Chapter 14 Function M-files

14.1 M-File Construction Rules

There is another type of m-File besides a MATLAB script file. It is a MATLAB function m-File. Typically, a MATLAB function file contains a sequence of MATLAB commands which process specific information passed to it in the form of input variables and generate output based on the information. A function m-File behaves like a black box from the user's standpoint because all that is evident are the variables passed to it from the MATLAB Workspace and those which are returned from it to the Workspace. Intermediate variables created in the function m-File are not stored in the MATLAB Workspace and hence will not overwrite any variables of the same name which happen to be present in the Workspace.

Function m-Files are created with the MATLAB Editor/Debugger and saved as text files with a .m extension similar to script files. All MATLAB built-in mathematical functions are function m-Files.

A simple function m-File called `floan.m` is shown in Example 14.1.1. Its main purpose is to calculate a mathematical function value based on values passed to it from a script file called `Loan_Interest.m` and return the computed value to the calling program. The calling script file `Loan_Interest.m` is included as well in order to see where the function calls occur.

Example 14.1.1

```
function i_R = floan(AA,PP,NN,ii)
% floan is a function called by the script file 'Loan_Interest.m'
% It accepts a monthly payment AA, loan amount PP, a loan duration NN
% (in years) and an interest rate ii. It returns a single value
% determined from evaluation of the function. The script file
% 'Loan_Interest.m' is a root solving program which attempts to find
% the root iR of f(A,P,N,i)=0.  The user selects from one of two root
% solvers, the Bisection Method or the Method of False Position.  The
% root iR found in 'Loan_Interest.m' is the annualized interest (in
% percent).

N_months=12*NN;  % Convert from years to months
   if ii~=0  % Check if ii not equal 0
     x=1+ii;
     i_R=AA/PP-(ii*x^N_months)/(x^N_months-1); % Calculate function
                                               % value when ii not equal 0
   else  %  ii equals 0
     i_R=AA/PP - 1/N_months % Calculate function value when ii equals 0
   end % if ii~=0  % Check if ii not equal 0
```

```
% Loan_Interest.m
% A script file that calculates the iterest rate on a loan
% It calls function floan.m with inputs P,A,N and i
% P:Loan amount
% A:Monthly installment
% N:Duration of loan expressed in years
% i:An interest rate (decimal value)
% iR: Annual per cent interest rate on loan

%------------------------------------------
clear,clc
echo off
P=35000;  % Loan amount
A=750;  % Monthly payment
N=5; % Loan duration (in years)
method='F';
iLinitial=0;
iUinitial=1;
eSTOP=0.001;
iL=iLinitial;
iU=iUinitial;
fiL=floan(A,P,N,iL);
fiU=floan(A,P,N,iU);
count=0;
i=0;
eA=eSTOP+1;
while (count<100)&(eA>eSTOP)
   iold=i;

%%%%% Calculate next i and f(i) %%%%%
   if method=='B'
         i=(iL+iU)/2;
   else
         i=iU-(fiU*(iL-iU)/(fiL-fiU));
   end % if
   fi=floan(A,P,N,i);
   inew=i;

%%%%% Determine new endpoint and f(i) at new endpoint %%%%%
    if fiL*fi<0
      iU=i;
      fiU=floan(A,P,N,iU);
   else
      iL=i;
      fiL=floan(A,P,N,iL);
   end % if

%%%%% Caclulate approximate relative error and increment counter %%%%%
   if count>0
   eA=abs((inew-iold)/inew);
   end % if
   count=count+1;

end % while

iR=1200*i; % Calculate annual per cent interest rate
fprintf('\n\n')
fprintf('Loan Amount = $%8.0f \nMonthly Payment = $%7.1f \nLoan Period =
%4.0f years \n',P,A,N)
fprintf('Annual Interest Rate = %6.2f per cent \n',iR)
```

```
Enter Loan Amount: 40000
Enter Monthly Installment: 900
Enter Loan Duration in Years: 5
Enter "B" for Bisection Method or "F" for False Position: F

Loan Amount = $   40000
Monthly Payment = $  900.0
Loan Period =    5 years
Annual Interest Rate =  12.50 per cent

Another Problem (Y) or (N)?   N
```

Several important characteristics of function m-Files are:

