[[1]](#footnote-1)

Fortran against C++ and Its Applications

Ryan Schwartz

*Abstract*—Fortran is a programming language developed initially for highly involved mathematical operations in the early 1950s. Although thought to be unused and outdates, Fortran is still very important to the continuing development of important, accuracy-dependent problems of todays world, like with rockets and nuclear reactors. There are key differences between the levels of accuracy Fortran provides and what a language like C provides. Fortran has just seen a new 2018 release with new features, along with IEEE standard 128 bit floating point precision, capable of beating C when it comes to accuracy.

# Introduction to Fortran

M

any programs used by engineers, scientists, or doctors are required to compute some heavy mathematical operations. These operations can vary from calculating current of a circuit to calculating the beats per minute of a heart pacemaker. If a calculation needs precision – *true precision –* then Fortran should be used to program these applications. Extensive, accurate, and highly optimized applications such as some types of weather forecasts, financial trading algorithms, and in-depth engineering simulators are created using Fortran due to its verbatim-like programming-to-assembly language relationship and high precision (Valters, n.d.).

# Fortran versus C

In the field of modern computing, typically only C++ and Fortran are considered when dealing with large scale numerical calculations. Fortran usually has the opinion of the public that it is much faster in everything than C, but is just harder to code. This is false. According to a benchmark on Debian OS, Fortran and C are the fastest modern coding languages (Elton, 2015). Nevertheless, C++ beats Fortran on all metrics except n-body simulation and spectra calculations (heavy physics topics). C++ seems to work more efficiently and faster on operating systems utilizing a quad core machine, whereas Fortran has not seen many evolutionary updates when it comes to recent technology advancements, such as multi-core machines. Also, Fortran seems to be much slower than C++ with any processes involving reading and writing data. Despite being equal to or slower than C++, Fortran still has the favor of easier coding syntax’ than traditional C++. According to Elton, it is generally easier for physicists to write Fortran programs than it is to write C++ because Fortran forces you to think in more of a physical, literal, lower level mind set – which physicists are great at. Physicists (and other engineers) are more likely to think about how the transfer of data from RAM and from RAM to the CPU will develop. This makes coding Fortran a lot simpler, as opposed to learning the ever-growing C++ syntax along with dynamic class capabilities, etc.

To add, Fortran variables are almost always passed by *reference* instead of by *value.* A Fortran compiler can skip the copying of data and go straight to the optimization of the computation. Fortran also allows its coders to declare the *intent* of a variable, allowing user warnings to arise if that variable is somewhere where it should not be. C++ has similar, more-difficult capabilities (Hardwick, 2011).

# Applications of Fortran

Fortran is really applicable in problems with large-scale physics simulations, such as astrophysical modeling of stars and galaxies, hydrodynamics, molecular dynamics, and large scale climate models (Elton, 2015).

An example already introduced is a climate simulator and weather forecaster. Many of these types of models are written in Fortran because of the amount of intense accuracy and precision that needs to take place behind the scenes in order to get correct results. Recent Fortran updates revolutionized the language altogether, allowing for IEEE 128 bit floating point precision, whereas C++ has 64 bit precision (Wikipedia.org, 2021). Weather such as rain, snow, heal, wind, temperature, geography, and pressure *each* have *their own* physics simulators to solve that one independent model. For anyone trying to *combine* all of these parameters, Fortran is the way to go for computational correctness.

Another example is CAD software, or Computer Aided Design software. Popular products such as AutoCAD, SolidWorks, ANSYS, and ADINA can all benefit from the uses of Fortran. In a paper done by Yu He Li and colleagues, the system integration between the machine architecture and Fortran can draw full physical bodies while fully utilizing the numerical calculation precision power of Fortran (Li, 2012). As an example, it can help develop 3D virtual automobile transmission gear teeth, perform all transmission capabilities, and create multiple output results for post-processing analysis with the highest precision of any programming language.

# Conclusion

Overall, Fortran is mainly used in place of the more modern programming language C++ mostly because it is generally easier for engineer minds to wrap their head around in applications involving physics simulations. If a system needs intensely accurate mathematical calculations, then, although C++ is not *too* far behind on accuracy, that extra boost of accuracy from Fortran will definitely be useful to some.

# References

Elton, D. C. (2015, July 16). *Why physicists still use Fortran*. Retrieved from moreisdifferent.com: https://www.moreisdifferent.com/2015/07/16/why-physicsts-still-use-fortran/

Hardwick, A. (2011, 12 29). *Messier Still - In the Object Orietned Programming*. Retrieved from https://www.duramecho.com: http://duramecho.com/ComputerInformation/WhyHowCppConst.html

Li, Y. H. (2012, December). *Computer Aided Design Based on Visual FORTRAN and AutoCAD*. Retrieved from https://www.scientific.net/AMR.628.245: https://www.scientific.net/AMR.628.245

Valters, D. (n.d.). *Introduction to Fortran*. Retrieved from https://ourcodingclub.github.io/tutorials/fortran-intro/.

Wikipedia.org. (2021, February 9). *Double-precision floating-point format*. Retrieved from Wikipedia.org: https://en.wikipedia.org/wiki/Double-precision\_floating-point\_format#C\_and\_C++

1. [↑](#footnote-ref-1)