introduction

Scientific computation is a field of science that utilizes advanced computing power to understand and solve complex problems. It includes the development and application of mathematical models, algorithms, and software that can perform numerical calculations, simulations, and data analysis in various scientific disciplines such as physics, chemistry, biology, and engineering. Scientific calculation can be seen as the third mode of science, along with theory and experimentation, which can complement and enhance the process of scientific discovery.

Scientific computing is a branch of science that uses high-performance numerical computing power to understand and solve complex real-world phenomena. This includes the development and application of mathematical models, algorithms, and software to perform numerical analysis, simulation, and data analysis in various scientific disciplines such as physics, chemistry, biology, and engineering. Scientific computing is being used as a third tool in the field of science, along with theory and experimentation, and is widely used with the development of high-performance computers as it can complement and enhance the process of scientific discovery. In recent years, with the development of AI/ML, the use of scientific computing in social science research such as economics and psychology has been increasing.

Scientific computing requires background knowledge and skills in various fields such as mathematics, computer science, programming languages, software engineering, and hardware architecture. In addition to this basic knowledge, high-performance computing resources such as supercomputers, HPC clusters, and cloud computing are required. At the same time, domain knowledge of the field in which scientific computing is to be used is essential.

For this reason, it is essential to have an appropriate curriculum on mathematical knowledge and algorithms in each field that is the background of scientific computing, as well as programming languages to implement them, along with the domain knowledge, and a step-by-step curriculum on IT knowledge and technology such as hardware architecture to carry out this.

In this article, we will look at programming languages and numerical libraries that are widely used in scientific computing and how to educate about them, and how to teach and learn about IT knowledge and skills related to Shuker Computer, HPC cluster, and cloud computing.

Typical Programming Languages for Scientific Computing

Depending on the scientific computing application, the hardware on which the application is performed, and the programmer's preference, the programming language suitable for scientific computing may vary. Nonetheless, some of the most popular programming languages that have traditionally been widely used in scientific computing and are still in use today are:

1) Fortran

Fortran is one of the 10 fastest-performing programming languages (https://www.geeksforgeeks.org/top-10-fastest-programming-languages/) for scientific computation, making it faster and more efficient than other general-purpose programming languages for differentiating large arrays of data. For this reason, it was developed to be used for programming related to numerical computation, and it is a representative programming language in the field of scientific computing that is compatible with many numerical computation-related libraries and codes through a long history and optimization.

2) C/C++

C and C+ are low-level programming languages that allow you to fine-tune your program's memory management and interaction with hardware. C and C+ are widely used languages for programming systems such as operating systems, drivers, and kernels, or for high-performance applications that require speed and flexibility, with widespread support for many compilers, libraries, and frameworks that enable parallel and distributed computing.

3) Python

Python is a high-level language that's easy to read, write, and learn, with a large and rich library and modules that provide a variety of functions for data analysis, machine learning, visualization, web development, and more. It's also an interpreted language that doesn't need to be compiled before it can be executed, making it more portable and easy to use. However, because it is slower than compilation languages such as Fortran or C/C++, it is often used as a scripting language or as a language for developing interfaces that call other languages or libraries rather than for performance-critical tasks.

These are some of the widely utilized and popular programming languages for scientific computing. However, various programming languages are used according to the requirements of each field and the tastes of developers. For example, Julia, R, and MATLAB are used in various fields of scientific computing. The choice of programming language can depend on a number of factors, including the area of the problem you want to solve, the resources available, the proficiency of the programmer, and the trade-off you want between speed and reliability. Therefore, it is not possible to specify the best programming language for scientific computing, but there are various programming languages that can be used depending on the situation.

Ranking Programming Languages for Scientific Computing

There are programming languages that are suitable for various scientific computing problems to be solved, such as numerical analysis, matrix solvers, and data analysis, and the ranking of these languages can vary depending on the criteria, sources, and timing of the ranking. A representative ranking of programming languages for scientific computing can be found in the following results.