1. The first line must contain the word `function` followed by its calling syntax (see function `floan.m`). The input and output variables on the function declaration line are local to the function, meaning they are not accessible to the calling file, which may be a script file or possibly another function file. They are merely temporary variables with values inside the function's workspace. In the previous example, input variables `AA`, `PP`, `NN`, and `ii` assume values passed to the function from the calling script file and the value of output variable `i_R` computed inside the function is passed back to the calling script file. Input variables can be modified inside the function, i.e. appear on the left side of an assignment statement, but the new values cannot be returned to the calling file.
2. Function names have similar constraints to MATLAB variable names.
3. The file named used to save the function m-File should be identical to the name appearing in the function declaration line although its not required. MATLAB references the file name stored on disk when executing a function.
4. A function m-File terminates after executing the last statement or whenever a `return` command is encountered.
5. Function m-Files can call script files and other functions which are referred to as subfunctions. The subfunctions are constructed in the same was as a function m-File and are appended to the end of the primary function in any order.

The next example shows a sequence of commands entered in the Command window. A function m-file '`f1.m`' is called with arguments `A`, `B` and `i` which have been assigned numerical values. The function '`f1.m`' , shown below contains a reference to a subfunction m-File '`noise.m`'. The '`noise.m`' subfunction declaration line appears at the end of the function '`f1.m`' listing.

Example 14.1.2

```
A=1;B=2;
u=0;
for i=1:3
   u=u+f1(A,B,i)
end % i

z =    0.9681
y =    3.9681
u =    3.9681

z =    0.6199
y =    5.6199
u =    9.5880

z =    0.6408
y =    7.6408
u =   17.2287

function y=f1(A,B,x)
% Linear function with noise component
y=A+B*x+noise(A)
function z=noise(C)
%  Noise function
z=C*rand
```

The subfunction '`noise.m`' is invoked by the function '`f1.m`' which passes it the value of variable `A`, namely 2, to be substituted for variable `C` wherever it appears in '`noise.m`'. Hence, the first output variable calculated is the subfunction variable `z` whose value is passed back to the calling function '`f1.m`' as the noise component in the calculation of `y`. Finally, the numerical value of `y` is returned to the Command window program to update variable `u`.

The comment lines following the function declaration deserve careful consideration. They will be displayed whenever a `help 'function name'` command is issued, usually from the Command window. For example, to obtain a description of functions '`floan`' and '`f1`', see Example 14.1.3.

Example 14.1.3

```
help floan  % Get description of function floan

  floan is a function called by the script file 'Loan_Interest.m'
  It accepts a loan amount P, a monthly payment A, a loan duration N
  (in years) and an interest rate i. It returns a single value
  determined from evaluation of the function. The script file
  'Loan_Interest.m' is a root solving program which attempts to find
  the root iR of f(A,P,N,i)=0.  The user selects from one of two root
  solvers, the Bisection Method or the Method of False Position.  The
  root iR found in 'Loan_Interest.m' is the annualized interest (in
  percent).
```

14.2 Input and Output Arguments

There are a number of rules governing the use of input and output arguments in MATLAB function calls. Several important features are listed below.

1. A function m-File can have zero input and zero output arguments
2. A function can be called with fewer input and output arguments than appear in the function declaration line.
3. The number of input and output arguments passed to the m-File function can be determined inside the function by using the MATLAB functions '`nargin`' and '`nargout`' respectively.
4. A function with outputs in the function declaration line is not required to return values for all of its outputs. Any output not assigned a value in the body of the m-File function will not return a value back to where its called from.

Example 14.2.1

```
function [x0,y0]=origin_close(x1,y1,x2,y2,x3,y3,x4,y4)
% This functions accepts up to 4 input points (xi,yi) and finds the
% one closest to the origin.  The coordinates of the closest point to
% the origin are returned in the outputs x0 and y0
switch nargin
case 0
  disp(' No points searched')
case 2
  x0=x1;y0=y1;
case 4
  d1=sqrt(x1*x1 + y1*y1);
  d2=sqrt(x2*x2 + y2*y2);
    if d1 < d2
      x0=x1;y0=y1;
    else
      x0=x2;y0=y2;
    end % if d1<d2
case 6
  d1=sqrt(x1*x1 + y1*y1);
  d2=sqrt(x2*x2 + y2*y2);
  d3=sqrt(x3*x3 + y3*y3);
    if d1==min([d1,d2,d3])
      x0=x1;y0=y1;
    elseif d2==min([d1,d2,d3])
      x0=x2;y0=y2;
    else
      x0=x3;y0=y3;
    end % if d1==min([d1,d2,d3])
 case 8
   d1=sqrt(x1*x1 + y1*y1);
   d2=sqrt(x2*x2 + y2*y2);
   d3=sqrt(x3*x3 + y3*y3);
   d4=sqrt(x4*x4 + y4*y4);
     if d1==min([d1,d2,d3,d4])
```

```
        x0=x1;y0=y1;
      elseif d2==min([d1,d2,d3,d4])
        x0=x2;y0=y2;
      elseif d3==min([d1,d2,d3,d4])
        x0=x3;y0=y3;
      else
        x0=x4;y0=y4;
      end % if d1==min([d1,d2,d3,d4])
end % switch nargin
```

