**Ch.6 User‐Defined Functions and Function Files**

A simple function in mathematics, f(x), associates a unique number to each value of x. The function can be expressed in the form y=f(x), where f(x) is usually a mathematical expression in terms of x. A value of y (output) is obtained when a value of x (input) is substituted in the expression. Many functions are programmed inside MATLAB as built‐in functions, and can be used in mathematical expressions simply by typing their name with an argument; examples are **sin(x)** and **cos(x)**. Frequently, in computer programs, there is a need to calculate the value of functions that are not built‐in. When a function expression is simple and needs to be calculated only once, it can be typed as part of the program. However, when a function needs to be evaluated many times for different values of arguments it is convenient to create a “user‐defined” function. Once the new function is created (saved) it can be used just like the built‐in functions. In addition to being used as a math function, function files can be used as subprograms in large programs. In this way large computer programs can be made up of smaller “building blocks” that can be tested independently.

A user‐defined function is a MATLAB program that is created by user, saved as a function file, and then can be used like a built‐in function. The main feature of a function file is that it has an input and an output. The input and the output can be one or several variables, and each can be a scalar, vector, or an array of any size. Schematically, a function file can be illustrated by:

**Function File**

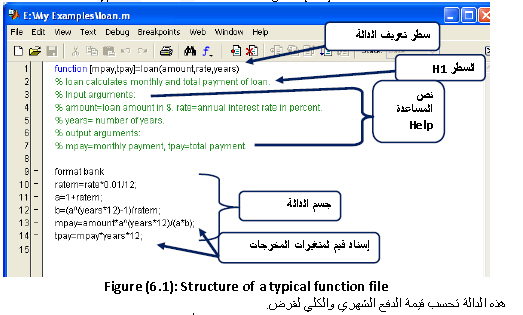
Input data Output

**6.1 Creating a Function File**

Function files are created and edited exactly the same way like script files in the Editor / Debugger Window.

**6.2 Structure of a Function File**

The structure of a typical function is shown in Figure (6.1).



The various parts of the function file are described in detail in following sections:

**6.2.1 Function Definition Line**

The first executable line in a function file must be the function definition line. Otherwise the file is considered a script file. The function definition line:

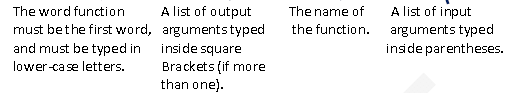
• Defines the file as a function file.

• Defines the name of the function.

• Defines the number and order of the input and output arguments.

The form of the function definition line is:

**function [output arguments ] = function\_name (input arguments)**



The name can be made up of letters, digits, and the underscore character. The rules for the name are the same as the rules for naming variables (see 1.8.1 pp.7). We must avoid names of built‐in functions, and names of variables already defined by the user or predefined by MATLAB.

**6.2.2 Input and Output Arguments**

The input and output arguments are used to transfer data into and out of the function. The input arguments are listed inside parentheses following the function name. Usually, there is at least one input argument, although it is possible to have a function that has no input arguments. If there is more than one, the input arguments are separated with commas. The computer code that performs the calculations within the function file is written in terms of the input arguments and assumes that the arguments have assigned numerical values. This means that the mathematical expressions in the function file must be written according to the dimensions of the arguments, since the arguments can be scalars, vectors, and matrices. The actual values of the input arguments are assigned when the function is used (called).

The output arguments transfer the output from the function file. Function files can have none, one, or several output arguments. If there is more than one, the output arguments are separated with commas or spaces. In order for the function file to work, the output arguments must be assigned values in the computer program that is in the function body. When a function does not have an output argument, the assignment operator in the function definition line can be omitted. A function without an output argument can, for example, generate a plot or print data to a file.

It is also possible to transfer strings into a function file. This is done by typing the string as part of the input variables (text enclosed in single quotes). Strings can be used to transfer names of other functions into the function file.

Usually, all the input to, and the output from, a function file are transferred through the input and output arguments. In addition, however, all the input and output features of script files are valid and can be used in function files.

