Python in Computational Science: Applications and Possibilities

Md. Golam Rashed
Department of Civil Engineering,
Ahsanullah University of Science and Technology (AUST),
Dhaka-1208, Bangladesh

Raquib Ahsan, PhD
Department of Civil Engineering,
Bangladesh University of Engineering and Technology (BUET),
Dhaka-1000, Bangladesh

ABSTRACT
This paper focuses on the role of python in dramatic increase in productivity and high-level of code reuse in computational science. The salient features of python make it an ideal language for scientific computing exposing the shortcomings of legacy languages and prototyping platforms. Python provides a rich collection of built-in data types such as strings, lists, dictionaries; dynamic typing and dynamic binding, modules, classes, exceptions handling, automatic memory management, multiprocessing, parallel computing capabilities. Python can also be used as a glue language to wrap around existing static compiled code to obtain optimum performance. The uptrend of adopting python as a general purpose language along with its vast collection of scientific libraries are also reviewed in this paper, which ensures the long term presence of python and its growing user base in the foreseeable future.

General Terms
Computational Science.

Keywords
Engineering Simulation, Computational Science, Scientific Computing, Open Source, Python.

1. INTRODUCTION
Computational science is now considered as the third branch of science along with theoretical and experimental science. It is essentially comprised of numerical algorithm [1] and computational mathematics [2]. Substantial effort in computational science has been devoted to the development of algorithms, the efficient implementation of programming languages, and validation of computational results. Computational science application programs often model real world changing conditions, Complex Engineering phenomena is one of them. Such programs can be developed by either coding in high-level language or by combining scripting interface to existing compiled library to address the concern over computation time [3]. The ease of scripting paradigm and the convenience of code reuse along with extended readability have made python one of the popular tools in computational science. Python is now widely used in various sub fields of engineering simulation [4].

2. LANGUAGES OF COMPUTATIONAL SCIENCE
Determining the best high-level language for computational science is a highly disputed matter because high-level language is a rather relative term. The most commonly used programming languages in computational science to date are FORTRAN and C/C++.

FORTRAN was the first successful high-level programming language to be developed and it arrived in the 1950's [5]. Before the advent of FORTRAN, all programming used to be coded in assembly language. Moreover, FORTRAN was specifically designed for scientific computing. In the early days of computers most computing was scientific in nature with some minor computing in business purposes where physicists and mathematicians were the original computer scientists. FORTRAN's main advantages are that it is very straight forward, and it interfaces well with most commonly available, pre-written subroutine libraries since these libraries generally consist of compiled FORTRAN code. FORTRAN's main disadvantages are all associated with its relative antiquity. For instance, FORTRAN's control statements are very basic, whereas its I/O facilities are primitive in comparison to modern languages [6].

C was developed in 1970's by computer scientists to write operating systems [7]. Indeed, all UNIX operating systems are written in C. C is an extremely flexible and powerful language. Amongst its major advantages are its good control statements and excellent I/O facilities. C's main disadvantage is that, since it was not specifically written to be a scientific language, some important scientific features such as complex arithmetic are missing. Although C was considered a high-level language at the time of its inception, it incorporates many comparatively low-level features, such as pointers. But this is hardly surprisingly, since C was originally designed to write operating systems. The low-level features of C such as the primitive implementation of arrays sometimes make scientific programming more complicated, and undoubtedly facilitate programming errors. On the other hand, these features allow scientific programmers to write extremely efficient code. Since efficiency is generally the most important concern in scientific computing, the low-level features of C are advantageous [6].

C++ is a major extension of C whose main aim is to facilitate object-orientated programming which was developed in the 1980's [8]. Object-orientation is a completely different approach to programming than the more traditional procedural approach. It is particularly well suited to large projects involving many people who are each writing different segments of the same code. However, object-orientation represents a large, and somewhat unnecessary, overhead for the type of straightforward, single person programming tasks considered in computational science [6].

Fortran 90 arrived in the 1990's [9]. FORTRAN 90 is a major extension to FORTRAN 77 which does away with many of the latter language's objectionable features. In addition, many modern features, such as dynamic memory allocation, are included in the language for the first time. The major disadvantage of this language is the absence of an inexpensive compiler. There seems little prospect of this situation changing in the near future [6].