The following MATLAB commands create a set of 4 data points and then calls the function '`origin_close.m`' to find the closest point to the origin.

```
x(1:4)=rand(1,4),y(1:4)=rand(1,4)
x1=x(1);y1=y(1);
x2=x(2);y2=y(2);
x3=x(3);y3=y(3);
x4=x(4);y4=y(4);
[xnear,ynear]=origin_close(x1,y1,x2,y2,x3,y3,x4,y4)

x =    0.0136    0.7895    0.8996    0.0022
y =    0.5163    0.6244    0.2652    0.3990

xnear =    0.0022
ynear =    0.3990
```

The function in Example 14.2.2 has no outputs. It results in the drawing of a green rectangle of specified size and location determined by the numerical values passed to its inputs. The inputs fix the lower and upper corner vertices subject to a set of constraints. A script file that calls the function several times is included.

Example 14.2.2

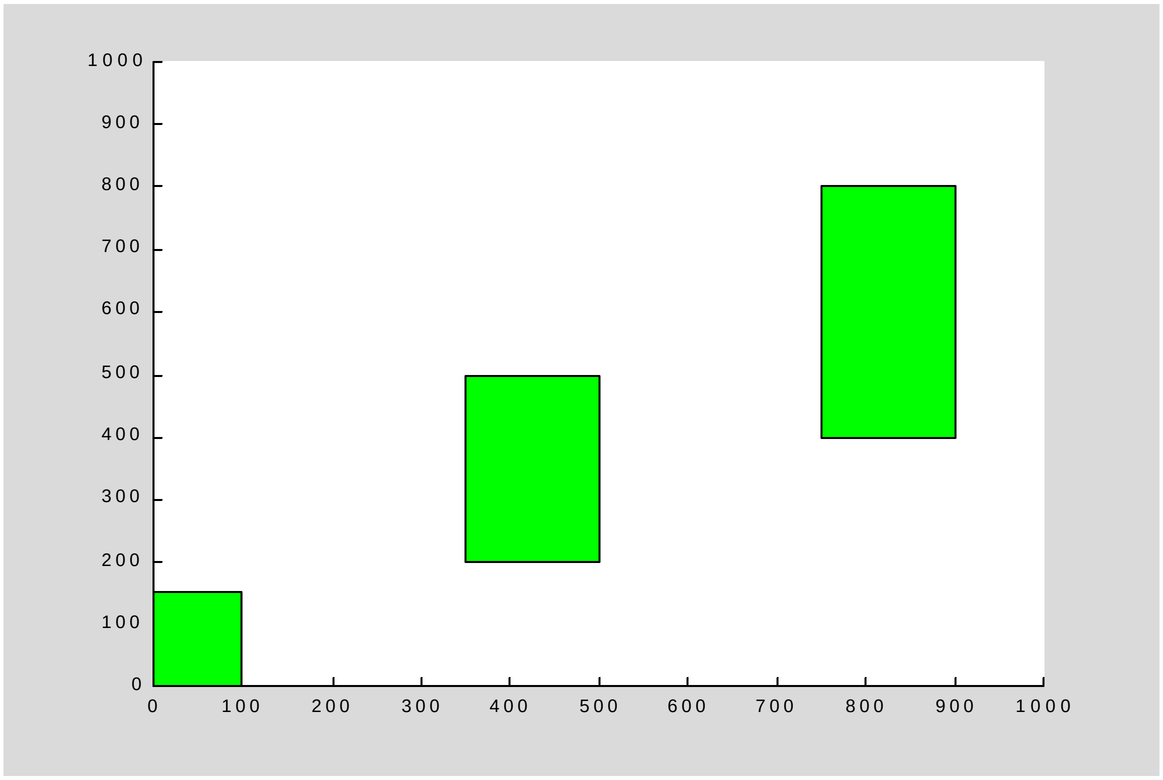
```
function road_median(xA,yA,xC,yC)
% MATLAB function to draw medians
% (xA,yA) and (xC,yC) are the coordinates of the lower left and upper
% right corners
% (0≤xA,xC≤1000 and 0≤yA,yC≤1000)

if xA>=0&yA>=0&xC<=1000&yC<=1000  % Check if vertices are within limits
  hold on
  v=[0 1000 0 1000];  % Set scale
  xB=xA;yB=yC; % Set coordinates of pt B (upper left)
  xD=xC;yD=yA;  % Set coordinates of pt D (lower right)
  x=[xA xB xC xD];
  y=[yA yB yC yD];
  fill(x,y,'g') % Draw green rectangle
  axis(v)
else
  warning('One or More Vertex Coordinates Outside of Limits')
end % if xA>0&yA>0&xC<1000&yC<1000

x1l=0;y1l=0;
x1u=100;y1u=150;
x2l=350;y2l=200;
```

```
x2u=500;y2u=500;
x3l=750;y3l=400;
x3u=900;y3u=800;
road_median(x1l,y1l,x1u,y1u)
road_median(x2l,y2l,x2u,y2u)
road_median(x3l,y3l,x3u,y3u)
road_median(x3l,y3l,x3u,y3u+300)
```

```
Warning: One or More Vertex Coordinates Outside of Limits
> In C:\MATLABR11\work\Matlab_Course\road_median.m at line 20
```