**Example 6.1:**

The following are examples of function definition lines:

function A=Rect\_Area(a,b) دالة بمتغيرين داخلين ومتغير خرج واحد

function [A]=Rect\_Area(a,b) دالة بمتغيرين داخلين ومتغير خرج واحد

function[V,S]=Sphere\_vol\_Area(r) دالة بمتغير داخل واحد ومتغيرين خرج

**6.2.3 The H1 Line and Help Text Lines**

The H1 line and help text lines are comment lines (lines that begin with the percent % sign) following the function definition line. They are optional, but frequently used to provide information about the function. The H1 line (H1 stands for "help 1" line.) is the first comment line and usually contains the name and a short definition of the function. The help text lines are comment lines that follow the H1 line. These lines contain an explanation of the function and any instructions related to the input and output arguments. The comment lines that are typed between the function definition line and the first non‐comment line (the H1 line and the help text) are displayed when the user types **help function\_name** in the Command Window. This is true for the MATLAB built‐in functions as well as the user‐defined functions. A function file can include additional comment lines in the function body. These lines are ignored by the **help** command.

**Example 6.2:**

Consider the function “loan” in Figure (6.1):

>> help loan

loan calculates monthly and total payment of loan.

Input arguments:

amount=loan amount in $. rate=annual interest rate in percent.

years= number of years.

output arguments:

mpay=monthly payment, tpay=total payment.

**6.2.4 Function Body**

The function body contains the computer program (code) that actually performs the computations. The code can use all MATLAB programming features. This includes calculations, assignments, any built‐in or user‐defined functions, flow control (conditional statements and loops), comments, blank lines, and interactive input and output.

**6.3 Local and Global Variables**

Global variables are variables that, once created in one part of MATLAB, are recognized in other parts of MATLAB. This is the case for variables in the Command Window and script files since both operate on variables in the workspace. Once a variable is created, it exists, can be used, and can be reassigned a new value in both the Command Window and script file.

Function files are normally don’t share their variables with other parts of the program, that is their variables are **local**. This means that the variables are defined and recognized only inside the function file. When a function file is executed, MATLAB uses an area of memory that is separated from the workspace. In a function file the input variables are assigned values each time the function is called. These variables are then used in the calculations within the function file. When the function file finishes its execution the value of the output arguments are transferred to the variables that were use when the function was called. All of this means that a function file can have variables with the same name as variables in the Command Window or in script files. The function file does not recognize variables with the same name that have been assigned values outside the function. The assignment of values to these variables in the function file will not change their assignment elsewhere.

Each function file has its own local variables which are not shared with other functions or with the workspace of the Command Window and the script file. It is possible, however, to make a variable common (recognized) in several different function files and perhaps in the workspace too. This is done by declaring the variable **global** with the global command that has the form:

**global= variable\_name**

Several variables can be declared global by listing them, separated with spaces, in the global command.



**Notes:**

• The variable has to be declared global in every file that the user wants it to be recognized in. The variable is then common only to these files.

• The **global** command must appear before the variable is used. It is recommended to enter the **global** command at the top of the file.

• The **global** command has to be entered in the Command Window, or/ and in a script file, for the variable to be recognized in the workspace.

• The variable value can be assigned, or reassigned, a value in any of the locations it is declared common.

• It is recommended to use long descriptive names (or use all capital letters) for global variables in order to distinguish between them and regular variables.

المتغيرات الكلية هي متغيرات عندما تنشأ في جزء من ماتلاب تتعرف عليها أجزاء ماتلاب الأخرى. مثلاً المتغيرات

في نافذة الإيعازات والملف النصي بينما المتغيرات في ملف الدوال خاصة به وتدعى متغيرات محلية. هذه المتغيرات

تعرف ويتم التعرف عليها فقط داخل ملف الدالة وعندما تنفذ الدالة فان ماتلاب يستخدم حيزاً في الذاكرة منفصل عن

ساحة العمل (حيز الذاكرة لنافذة الايعازات والملفات النصية). في ملف دالة تسند قيم للمتغيرات الداخلة كل مرة يتم

فيها استدعاء الدالة ثم تستخدم هذه المتغيرات ضمن ملف الدالة وحينما ينتهي تنفيذ الملف فان قيم المخرجات تسند

للمتغيرات التي تم بها استدعاء الدالة. هذا يعني ان ملفات الدوال يمكنها ان تستخدم نفس أسماء المتغيرات في نافذة

الايعازات أو الملفات النصية لان ملفات الدوال لا تتعرف عليها وبالتالي لا تتغير قيمها.

