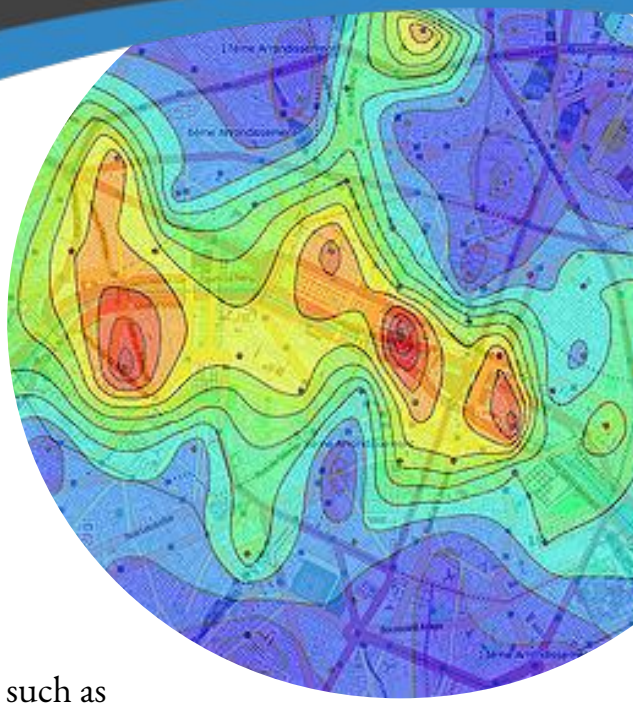
SciPy



Project Overview

SciPy is open-source software for mathematics, science, and engineering. It includes modules for statistics, optimization, integration, linear algebra, Fourier transforms, signal and image processing, ODE solvers, and more. SciPy provides many user-friendly and efficient numerical routines such as for numerical integration and optimization. SciPy runs on all popular operating systems, is easy to use, and powerful enough to be depended upon by the world's leading scientists & engineers.

Roadmap Priorities

- **Evolve BLAS and LAPACK support**
  - **Library support**
  - **Support for newer LAPACK features**
- **Implement sparse arrays in addition to sparse matrices**
- **Fourier transform enhancements**
- **Support for distributed arrays and GPU arrays**
- **Improve benchmark system for optimize**
- **Linear programming enhancements**
- **Improve source builds on Windows**

SciPy

Evolve BLAS and LAPACK support

The Python and Cython interfaces to BLAS and LAPACK in `scipy.linalg` are one of the most important things that SciPy provides. In general, `scipy.linalg` is in good shape, however a number of important improvements can be made:

1. **Library support:** Released wheels of SciPy now ship with OpenBLAS, which is currently the only feasible performant option (ATLAS is too slow, MKL cannot be the default due to licensing issues, and Accelerate support has been dropped because Apple does not update it anymore). However, OpenBLAS is not very stable and sometimes its releases cause downstream breakages for SciPy. OpenBLAS also has issues with threading (which, e.g., interfere with using SciPy with PyPy). At the very least, SciPy needs better support for debugging OpenBLAS issues as well as better documentation on how to build SciPy with OpenBLAS. One option is to use BLIS for a BLAS interface (see numpy gh-7273).
2. **Support for newer LAPACK features:** In SciPy 1.2.0, the minimum supported version of LAPACK was increased to 3.4.0. Now that SciPy has dropped Python 2.7 support, that version can be increased further (MKL + Python 2.7 was the blocker for >3.4.0 previously) and SciPy can start adding support for new features in LAPACK.

Implement sparse arrays in addition to sparse matrices

The sparse matrix formats are mostly feature-complete, however the main issue is that they act like `numpy.matrix` (and is pending future deprecation). SciPy needs sparse *arrays* which act like `numpy.ndarray`. This is being worked on in pydata/sparse. The tentative plan for integration of this package is to:

- Start depending on pydata/sparse once it's feature complete enough and has decent enough performance (pydata/sparse still needs a CSC/CSR equivalent representation).
- Add support for pydata/sparse to `scipy.sparse.linalg` (and perhaps to `scipy.sparse.csgraph` after that).
- Indicate in the documentation that for new code users should prefer pydata/sparse over sparse matrices.
- When NumPy deprecates `numpy.matrix`, SciPy will vend that code or maintain it as a stand-alone package.

Fourier transform enhancements

We want to integrate PocketFFT into scipy.fftpack for significant performance improvements (see this NumPy PR for details), add a backend system to support PyFFTW and mkl-fft, and align the function signatures of numpy.fft and scipy.fftpack.

Support for distributed arrays and GPU arrays

NumPy is splitting its API from its execution engine with the `__array_function__` and `__array_ufunc__`. This will enable parts of SciPy to accept distributed arrays (e.g. `dask.array.Array`) and GPU arrays (e.g. `cupy.ndarray`) that implement the ndarray interface. At the moment it is not yet clear which algorithms will work out of the box, and if there are significant performance gains when they do. SciPy wants to create a map of which parts of the SciPy API work, and improve support over time.

Improve benchmark system for optimize

`scipy.optimize` has an extensive set of benchmarks for accuracy and speed of the global optimizers. That has allowed adding new optimizers (e.g. `shgo` and `dual_annealing`) with significantly better performance than the existing ones. The optimize benchmark system itself is slow and hard to use, however. SciPy needs to make it faster and make it easier to compare performance of optimizers via plotting performance profiles.

Linear programming enhancements

Recently, all known issues with `optimize.linprog` have been solved. Now we have many ideas for additional functionality, such as:

- Integer constraints,
- Sparse matrix support, and
- Performance improvements

See this issue for more details.

Improve source builds on Windows

SciPy critically relies on Fortran code. This is still problematic on Windows. There are currently only two options: using Intel Fortran, or using MSVC + gfortran. The former is expensive, while the latter works (and is what is used for releases) but is quite hard to do correctly. For allowing contributors and end users to reliably build SciPy on Windows, using the Flang compiler looks like the best way forward long-term. Until Flang support materializes, we need to streamline and better document the MSVC + gfortran build.