Of the above languages, we can immediately rule out FORTRAN, because of the shortcomings such as limited type checking, lack of extensibility, reliance on global data etc. FORTRAN has become domain specific as a calculation tool and is not a general purpose language. Also the archaic features of FORTRAN are too embarrassing to use in the 21st century complex simulation problems. Most graphical interface and visualization packages have no native FORTRAN support. C/C++ is the fastest and efficient language to code. Almost all visualization and user interface packages support the C/C++ language. But being a low-level language, it is very hard for scientists and engineers to cope with. MATHEMATICA and MATLAB made prototyping easy but they cannot be used in large-scale complex engineering problems where full control is needed to optimize the code [10]. Figure 1 shows that dynamically typed languages have gained popularity over the statically typed counterparts in the last decade. Among them popularity of python as a general purpose programming language as well as an ideal programming language for computational science is on the rise (Figure 2).

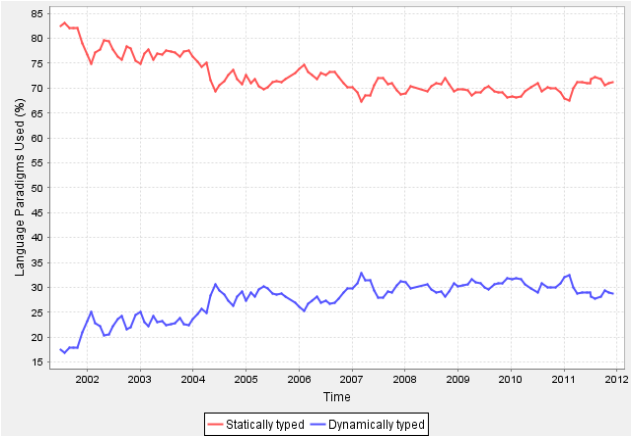

Figure 1: Uptrend of Dynamically typed language [11]

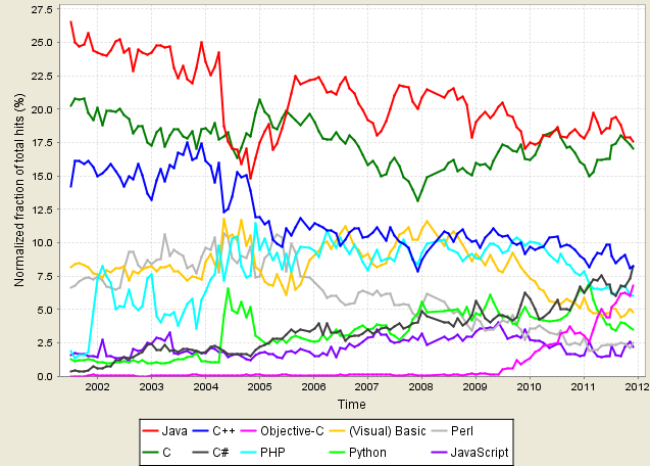

Figure 2: Increasing popularity of Python among the top 10 most popular languages [11]

3. PYTHON

Python is a general-purpose, high-level programming language with remarkable power and very clear syntax whose design philosophy emphasizes code readability and therefore reduces the cost of program maintenance. Python is the only major language to use indentation as a way of creating code blocks. This makes Python codes look similar regardless of whoever wrote it, which increases code readability [12]. Python supports multiple programming paradigms such as object-oriented, imperative and functional programming styles. Its standard library is large and comprehensive. It features a fully dynamic type system and a cycle detecting garbage collector for automatic memory management, similar to Scheme, Ruby, Perl, and Tcl. Python is often used as a scripting language like other dynamic languages, but is also used in a wide range of non-scripting contexts. Python code can be packaged into standalone executable programs using third party tools.

Python was designed to be highly extensible instead of requiring all desired functionality to be built into the language's core. New built-in modules can be easily written with C, C++ or Cython [13]. Python can also be used as an extension language for existing modules and applications that need a programmable interface. That is why python is being adopted as the best language to model complex engineering phenomena in today's scientific computing [14].

