Origin C Versus LabTalk

LabTalk is Origin's original scripting language. It allows access to many internal features of Origin including its internal objects, such as graphs, worksheets, and other windows, data plots and their properties, menus, etc.

Origin C is a new addition to Origin version 7. It is a compiled language, which has elements of C++, and is therefore object-oriented. The execution is faster than that of LabTalk scripts.

To assess the strengths and weaknesses of both languages, one can use the following comparison table:

Table 1: Comparison of LabTalk and Origin C

	LabTalk	**Origin C**
Compilation	No.	Yes (programs can also be saved to disk in a precompiled form for faster repeated use).
Speed	Interpreted language, so the speed is not as good, especially in case of loops with many iterations.	Compiled language, so the speed is much better (up to 20 times) than in LabTalk. It is especially well suited for long-winded calculations. It is also very good for defining one's own fitting functions for nonlinear curve fitting.
Access to internal Origin objects	Yes. As LabTalk has existed much longer than Origin C, in some respects the access is better than in Origin C.	Yes. The access is object-oriented, although, as a new language, it may not cover as much internal objects and properties as LabTalk. This should improve in the future.
Case sensitivity - whether commands and variables must be in the correct (upper or lower) case	No. For example, variable `a` and variable `A` are considered to be the same variable.	Yes. Origin C is case sensitive which means, for example, that variable `A` and variable `a` are considered to be different variables. The same is true with function names.
Functions	No. LabTalk has the ability to call sections in script files with .OGS extensions, which allows passing of simple text arguments.	Yes. Standard rules of C language for calling functions apply. This is much more convenient than calling script file sections in LabTalk.
How to execute	LabTalk scripts are usually organized in script files with the extension .OGS by means of sections. They can be called using the command `run.section()` from either Script window or from another LabTalk script. Also, LabTalk scripts can be typed directly into the Script window and executed from there, or they can be associated with the menu items or toolbar buttons, or with buttons on various Origin windows (graphs, worksheets, etc.). LabTalk scripts can also be executed from Origin C functions.	Origin C code is always organized in functions. Functions can be called from other Origin C functions in the standard way by passing arguments of different types. They can be called from Script window, from LabTalk scripts, from menu items and toolbars buttons, as well as from buttons on various Origin windows (graphs, worksheets, etc.).

	LabTalk	Origin C
Variable types	Yes. Only numeric (double precision) and a limited number of string variables are supported. Variables representing internal Origin objects are not supported. Instead, it is possible to refer to various global objects, such as the active window, layer, etc.	Yes. All standard C types are supported, as are pointers. Also, variables representing internal Origin objects are supported (access to those objects is object-oriented). All variables must be declared before being used.
Local variables	No.	Yes. Local variables in functions must be declared before being used, as it is standard for C language.
Global variables	Yes. All variables in LabTalk are global variables. All global variables are either numeric (they do not have to be declared before being used since they are defined and space in memory is reserved for them once they are used for the first time) or string (there are 26 string variables in LabTalk, named %A, %B, etc., some of which are reserved, such as %H, which always contains the name of the active window).	Yes. All global variables must be declared outside functions before being used.
Multidimensional objects	No.	Yes. Origin C supports `vector` and `matrix` classes (and the associated classes `Dataset` and `Matrix` which can be used to access Origin's internal datasets and matrices) which can be dereferenced using `[]` notation (`vector v;…;. v[3] = …;`) to access individual elements.
Collections	No.	Yes. Origin C supports many collections of internal Origin objects, such as collection of all windows, all columns in a worksheet, all data plots in a graphic layer, etc. Collections allow easy enumeration and access of the items being held in the collection.
Control structures	LabTalk supports C-like if-else and switch statements. It also supports C-like for-loop, as well as LabTalk-specific `repeat` and `loop` looping control structures.	It supports all C-style control structures (`if-else`, `switch`, `for`, `while`, `do-while`, `goto`). It also supports `foreach` loop, which can be used to easily enumerate all the members of a collection.
Writing user-defined fitting functions	Yes.	Yes. Here the speed of compiled Origin C functions becomes very important.
Calling external functions (functions written in external dynamic link libraries)	No.	Yes. A function implemented in an external DLL (the function must be exported from the DLL in a standard way) can be called from Origin C. This enables proprietary analysis and other routines that users have implemented in standard Windows DLLs to be applied to Origin's data. To make a call is as simple as calling another Origin C function.