Frequently Asked Questions

What is SciPy?

SciPy is a set of open source (BSD licensed) scientific and numerical tools for Python. It currently supports special functions, integration, ordinary differential equation (ODE) solvers, gradient optimization, parallel programming tools, an expression-to-C++ compiler for fast execution, and others. A good rule of thumb is that if it's covered in a general textbook on numerical computing (for example, the well-known Numerical Recipes series), it's probably implemented in SciPy.

How much does it cost?

SciPy is freely available. It is distributed as open source software, meaning that you have complete access to the source code and can use it in any way allowed by its liberal BSD license.

On this page

What are SciPy's licensing terms?

SciPy's license is free for both commercial and non-commercial use, per the terms of the BSD license here.

How can SciPy be fast if it is written in an interpreted language like Python?

Actually, the time-critical loops are usually implemented in C, C++, or Fortran. Parts of SciPy are thin layers of code on top of the scientific routines that are freely available at https://www.netlib.org/. Netlib is a huge repository of incredibly valuable and robust scientific algorithms written in C and Fortran. It would be silly to rewrite these algorithms and would take years to debug them. SciPy uses a variety of methods to generate "wrappers" around these algorithms so that they can be used in Python. Some wrappers were generated by hand coding them in C. The rest were generated using either SWIG or f2py. Some of the newer contributions to SciPy are either written entirely or wrapped with Cython or Pythran.

A second answer is that for difficult problems, a better algorithm can make a tremendous difference in the time it takes to solve a

problem. So, using SciPy's built-in algorithms may be much faster than a simple algorithm coded in C.

I've found a bug. What do I do?

The SciPy development team works hard to make SciPy as reliable as possible, but, as in any software product, bugs do occur. If you find bugs that affect your software, please tell us by entering a ticket in the SciPy bug tracker.

How can I get involved in SciPy?

Head to our community page. We are keen for more people to help out writing code, tests, documentation, and helping out with the website.

Is there commercial support available?

Yes, commercial support is offered for SciPy by a number of companies, for example Anaconda, Enthought, and Quansight.

NumPy vs. SciPy vs. other

packages

What is the difference between NumPy and SciPy?

In an ideal world, NumPy would contain nothing but the array data type and the most basic operations: indexing, sorting, reshaping, basic elementwise functions, etc. All numerical code would reside in SciPy. However, one of NumPy's important goals is compatibility, so NumPy tries to retain all features supported by either of its predecessors. Thus, NumPy contains some linear algebra functions and Fourier transforms, even though these more properly belong in SciPy. In any case, SciPy contains more fully-featured versions of the linear algebra modules, as well as many other numerical algorithms. If you are doing scientific computing with Python, you should probably install both NumPy and SciPy. Most new features belong in SciPy rather than NumPy.

How do I make plots using SciPy?

Plotting functionality is beyond the scope of SciPy, which focus on numerical objects and algorithms. Several packages exist that integrate closely with SciPy to produce high

quality plots, such as the immensely popular Matplotlib. Other popular options are Bokeh, Plotly and Altair.

How do I make 3D plots/visualizations using SciPy?

Like 2D plotting, 3D graphics is beyond the scope of SciPy, but just as in the 2D case, packages exist that integrate with SciPy. Matplotlib provides basic 3D plotting in the `mplot3d` subpackage, whereas Mayavi provides a wide range of high-quality 3D visualization features, utilizing the powerful VTK engine.

Why both `numpy.linalg` and `scipy.linalg`? What's the difference?

`scipy.linalg` is a more complete wrapping of Fortran LAPACK using f2py.

One of the design goals of NumPy was to make it buildable without a Fortran compiler, and if you don't have LAPACK available, NumPy will use its own implementation. SciPy requires a Fortran compiler to be built, and heavily depends on wrapped Fortran code.

The `linalg` modules in NumPy and SciPy have some common functions but with different docstrings, and `scipy.linalg` contains functions not found in `numpy.linalg`, such as functions related to LU decomposition and the Schur decomposition, multiple ways of calculating the pseudoinverse, and matrix transcendentals, like the matrix logarithm. Some functions that exist in both have augmented functionality in `scipy.linalg`; for example, `scipy.linalg.eig` can take a second matrix argument for solving generalized eigenvalue problems.

Python version support

Do NumPy and SciPy still support Python 2.7?

The last version of NumPy to support Python 2.7 is NumPy 1.16.x. The last SciPy version to do so is SciPy 1.2.x. The first release of NumPy to support Python 3.x was NumPy 1.5.0. Python 3 support in SciPy was introduced in SciPy 0.9.0.

Does SciPy work with PyPy?

In general, yes. Recent improvements in PyPy

have made the scientific Python stack work with PyPy. Since much of SciPy is implemented as C extension modules, the code may not run any faster (for most cases it's significantly slower still, however, PyPy is actively working on improving this). As always when benchmarking, your experience is the best guide.

Does SciPy work with Jython or C#/.NET?

No, neither is supported. Jython never worked, because it runs on top of the Java Virtual Machine and has no way to interface with extensions written in C for the standard Python (CPython) interpreter.

Some years ago, there was an effort to make NumPy and SciPy compatible with .NET. Some users at the time reported success in using NumPy with Ironclad on 32-bit Windows. Lastly, Pyjion is a new project which reportedly could work with SciPy.

In any case, these runtime/compilers are out of scope of SciPy and not officially supported by the development team.

Where to get help

See the community page.