A 2022 survey by IEEE Spectrum, a magazine covering technology and engineering topics (https://spectrum.ieee.org/top-programming-languages-2022), the top 10 programming languages for scientific computing are Python, C, C++, Fortran, R, Julia, MATLAB, Java, JavaScript, and SQL. The rankings are a combination of multiple metrics from a variety of sources, including web searches, social media mentions, job postings, and open source projects, and may change as you adjust the weighting of other metrics and filter by different platforms and domains. In addition, the 2023 survey (https://spectrum.ieee.org/the-top-programming-languages-2023), Python is still the most popular programming language. Python is widely used not only in scientific computing, but also in fields such as AI due to its powerful and extensive library. In addition, Python is provided with enough computing power to use Python on inexpensive CPUs, making it useful for embedded development. The next most popular languages are SQL, C, Java, C++, and R.

Slant (https://www.slant.co/topics/16647/~languages-for-scientific-computation), a website that provides recommendations for a wide range of products and services based on user reviews and votes, recommends Julia, Python, C, Fortran, R, Nim, Wolfram Mathematica, C++, and MATLAB as the best languages for scientific computing in 2023. These rankings take into account users' opinions and experiences, as well as the strengths and weaknesses of each language. It also provides links to tutorials and resources for learning each language.

According to an online education platform called Flatiron School, the top 10 programming languages (https://flatironschool.com/blog/data-science-programming-languages/) used in the field of data science in 2021 are Python, JavaScript, Java, R, C/C++, SQL, MATLAB, Scala, SAS, and Ruby.

According to Robotical.io https://robotical.io/blog/the-10-best-data-science-programming-

languages-to-learn/, a website that provides robotics training and products for children and adults, the top 10 data science programming languages to learn in 2022 are Python, JavaScript, Java, SQL, R, Scala, Julia, C/C++, MATLAB, and SAS. The rankings take into account the ease of learning, usability, performance, and usability of each language in the field of data science.

Finally, according to edX (https://www.edx.org/resources/9-top-programming-languages-for-data-science), a non-profit online learning platform, the top nine programming languages used in the data science field are Python, R, SQL, Java, C/C++, MATLAB, Scala, Julia, and SAS. This ranking takes into account the characteristics, advantages, disadvantages, and use cases of each language.

As can be seen from the various surveys above, there is no clear answer or ranking of the best programming languages for scientific computing. Each programming language has its own advantages and disadvantages, depending on the problem domain, available resources, and the skill level of the programmer. Therefore, when choosing a programming language for scientific computing, it is important to consider the user's goals and preferences. However, considering various surveys and user preferences in the field of traditional computational science, recent data science, artificial intelligence, etc., Python, C/C++, MATLAB, and R can be seen as representative programming languages for scientific computing. At the same time, it can be seen that Julia is also highly recommended as a programming language for scientific computing in many places.

IEEE Spectrum 2022	IEEE Spectrum 2023	Slant 2023	Flatiron School 2021	Robotical.io 2022	edX
Python C/C++ Fortran R Julia MATLAB Java JavaScript SQL	Python SQL C/C++ Java R	Julia Python C/C++ Fortran R Before Mathematica MATLAB	Python JavaScript Java R C/C++ SQL MATLAB Ladder SAS Ruby	Python JavaScript Java SQL R Ladder Julia C/C++ MATLAB SAS	Python R SQL Java C/C++ MATLAB Ladder Julia SAS

Numerical Library for Scientific Computing

Representative numerical libraries widely used in scientific computing and the languages that support them are as follows.

1) GNU Scientific Library (https://www.gnu.org/software/gsl/)

The GNU Science Library (GSL) is a numerical library for C and C++ programmers and is released under the GNU Public license. The library provides a variety of mathematical routines such as random number generators, special functions, least squares fit, and more, and includes a total of more than 1000 functions and an extensive test suite.