Being portable, Python runs on essentially all UNIX systems, as well as on DOS/Windows platforms and on the Mac. The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms and may be freely distributed. Additionally, many free third party Python modules, programs and tools, and documentations are widely available.

Debugging in python is very easy since there is no compilation step; the edit-test-debug cycle is very fast. The interpreter raises an exception upon discovery of an error. The interpreter prints a stack trace when the program doesn't catch the exception. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on [15].

3.1 Comparing Python to Other Languages

The advantage of python over legacy languages in computational science is clearly evident from the comparison presented in table 1 among C/C++, FORTRAN and Python. Many other languages such as Java, Visual Basic, Perl, Ruby, Tcl, smalltalk, COBOL, Ada, Algol, Pascal, Haskell and Common Lisp and Scheme are also used in scientific computing but they are too specialized to adopt for scientific use.

While Visual Basic is easy to read and understand by scientists, it runs only in windows and does not provide with fast execution speed. Consequently, interest in Visual basic has waned over the time.

Python programs are generally expected to run slower than Java programs, but they also take much less time to develop. Python programs are typically 3 to 5 times shorter than equivalent Java programs because of Python's built-in high-level data types and its dynamic typing [16].

Although Python and Perl came from the same UNIX scripting environment, they have different philosophy. Perl emphasizes support for common application-oriented tasks while Python emphasizes support for common programming methodologies such as data structure design and object-oriented programming which makes Python applicable well beyond Perl's niche [16].

Tcl traditionally stores all data as strings, is weak on data structures, and executes typical code much slower than Python. However, Tcl 8.0 addresses the speed issues by providing a bytecode compiler with limited data type support, and adds namespaces. However, it is still a much more cumbersome programming language [16].

Smalltalk, Common Lisp and Scheme are similar to python due to their dynamic typing nature but are far away from python's philosophy when it comes to syntax, built-in data types and data structure [16].

Table 1: Comparison of C/C++, FORTRAN and Python [10]

Language	C/C++	Fortran	Python
Intended use	Application, System	Application, Numerical Computing.	Application, General, Web, Scripting.
Paradigm	Imperative, procedural, object-oriented (C++).	Generic, imperative, object-oriented, procedural.	Aspect-oriented, functional, imperative, object-oriented, reflective.
Type strength	strong	strong	strong
Type safety	unsafe	safe	safe
Expression of types	explicit	explicit	implicit
Type checking	static	static	dynamic
Failsafe I/O	No	No	Yes
Statements ratio	1 / 2.5	2	6
Lines ratio*	1	0.8	6.5

* The ratio of line count tests won by each language to the number won by C when using the Compare to feature at Shootout.alioth.debian.org. C gcc was used for C, C++ g++ was used for C++, and FORTRAN G95 was used for FORTRAN.

3.2 Advantages of Python in Scientific Computing

Python is used as the core language of many scientific software covering fields of Astronomy, Artificial intelligence & machine learning, Bayesian Statistics, Biology, Neuroscience, Dynamical systems, Economics and Econometrics, Electromagnetism, Electrical Engineering, Geosciences, Molecular modeling, Signal processing, number theory etc. [17]. Python has been successfully embedded in a number of software products as a scripting language, including in finite element method simulation software such as Abaqus and in geographic information system application ArcGIS, which are widely used in engineering community [18].

3.2.1 Code re-uses & Speed

There is a vast collection of well tested and optimized numerical codes such as BLAS & Lapack, written in FORTRAN or C [19] [20]. Code re-use would mean to integrate the pure computing parts of such codes with new developments in another language. Combining C++ and FORTRAN, or Java and C, quickly gives a lot of complexity for the differences in data structures. Python offers the benefits of object-oriented and generic programming, together with a syntax that is simpler and clearer than C++ and Java. Also, there are several tools which make calling FORTRAN, C code easy. Hence, the idea is to write the managing code segments in Python, using efficient data structures and algorithms in new or old FORTRAN, C/C++ code.