**6.4 Saving a Function File**

A function file must be saved before it can be used. This is done by choosing **Save** or **Save as**…from the File menu, selecting a location, and entering a name for the file. It is highly recommended that the file is saved with name that is identical to the function name in the function definition line. In this way the function is called (used) by using the function name. Like script files, function files are saved with extension .m.

**6.5 Using a Function File**

A user‐defined function is used in the same way as a built‐in function. The function can be called from a Command Window, from a script file, or from another function. To use the function file, the directory where it was saved must be either be in the current directory or be in the search path.

A function can be used by assigning its output to a variable (or variables), as a part of mathematical expression, as an argument in another function, or just by typing its name in the Command Window or in a script file. In all cases the user must know exactly what the input and output arguments are. An input argument can be a number, a computable expression, or it can be a variable that has an assigned value. The arguments are assigned according to their position in the input and output argument lists in the function definition line.

**Example 6.4:**

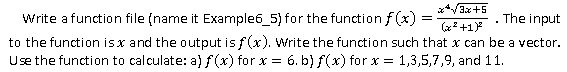
Find the monthly pay and the total pay for a loan of amount 25000 $ with interest rate 7.5% for 4 years.

**Solution:**

We can solve the problem in two ways:

|  |  |
| --- | --- |
| 1) >> [month total]=loan(25000,7.5,4)  month =  600.72  total =  28834.47 | 2) >> a=25000 ; b=7.5 ; c=4;  >> [x,y]=loan(a,b,c)  x =  **المدخلات لم تستخدم بقيمها العددية مباشرةً وانما عرفت كمتغيرات** **واستخدمت المتغيرات**  600.72  y =  28834.47 |

**Example 6.5:**



**Solution:**

The function file for the function f(x) is:

function y=Example6\_5(x) **سطر تعريف الدالة**

y=(x.^4.\*sqrt(3\*x+5))./(x.^2+1).^2; **تخصيص لمتغير الخرج**

Note that the mathematical expression in the function file is written for element‐by-element calculations. In this way if x is a vector y will also be a vector.

a) Calculating the function for x=6, can be done by typing Example6\_5(6) in the Command Window, or by assigning the value of the function to a new variable.

|  |  |
| --- | --- |
| >> Example6\_5(6)  **طباعة اسم الدالة والمتغير**  **المطلوب إيجاد قيمة الدالة له**  ans =  4.54 | >> A=Example6\_5(6)  **إسناد قيمة الدالة لمتغيرA**  A =  4.54 |

b) To calculate the function for several values of x, a vector with the values of x is first created, and then used for the argument of the function.

|  |  |
| --- | --- |
| >> x=1:2:11  x =  1 3 5 7 9 11  >> Example6\_5(x)  ans =  0.7071 3.0307 4.1347 4.8971  5.5197 6.0638 | >> B=Example6\_5(x)  B =  0.7071 3.0307 4.1347 4.8971  5.5197 6.0638  >> B=Example6\_5([1:2:11])  B =  0.7071 3.0307 4.1347 4.8971  5.5197 6.0638 |

**Example 6.6:**

Write a user‐defined function (name it F to C) that converts temperature in degrees F to degrees C. Use the function to solve the following problem. The change in the length of an object, ΔL, due to a change in the temperature, ΔT, is given by: ΔL=ΔT, where is the coefficient of thermal expansion. Determine the change in the area of a rectangular (4.5 m by 2.25 m) aluminum ( 23. 10-6 1/°C) plate if the temperature changes from 40°F to 92°F.