The next example contains a function which draws a traffic signal face at a location set in the calling script file. The script file also controls the face colors which are lit by virtue of input argument values passed to the function. The function is listed below followed by the script file.

Example 14.2.3

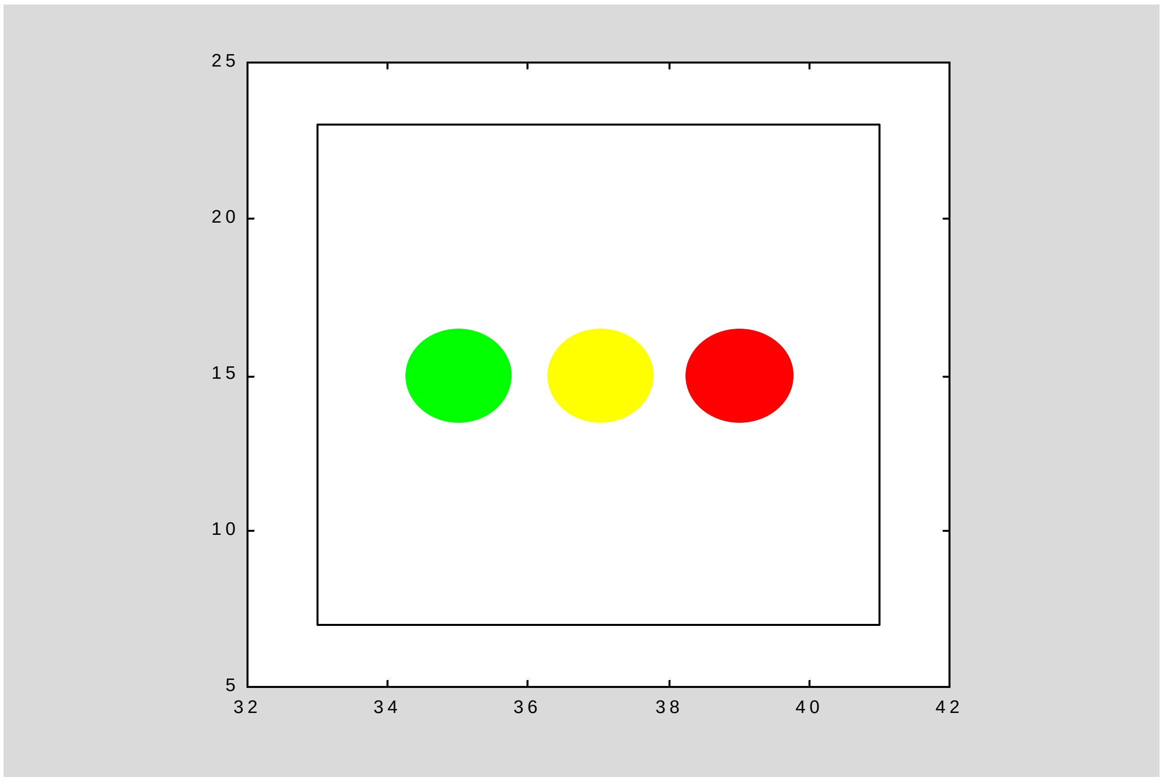
```
function TrafficSignal(x,y,visible_g,visible_y,visible_r)
%Traffic Signal plots traffic signal with green, yellow or red visible
% at location (x,y)

%%% BEGIN PLACEMENT OF TRAFFIC SIGNALS %%%
x_ts_g=x + 10; % x coordinate of traffic signal green light
y_ts_g=y - 10; % y coordinate of traffic signal green light
x_ts_y=x_ts_g + 2; % x coordinate of traffic signal yellow light
y_ts_y=y_ts_g; % y coordinate of traffic signal yellow light
x_ts_r=x_ts_y + 2; % x coordinate of traffic signal red light
```

```
y_ts_r=y_ts_y; % y coordinate of traffic signal red light
x_ts_A=x_ts_g - 2; % x coordinate of traffic signal lower left corner
y_ts_A=y_ts_g - 8; % y coordinate of traffic signal lower left corner
x_ts_B=x_ts_r + 2; % x coordinate of traffic signal lower right corner
y_ts_B=y_ts_A; % y coordinate of traffic signal lower right corner
x_ts_C=x_ts_B; % x coordinate of traffic signal upper right corner
y_ts_C=y_ts_B + 16; % y coordinate of traffic signal upper right corner
x_ts_D=x_ts_A; % x coordinate of traffic signal upper left corner
y_ts_D=y_ts_C; % y coordinate of traffic signal upper left corner
x_ts=[x_ts_A,x_ts_B,x_ts_C,x_ts_D,x_ts_A]; % Vector of x coordinates of
traffic signal corners
y_ts=[y_ts_A,y_ts_B,y_ts_C,y_ts_D,y_ts_A]; % Vector of y coordinates of
traffic signal corners
plot(x_ts,y_ts,'k') % Plot outline of traffic signal
hold on
plot(x_ts_g,y_ts_g,'o','MarkerSize',40,'MarkerEdgeColor','none','MarkerF
aceColor','g','Visible',visible_g)
plot(x_ts_y,y_ts_y,'o','MarkerSize',40,'MarkerEdgeColor','none','MarkerF
aceColor','y','Visible',visible_y)
plot(x_ts_r,y_ts_r,'o','MarkerSize',40,'MarkerEdgeColor','none','MarkerF
aceColor','r','Visible',visible_r)
axis square
axis([32 42 5 25])

%%% END PLACEMENT OF TRAFFIC SIGNALS %%%

 % Script file TrafficSignalCall.m
% Script file that calls function 'TrafficSignal.m' to position traffic
% signal face and control which colors are lit
TrafficSignal(25, 25, 'on', 'on', 'on')
```