User time is more valuable than CPU time for prototyping scientific code, so an interpreted language like Python is acceptable. Python executes bit slower than C/C++, but it makes that up with high-level coding with extremely readable, simple and elegant syntax while reducing coding time. In addition, Python variant Cython can be used to obtain C/C++ like speed. One should have clear conception about where, when and how much performance is needed. For extreme performance, existing C, C++ and FORTRAN codes can be wrapped easily with python [13].

3.2.2 Parallel computing

Various Python projects are currently underway that provide different parallel architectures, including shared memory architectures and message passing interface for distributed memory architectures [21]. Vendors of algorithms for high performance computing applications are recognizing the growth in Python and providing options for customers parallelization needs.

3.2.3 Graphical user interfaces

Python has various graphical user interface (GUI) frameworks available; from the native Tkinter to a number of other cross-platform solutions such as Gtk, Qt, Tk and wxWidgets. GUI programming in Python means that adding cross-platform GUIs on top of a scientific application which is an efficient process that requires much less code than in C/C++ [22].

3.2.4 Scientific libraries

Python has all major scientific libraries available either as the standard library or as third party open source library. SciPy is an open source library of algorithms and mathematical tools for the Python programming language. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, Fast Fourier Transformation, signal and image processing, ODE solvers and other tasks common in science and engineering [17]. It has a similar audience to applications such as MATLAB, MATHEMATICA, GNU Octave, and Scilab. It also includes a library called Weave that makes it easier to include C++ code in Python and compliments other solutions such as SWIG and F2Py for FORTRAN to Python binding. The basic data structure in SciPy is a multidimensional array provided by the NumPy module. NumPy is a Python package that provides extended math capabilities [23]. These include new data types, such as long integers of unlimited size and complex numbers. It also provides a new array data type that allows for the construction of vectors and matrices. All the basic operations that can be applied to these new data types also are included. SymPy is a symbolic manipulation package, written in pure Python. Its aim is to become a full featured computer algebra system in Python, while keeping the code as simple as possible in order to be comprehensible and easily extensible [24]. The Python and C++ interfaces to the vtk library for visualization of stationary and time-dependent scalar and vector fields in 2D and 3D; results in a tool that can give the researcher the best of all worlds, complete programming control for automation and real-time

visualization, or User friendly visual construction of visualization pipelines [25]. Python Imaging Library with various filters built-in provides basic image handling and processing for various image types including jpg, gif, tiff, and bmp; Reads and writes graphics files, Allows pixel by pixel data access and has functions for cropping and transposing an image. Matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms [26]. Mayavi has complemented the 3D visualization capability in python during the post processing phase of an engineering simulation [27]. The two major scientific library distribution packages are Pythonxy and Enthought Python distributions, which offer all major scientific libraries, bundled into one single package to use in multiple platforms [14].

4. CONCLUSION

Python is an accepted high-level scripting language with a growing community in academia and industry. It is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. The only downside for python is the slowness associated with it due to it being an interpreted language. But python makes it up by providing elegant syntax and dynamic typing, together with coupling capabilities with existing libraries which makes it an ideal language for computational science programming.

This paper focuses on the impact that the introduction of Python programming language can have on the computational science programming practice where emphasis is given on static languages in compiled form. A review on increasing popularity and rate of adoption of python in scientific community is also presented. The use of python in computational science programming offers three main benefits:

1. Python code is highly readable, which results in easy code maintenance and future development.
2. Dynamic typing in python provides on the fly prototyping, no need to compilation.
3. Use of python as a glue to existing compiled code in static languages saves time and offer high degree of code reuse.

5. REFERENCES

[1] Nonweiler, T. R. (1986). Computational Mathematics: An Introduction to Numerical Approximation. John Wiley & Sons Inc.

[2] Yang, X.-s. (2008). Introduction To Computational Mathematics. World Scientific Publishing Company.

[3] Rashed, M. G. (2012). Development and Evaluation of An Open Source Finite Element Analysis Framework. Proceedings of the 1st International Conference on Civil Engineering for Sustainable Development. Khulna: KUET, Bangladesh.

