be a vector!') % エラーメッセージの表示
end
if m>n                    % 列ベクトルである場合の計算
   mean = sum(x)/m;
   var = sum(x.^2)/m;
elseif m<=n               % 行ベクトル、またはスカラである場合の計算
   mean = sum(x)/n;
   var = sum(x.^2)/n;
end
```

上記で作成した関数をファイル my_mv.m に保存する。（一般的にファイルの名前は function の名前(my_mv)と同じものをつけることが多い。）

第 1 行目の Function 定義部の記述から判るように、my_mv 関数は 1 入力・2 出力の関数である。すなわち、1 個のベクトルを入力引数として与えれば、関数内で計算された平均値と分散値がそれぞれ出力引数から得られる。

上述したように、第 2 行目から始まるのコメント部は関数(my_mv)の目的・使用法などが記述されており、Command Window で help を実行すると次のように表示される。このようにコメントを残しておけば、その目的・使用法を後に忘れたとしても役に立つ。

```
>> help my_mv

   ベクトルの平均値,分散を計算する関数
   x  - 行，または列ベクトル
   mean  -  平均値  （スカラ）
   var -    分散     （スカラ）
  埼工大__電子工学科__専門実験__第 5 章; Feb. 15, 2003; by 曹  建庭
```

上記で実際に作成した関数 my_mv をコマンドラインから実行してみると、

```
>> a=[1 2];
>> [u v]=my_mv(a)
u =
    1.5000
v =
    2.5000
```

のようになり、関数 my_mv は正しく動作していることが判る。また、列ベクトル a=[1　2]' を入力すると、下記のような結果が得られる。しかし、行列 a=[1 2;1 2]; を入力して実行するとこの関数は正しく機能せず、次のエラーを返して動作が中断される。

```
>> a=[1 2;1 2];
[u v]=my_mv(a)
??? エラー: ==> my_mv
Input must be a vector!
```

また、この例題を実行することで以下のようなことが分かる：

1) Command Window で定義した変数名(a u v)は必ずしも第 1 行目の Function 定義部で定義された引数名(x, mean, var)と同じである必要がない。
2) 例題 5_4 を実行した後に Workspace で使われている変数を確認すると、変数(a u v)のみが表示されるが、ファンクション M-ファイル my_mv.m に記述されている、関数 my_mv で使用されたローカル変数(x, m, n, mean, var)であるので表示されない。

例題7_4：例題7_2のプログラムを関数の形で作成してみよう。

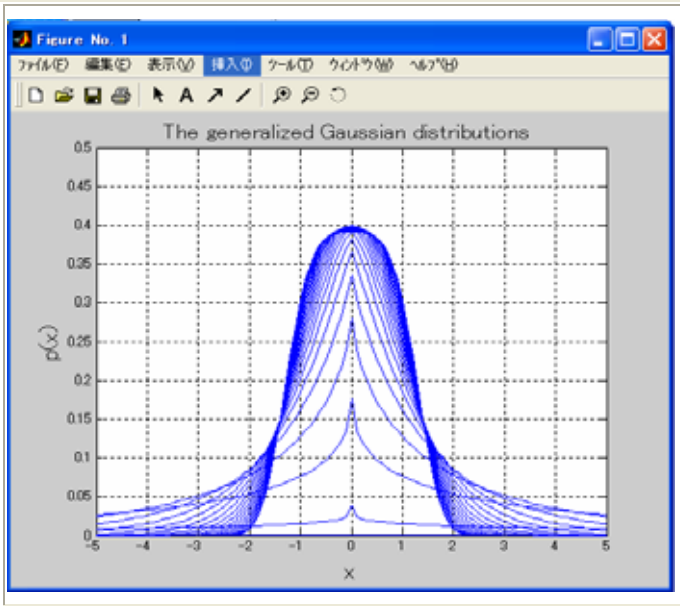
プログラム名：　my_pdf.m

```
function p=my_pdf(x,alpha)
%  ベクトルの平均値,分散を計算する関数
%
%  x  - 確率変数のベクトル, 例えば, x=-5:0.1:5
%  alpha  -  分布を制御する変数ベクトルまたはスカラで与えられる正の値である。
%  p -   確率密度関数の値、サイズは x と同じベクトル
%
% 埼工大__電子工学科__専門実験__第 5 章; Feb. 15, 2003; by 曹　建庭
```

```
% 入力引数の個数の判別
if nargin <= 1, error('Not enough input arguments.');end

% x, alpha 各変数のサイズの判断
[m n] = size(x);[u v]=size(alpha);
if (~((m == 1) | (n == 1)) | (m == 1 & n == 1))      % ベクトルの判別
    error('Input must be a vector!')                  % エラーメッセージの表示
elseif (u>1 & v>1) | (u==0 | v==0)                    % ベクトルとスカラの判別
    error('Input must be a vector or a scale !')      % エラーメッセージの表示
end

% alpha が正の値であることを判別
if any(le(alpha,zeros(u,v)))==1       % alpha のすべての要素  <= zeros(u,v)
   error('Input must be a positive value!')  % エラーメッセージの表示
end

% 変数 lambda, p の定義
lambda = 1/sqrt(2);p=[];

% loop の計算とプロット
for i=1:max([u v])
    p(i,:)=alpha(i)*lambda/(2*gamma(1/alpha(i)))*exp(-abs(lambda*x).^alpha(i));
    plot(x,p(i,:));hold on;
end
% プロットの修飾
title('The generalized Gaussian distributions ','FontSize',14);
ylabel('p(x)','FontSize',16);xlabel('x','FontSize',16);
axis([min(x) max(x) 0 0.5]);grid;
hold off
```

実行例

```
>> x=-5:0.1:5;

>> alpha=0.1:0.2:4;

>> p=my_pdf(x,alpha);
```



7.3 実験課題

1. 反復文（for … end）及び関数 sum をそれぞれ利用し、次の課題に対して Script M-ファイル形式のプログラム、計算結果と演算時間を示せ。なお、プログラムの演算時間を計るために、ストップウオッチタイマ関数（tic, toc）を利用せよ。
   (1) 整数 1～10000 までの和。
   (2) 整数 1～10000 までの自乗の和。

2. ある数値列の平均値及び分散をそれぞれ計算するFunction M-ファイル形式のプログラムを作成せよ。また、正規分布の乱数関数randn(1,n)で生成したデータに対して作成したプログラムを利用して①n=10、②n=200、③n=10000の場合、乱数の平均値及び分散の値を考察せよ。

3. 一元二次方程式 $ax^2 + bx + c = 0$ の解を求めるFunction M-ファイル形式のプログラムを作成せよ。また、種々の適当な係数を与えて作成した関数プログラムを検証せよ。