1.4 MATLAB: Functions

In MATLAB, there are two ways to handle subroutines: by script files or as functions. Script files use the variables of the current workspace, functions have their own variables (i.e., their own workspace). Both get activated by using their name in an appropriate expression. Some functions take arguments, others not; some deliver numerical results, others just do something (as, e.g., plotting) without returning any result.[7] Script files neither have arguments nor do they return anything.

MATLAB comes with many functions, ranging from generation of function values (as, e.g., `sqrt`, `log10`, `atan`, `erf` etc.) over matrix decompositions to numerical integration or solving differential equations. There are two ways for a user to create its own functions: A mathematical expression that can be evaluated in a single step can be declared as an *inline function* (or, from MATLAB 7 on, it can be coded as a so-called *anonymous function*[8]). For computing more intricate functions that require intermediate steps, the function's code can be stored in a so-called M-file, following a certain syntax. Placing this M-file in the proper directory makes the function available to MATLAB as if it were an intrinsic function.

The arguments of a function must be given to the right of the function name in a bracketed, comma-separated list. These arguments are never modified by MATLAB, so also expressions (that possibly involve function calls) can be used. Even more, there are several ways that a function itself can be transferred as an argument to another function.

If the function delivers an output value, this can be used in an expression, or be attributed to a variable by a usual assignment; otherwise the output is attributed to `ans`. Accordingly, if there are several output values, they can be attributed to a vector, denoted by a comma-separated list enclosed in square brackets.

For writing a function it is important to know that variables used in the function declaration are local to the function, i.e. they are kept in their own workspace and do not interfere with variables of the same name that are already present. Therefore, the latter cannot be accessed; this is called *shadowing*. An exception holds for functions that are defined as sub-functions inside a function; such a sub-function is called a *nested function*. Nested functions share the same variables (i.e. the same workspace) as their "nesting function."

In general, "when MATLAB performs a calculation, it does so using the values it knows at the time the requested command is evaluated."[9] Then, in situations where the sequence of the need for some data conflicts with the sequence of them becoming available, using nested functions is very helpful, since they allow handling such a situation without need for global variables. Although global variables can be declared in MATLAB, programming usually is less troublesome when they are not used, since they frequently lead to unexpected problems.

[7] In MATLAB there is no distinction between "functions" (having arguments and delivering results) and "procedures" (just doing something without returning a result). — Note that, strictly speaking, also plotting returns something, namely a so-called *handle* of the plotted object.

[8] An anonymous function is only anonymous in the sense that it does not have a regular name. Instead, it can be referenced only via an object called a *function handle*.

[9] Hanselman/Littlefield, *Mastering MATLAB 5*, page 9