14.3 Function Workspaces

Function m-Files create their own temporary workspaces for storage of variables created inside the function. The variables are local to the function workspace, meaning they are not accessible in the MATLAB Workspace or any other function workspace. This explains why the same name can be used for a variable in a script file, which is stored in the MATLAB Workspace, and a variable created in a function m-File, which is stored in the function's workspace. After the function executes, the function workspace is deleted and the variable remains only in the MATLAB Workspace.

In the example below, a scalar '`m`' and a 1×2 array '`pt`' are saved in the MATLAB Workspace and then passed to a function which uses the same variable names to create data for plotting a line.

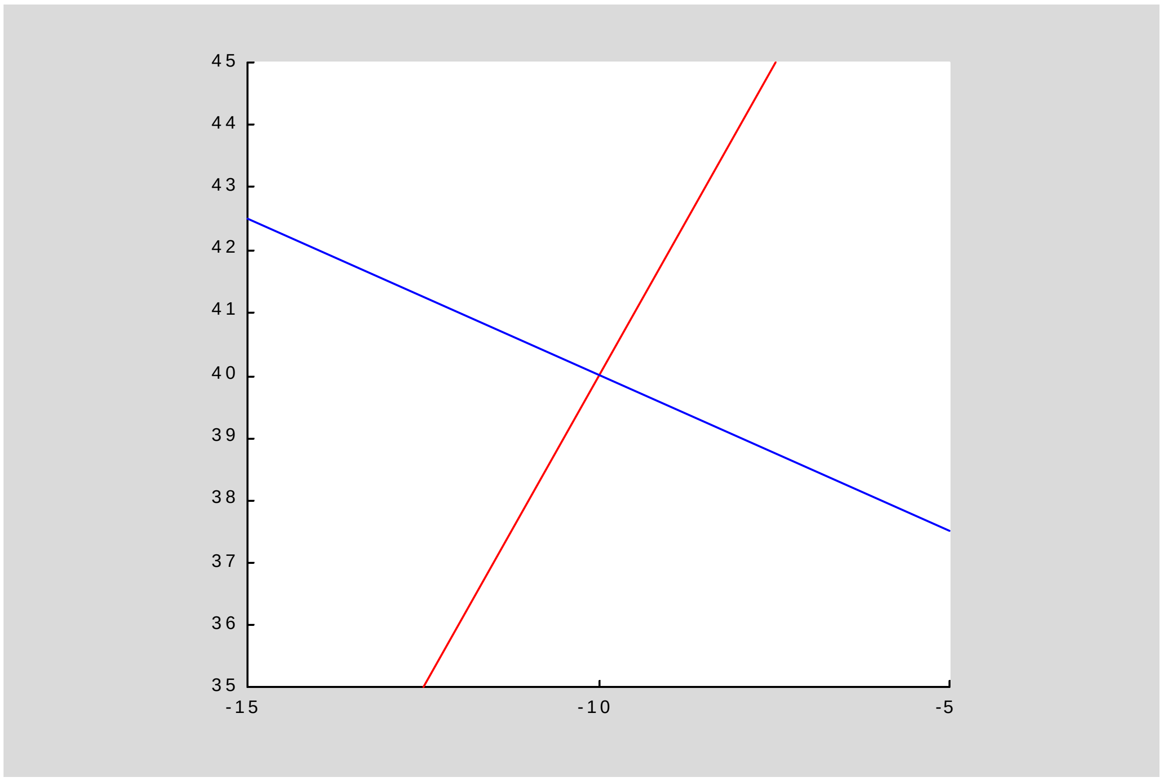
Example 14.3.1

```
function ortho_line(m,pt)
% This function receives the slope 'm' and a 1 by 2 array 'pt' of x and
% y coordinates for a line.  It calculates the slope of the orthogonal
% line through the same point and then plots the original line in red
% and the one perpendicular to it in blue.

x0=pt(1);  % Store 1st component of array 'pt' in x0
y0=pt(2);  % Store 2nd component of array 'pt' in y0
x=linspace(x0-5,x0+5,2); % Create 1 by 2 array x of end values
y1=y0 + m*(x-x0); % Find coresponding y values on line
m=-1/m; % Find slope of orthogonal line
y2=y0 + m*(x-x0); % Find corresponding values on orthogonal line
v=[x0-5,x0+5,y0-5,y0+5]; % Set scale for plotting
plot(x,y1,'r',x,y2,'b') % Plot both lines
axis(v)
axis square
```

The Command window statements are

```
m=2;
pt=[-10 40];
ortho_line(m,pt)
```



In this example, there are no variables created in the function 'ortho_line' and the function's workspace would ordinarily be empty. The numerical value of the MATLAB Workspace variable 'pt' is read by the function but the array itself does not get copied into the function's workspace. The variable 'm' on the other hand is copied into the function's workspace because it is modified inside the function.