**Solution:**

A user‐defined function that converts degrees F to degrees C is:



A script file (named Example6\_6) that calculates the change of the area of the plate due to temperature is:

a1=4.5; b1=2.25; T1=40; T2=92; alpha=23e‐6;

**استخدام الدالة FtoC لحساب فرق الدرجات**

deltaT=FtoC(T2)‐FtoC(T1);

a2=a1+alpha\*a1\*deltaT;

**حساب الطول a2 و العرضb2 الجديدين**

b2=b1+alpha\*b1\*deltaT;

**حساب التغير في المساحة**

AreaChange=a2\*b2‐a1\*b1;

fprintf( ' The change in the area is %6.5f meters square.', AreaChange)

Executing the script file in the Command Window gives the solution:

>> Example6\_6

The change in the area is 0.01346 meters square.

**6.6 Inline Functions**

Function files can be used for simple mathematical functions, for large and complicated math functions that require extensive programming, and as subprograms in large computer programs. In cases when the value of a relatively simple mathematical function has to be determined many times within a program, MATLAB provides the option of using inline functions. An inline function is defined within the computer code (not as a separate file like a function file) and is then in the code. Inline function can be defined in any part of MATLAB.

Inline functions are created with the **inline** command according to the following format:

**name= inline ( ‘ math expression typed as a string ‘ )**

• The mathematical expression can have one or several variables. The format for inline

function with several independent variables is:

**name=inline ( ‘ mathematical expression ‘ , ‘ arg1 ‘ , ‘ arg2 ‘ , ‘ ag3 ‘ )**

• Any letter except i and j can be used for the independent variables in the expression.

• The mathematical expression can include any built‐in or user‐defined functions,

• The expression must be written according to the dimension of the argument.

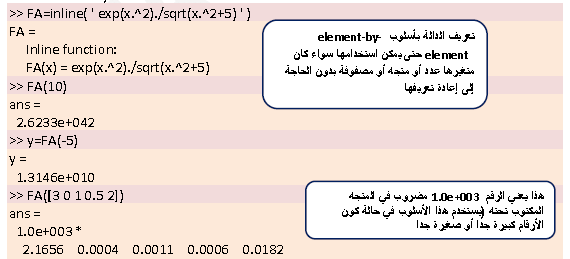
• The expression can include pre‐assigned variables.

• Once the function is defined it can be used by typing its name and value for the argument (or arguments) in parentheses.

• The inline function can also be used as an argument in other functions.

**Example 6.7:**

The function: f(x) = can be defined (in the Command Window) as an inline function for x by:



**6.7 The feval Command**

The feval (function evaluate) command evaluates the value of a function for a given value (or values) of the function’s argument (or arguments). The format of the command is:

**[y1,y2, …]=feval( ‘ function name ‘, x1,x2, … )**

• The function name is typed as a string.

• The function can be a build‐in or a user‐defined function.

الايعاز **feval** يستخدم ليجاد قيمة لدالة معينة لقيمة( أوقيم) معطاة والدالة يمكن أن تكون من دوال الماتلاب المعرفة أو دالة معرفة من قبل المستخدم.

**Example 6.8:**

Consider example (6.4):

>> [m,t]=feval('loan',25000,7.5,4)

**استخدام الايعاز feval لايجاد كمية الدفع الشهري والمبلغ المدفوع الكلي للقرض في المثال ( 6.4 ). لاحظ ان الدالة loan هي دالة معرفة من قبل المستخدم ومخزونة كملف دالة بالاسمloan.m .**

m =

600.72

t =

28834.47

**Example 6.9:**

Consider example (6.7):

>> FA=inline(' exp(x.^2)./sqrt(x.^2+5) ' )

**استخدام الايعاز feval لايجاد قيمة الدالة FA المعرفة من قبل المستخدم للعدد 10 وللمتجه ( 3,0,1,0.5,2). لاحظ استدعاء الدالة FA بدون وضع علامتي الاقتباس ‘ ‘ .**

FA =

Inline function:

FA(x) = exp(x.^2)./sqrt(x.^2+5)

>> feval(FA,10)

ans =

2.6233e+042

>> z=feval(FA,[3 0 1 0.5 2])

z =

2165.64 0.45 1.11 0.56 18.20

**Example 6.12:**

>> feval('sqrt',16)

**استخدام الايعاز feval لايجاد قيمة دالة الماتلابsqrt للعدد 16**

ans =

4