2) BLAS (Basic Linear Algebra Subprograms, https://www.netlib.org/blas/)

First developed in 1979 as the Fortran library, BLAS, a basic linear algebra subprogram, is a set of low-level routines for performing common linear algebraic operations such as vector addition, scalar multiplication, dot product, linear combination, and matrix multiplication, providing bindings for both C (the "CBLAS Interface") and Fortran (the "BLAS Interface"). BLAS provides libraries that are optimized for specific hardware (Intel MKL, NVIDIA CuBLAS, etc.) and can provide significant performance improvements. Many numerical software applications, including LAPACK, LINPACK, Armadillo, GNU Octave, Mathematica, MATLAB, NumPy, R, Julia, and Lisp-Stat, use BLAS-compatible libraries to perform linear algebra calculations.

3) LAPACK (Linear Algebra Package, https://www.netlib.org/lapack/)

LAPACK is a software library for performing linear algebraic operations, such as equation systems, eigenvalue problems, outlier problems, etc., written in Fortran, but also providing interfaces to C and C++. And linear algebraic functions and matrix operations in software such as MATLAB are based on LAPACK and continue to take advantage of the performance and accuracy of its routines. Like BLAS, LAPACK is supported in most popular programming languages for scientific computing.

4) NumPy (https://numpy.org/)

NumPy is a numerical library for the Python programming language that supports large multidimensional arrays and matrices, and provides a large collection of high-level mathematical functions that operate on these arrays.

5) SciPy (https://docs.scipy.org/doc/scipy/)

SciPy is Python-based, open-source software for math, science, and engineering that provides many modules and functions for various kinds of numerical calculations, including statistics, optimization, integration, linear algebra, Fourier transforms, signal and image processing, ODE interpreters, and more. SciPy is based on NumPy, a library for efficient array manipulation. In addition, specialized data structures such as sparse matrices and k-dimensional trees, as well as basic algorithms for scientific calculations, are extended based on NumPy. Numerical libraries such as BLAS and LAPACK are also supported through SciPy.

6) Pandas (https://pandas.pydata.org/)

Pandas is a fast, powerful, flexible, and easy-to-use open-source data analysis and manipulation tool based on the Python programming language.

Specialized software and languages for scientific computing

In addition to numerical libraries, the following software developed for numerical computation is widely used in the field of scientific computing.

1) MATLAB

MATLAB is a widely used commercial software for performing numerical calculations. MATLAB provides its own programming language, which is mostly compatible with open source software called GNU Octave. MATLAB also supports calling functions written in other languages, such as C, C++, Java, and Python, and provides many built-in functions and toolboxes for manipulating and interpreting matrices.

2) R

R provides a programming language and environment for statistical calculations and graphics, with many functions for manipulating data structures such as vectors, matrices, arrays, data frames, etc. It also includes many packages for performing numerical analysis, such as Matrix, lme4, MASS, etc., and provides functions such as solving linear systems, finding eigenvalues and eigenvectors, and calculating matrix decompositions.

3) Julia

Julia is a high-level dynamic language that is superficially similar to MATLAB, designed for high-performance computing, and supports multi-dispatch, parallelism, and metaprogramming. It also

includes packages for various fields such as data analysis, machine learning, and optimization, and provides a standard library for linear algebra and functions such as solving linear systems, finding eigenvalues and eigenvectors, and calculating matrix factorization.

Languages that support MPI programming

MPI programming is supported in a variety of languages, including C, C++, Fortran, Java, Python, R, and MATLAB. Some of these languages have native bindings to MPI, while others use external libraries or packages to access MPI functionality. Here are some examples of other languages using MPI:

1) C

If you want to use MPI in C, you need to include an mpi.h header file. MPI programs can be compiled with the mpicc compiler and run with the mpirun command. MPI functions begin with a MPI_ prefix. For example, MPI_Init, MPI_Comm_size, MPI_Send, MPI_Recv, etc.

2) C++

To use MPI in C++, you need to include the mpi.h header file, and use the mpiCC compiler and the mpirun instruction just like you do in C. In C++, you can use the MPI namespace to access MPI functions. For example, you can access . For example, MPI::Init, MPI::COMM_WORLD. Get_size, MPI::COMM_WORLD. Send, MPI::COMM_WORLD. Recv, etc.