If necessary, functions can share variables with the MATLAB Workspace if they are declared global in each workspace.

Example 14.3.2

```
% Script file cyl_call.m
% This file calls function cyl.m which returns the volume
% of a cylinder with radius R and height h. The radius R
% is declared global in both files.

global R
R=4; % Set value of global variable
h=10; % Input passed to function cyl.m
[vol,side]=cyl(h);  % Call function cyl.m with input h
R,h,vol,side
```

```
function [y1,y2]=cyl(h)
% This function is called by script file cyl_call.m  and calculates the
% volume of a cylinder with radius R and height h. The radius R is
% global in both files. It also calculates the side of a cube with the
% same volume.  The volume is returned in output y1 and the cube side
% is returned in output y2.

global R
y1=pi*h*R^2;  % Cylinder volume
y2=y1^(1/3);  % Side of cube
```

Running the script file cyl_call results in

```
R =      4
h =     10
vol =  502.6548
side =   7.9510
```

14.4 Function m-Files and the MATLAB Search Path

When MATLAB encounters `name` on the right hand side of an assignment in the MATLAB Command window or inside a script or function m-File, it follows a hierarchial set of rules to determine what to do.

1. It looks for `name` as a variable in the current workspace.
2. It checks for a built-in function `name`.
3. It checks for a subfunction `name` in the m-File where `name` appeared.
4. It checks if `name.m` exists in the current directory.
5. It checks if `name.m` exists in any of the directories on the MATLAB search path. The MATLAB search path is searched in the order in which it is specified.

When you create your own m-Files they should be stored in directories which have been added to the MATLAB search path.