[4] Hassan, S., Rao, L. C., & K, G. B. (2012). Script Enhanced Unit Cell Approach for the Simulation of Compressive Behaviour in Fiber Reinforced Cement Composites. International Journal of Computer Applications , 44 (20), 32-37.

[5] McCracken, D. D. (1961). A Guide to FORTRAN Programming. New York: Wiley.

[6] Fitzpatrick, R. (2011, 03 31). Introduction to Computational Physics. Retrieved 04 20, 2012, from Home Page for Richard Fitzpatrick: http://farside.ph.utexas.edu/teaching/329/lectures/lectures.html

[7] Ritchie, D. M. (1993). The Development of the C Language. The second ACM SIGPLAN History of Programming Languages Conference (pp. 201–208). New York: ACM.

[8] Stroustrup, B. (2000). The C++ Programming Language. Addison-Wesley.

[9] Akin, E. (2003). Object Oriented Programming via Fortran 90/95. Cambridge: Cambridge University Press.

[10] Rashed, M. G., Ahsan, R., & Chowdhury, S. R. (2012). Numerical Modelling of Concrete Tensile Strength Test by Wrapping Scripting Language with Compiled Library. International Journal of Computer Applications , 40 (14), 34-38.

[11] TIOBE Programming Community Index. (2012, 04 20). Retrieved 04 20, 2012, from TIOBE Software: The Coding Standards Company: http://www.tiobe.com/index.php/tiobe_index

[12] Millman, K. (2011). Python for Scientists and Engineers. Computing in Science & Engineering , 13 (2), 9 - 12.

[13] Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D., & Smith, K. (2011). Cython: The Best of Both Worlds. Computing in Science & Engineering , 13 (2), 31 - 39.

[14] Pérez, F., Granger, B., & Hunter, J. (2011). Python: An Ecosystem for Scientific Computing. Computing in Science & Engineering , 13 (2), 13 - 21.

[15] Rossum, G. v. (2012, 04 20). What is Python? Executive Summary. Retrieved 04 20, 2012, from Python Documentation Index: http://www.python.org/doc/essays/blurb/

[16] Rossum, G. v. (1997, 12 30). Comparing Python to Other Languages. Retrieved 4 20, 2012, from Python Programming Language – Official Website: http://www.python.org/doc/essays/comparisons/

[17] SciPy. (2012, 4 20). Retrieved 4 20, 2012, from SciPy: http://www.scipy.org/Topical_Software

[18] Butler, H. (2004). A Guide to the Python Universe for Esri Users. 24th Annual Esri International User Conference. User Conference Proceedings.

[19] Duff, I. S., Heroux, M. A., & Pozo, R. (2002). An overview of the sparse basic linear algebra subprograms: The new standard from the BLAS technical forum. ACM Transactions on Mathematical Software , 28 (2), 239-267.

[20] Anderson, E., Bai, Z., Bischof, C., Blackford, L. S., Demmel, J., Dongarra, J. J., et al. (1999). LAPACK Users' guide (third ed.). Philadelphia: Society for Industrial and Applied Mathematics.

[21] Bruaset, A. M., & Tveito, A. (Eds.). (2006). Numerical Solution of Partial Differential Equations on Parallel Computers. Berlin: Springer.

[22] Summerfield, M. (2007). Rapid GUI Programming with Python and Qt. New Jersey: Prentice Hall.

[23] van der Walt, S., Colbert, S., & Varoquaux, G. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. Computing in Science & Engineering , 13 (2), 22 - 30.

[24] Joyner, D., Čertík, O., Meurer, A., & Granger, B. E. (2011). Open source computer algebra systems: SymPy. ACM Communications in Computer Algebra , 225-234.

[25] Kloss, G. K. (2009). Python Data Plotting and Visualisation Extravaganza. Proceedings of the First Kiwi PyCon (New Zealand). Christchurch: The Python Papers Monograph.

[26] Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science and Engineering , 90-95.

[27] Ramachandran, P., & Varoquaux, G. (2011). Mayavi: 3D Visualization of Scientific Data. Computing in Science & Engineering , 13 (2), 40 - 51.