3) Fortran

To use MPI in the Fortran language, you need to include the mpif.h module. MPI programs are compiled with the mpif90 compiler and executed with the mpirun command. MPI functions are capitalized, and arguments must be explicitly declared. For example, CALL MPI_INIT, CALL MPI_COMM_SIZE, CALL MPI_SEND, CALL MPI_RECV, etc.

4) Java

To use MPI in the Java language, you must add the mpi.jar file to the classpath. MPI programs are compiled like normal Java programs and executed with the mpirun command. In Java, you can import mpi packages to access MPI objects and methods. For example, MPI. Init, MPI.

COMM_WORLD. Size, MPI. COMM_WORLD. Send, MPI. COMM_WORLD. Recv, etc.

5) Python

To use MPI in the Python language, you need to install the mpi4py package. Python scripts can be written like regular Python scripts and run with the mpirun command. In Python, you can import the mpi4py module to access MPI objects and methods. For example, mpi4py. MPI. Init, mpi4py. MPI. COMM_WORLD. Get_size, mpi4py. MPI. COMM_WORLD. Send, mpi4py. MPI. COMM_WORLD. Recv, etc.

6) R

To use MPI in the R language, you need to install the Rmpi package. R scripts can be written like regular R scripts, and functions in the Rmpi package can be used to start and terminate MPI programs. In R, you can access MPI features using functions in the Rmpi package. For example, mpi.init, mpi.comm.size, mpi.send, mpi.recv, etc.

7) MATLAB

To use MPI in the MATLAB language, you must install Parallel Computing Toolbox. MATLAB scripts can be written like regular MATLAB scripts and run with the mpirun command from the command window in MATLAB. In MATLAB, you can use mpi packages to access MPI objects and methods. For example, mpi.init, mpi.comm.size, mpi.comm.send, mpi.comm.recv, and so on.

8) Julia

Julia has a package called MPI.jl that provides a Julia interface to MPI inspired by mpi4py. You can use the Pkg Manager to install MPI.jl and write parallel programs that communicate between multiple processes. You can also use the Distributed module in the standard library to implement distributed memory parallel computing in Julia.

As can be seen from the above, the programming languages widely used in scientific computing, such as Python, C/C++, Fortran, MATLAB, R, and Julia, support not only most numerical libraries but also MPI, which can be said to be the standard for parallel programming.

Scientific Computing Programming Language Textbook

The following textbooks have already been published for the use of Python, MATLAB, and R in scientific computing along with C/C++, and it is thought that it will be possible to prepare for the era of exascale supercomputing by teaching and learning scientific computing along with domain knowledge.

1) Guide to Scientific Computing in C++ (https://link.springer.com/book/10.1007/978-3-319-73132-2)

2) Scientific Computing with Python (https://www.amazon.com/Scientific-Computing-Python-High-performance-scientific/dp/1838822321)

3) Scientific Computing with MATLAB(https://link.springer.com/book/10.1007/978-3-642-59339-0)

4) Mastering Scientific Computing with R (https://www.amazon.com/Mastering-Scientific-Computing-Paul-Gerrard/dp/1783555254)

5) Python for the Scientific Computing https://aaltoscicomp.github.io/python-for-scicomp/

conclusion

As mentioned above, considering the ranking of programming languages for various scientific computing, the support of representative numerical libraries, and the support for MPI, which can be said to be the standard for parallel programs, it can be said that in addition to Fortran and C/C++, which have been widely used as programming languages for scientific computing, scientific computing includes programming languages that are widely used not only in scientific computing, but also in various IT fields and AI/ML fields such as Python, Java, MATLAB, and R. Accordingly, we believe that there is a need for a curriculum and learning on scientific computing based on this language.

In particular, languages such as PYTHON, JAVA, AND R are programming languages that are commonly used not only in scientific computing but also in IT fields such as web development, and are widely used and taught in humanities and social sciences and economics as well as science and technology according to the recent use in AI/ML and data analysis. Therefore, it is necessary to educate domain knowledge and numerical libraries based on Python, R, and JAVA, and to develop in-depth curricula such as hardware, infrastructure, and cloud computing along with low-level languages such as C/C++ based on this.