Proposal for GSoC 2022: Scipy

Restructuring the F2PY frontend and replacing the build system

1. Contact Details

Name:	Namami Shanker
Nickname:	Richie
Country:	India (+5:30 GMT)
Email:	namami2011@gmail.com
Phone:	+91 8840273138
GitHub:	https://github.com/NamamiShanker
Personal blog:	https://namamishanker.github.io/
LinkedIn:	https://https://www.linkedin.com/in/namamishanker/

2. Self Introduction

I am Namami Shanker, an Electrical Engineering 3rd year undergraduate at the Indian Institute of Technology, Goa. I am an open-source software advocate experienced in python, Scientific python libraries and backend development.

I have work experience as a python/backend intern at DroneBase, a diagnosis and analysis company in the solar energy sector. I worked with a team of 20+ experienced engineers to develop backend APIs and test suites with Flask and Pytest. I gained experience in API development, Database engineering and writing scalable, testable and maintainable code.

I am also the lead backend engineer at IIT Goa's software development club. I have created many working web/mobile applications for various fests and events for the Institute and Clubs. I have experience in Linux sysadmin skills. I pursued OpenSUSE Cloud-Native Fundamentals Scholarship Program, through which I gained knowledge in container management, orchestration, and CI/CD skills.

2.1. Open-Source Contributions

I have been active in open-source communities since June 2021, when I first participated in GirlScript Summer of Code. I was the most significant contributor to Dynamic CLI (now a member of the organization). Since I am an engineering student, I drifted towards Scientific Python libraries. I started tracking Scipy's issues and submitting pull requests while communicating with the maintainers. I have been striving towards making meaningful contributions while learning continuously since the start of my involvement with the Scientific Python community.

The following are my contributions to the SciPy community (chronological): -

1. DEP: Remove usage of numpy.compat
2. DOC: Remove link to alpha in scipy.stats.dirichlet
3. BUG: Update chi_gen to use scipy.special.gammaln
4. BUG: Fix owens_t function when a tends to infinity
5. Fix incorrect error message in multivariate_normal
6. ENH: Add freezability to remaining multivariate distributions
7. ENH: stats: Add freezability to unitary_group
8. ENH: stats: Implement frozen random_correlation

I have started working on the F2PY module within NumPy and have contributed a documentation page: -

1. DOC: Add F2PY tests documentation

Apart from this, I have been in constant touch with maintainers regarding goals of SciPy, upcoming projects etc. I was introduced to F2PY through SciPy's idea-list for GSoC. I started studying it and discussed many of my ideas with Mr Rohit Goswami. I have written the following articles while studying F2PY: -

1. Building fast libraries: NumPy and SciPy
2. F2PY Tests

3. Project

3.1. Project Name

Restructuring the F2PY frontend and replacing the build system.

3.2. Project Description

F2PY is an open-source utility that provides an easy connection between Python and Fortran languages. Initially, F2PY developed outside the NumPy repository and was imported into NumPy around 2007. It has not had any major releases since 2009, ever since it became feature-complete for Fortran 77. Beginners browsing its codebase in the current state might find it challenging to read, understand and contribute. The frontend of F2PY is a handwritten command line parser that predates the intrinsic ‘argparse’ module (introduced in Python 3.2, 2011) by several years.

I propose modernising the CLI with the “argparse” python library, simplifying the frontend codebase and making it more developer-friendly to contribute to (discussed in sec. 3.3.1).

Since “np.distutils” is set for deprecation, I propose adding an option to build the F2PY’s generated CPython extension modules with Meson. Meson will significantly speed up building standalone extension modules through the F2PY CLI. There is no change proposed to the process of generating C and Fortran wrappers from Fortran source files. Generation of the source code for integration in Python projects will be carried out in the same way. Currently, “np.distutils” is used by F2PY to build generated extension modules. Adding a Meson backend will provide faster builds and long-term backend support for F2PY (discussed in sec. 3.3.3).

Additionally, I plan to re-implement the class-based test-suite of F2PY in a modern pytest manner and add a developer’s guide to F2PY (discussed in sec. 3.3.2).

3.3. Predesign

3.3.1. CLI Design: F2PY uses “f2py2e.py” as a CLI interface to parse user input. “f2pyarg.py” is an ongoing re-implementation of this CLI using the “argparse” module, but it needs to be completed. I intend to complete this rewrite and provide a modernised, developer-friendly CLI. Additionally, I plan to deliver developer-guide documentation and a test suite covering the CLI.

3.3.2. Test suite Design: F2PY's current test suite predates “pytest” and does not use fixtures. I shall work on refactoring the existing test suite, using pytest features such as fixtures, setups and tear-downs.

3.3.3. Backend design: F2PY currently uses “np.distutils” for compiling generated C files. Since it is to be deprecated, I intend to work on the addition of a flag that will switch the build backend to Meson for

F2Py modules.

As discussed with the NumPy developer community and noted in the documentation, the goal is to transition gradually. F2PY will support both “np.distutils” and “meson” building options for testing, after which the former can be removed completely.

Detailed back-end proposal: F2PY is primarily used for signature file generation (-h flag), extension module construction (-m flag), and module building (-c flag). The new CLI will not require extensive refactoring of files pertaining to the first two features. However, compiling and building modules (Ex - “f2py2e.py:run_compile”) will receive heavy refactoring to incorporate meson building.

I plan to create a template “meson.build.src” file that the CLI will use to generate a custom build file. The refactored “run_compile” method will generate a new “meson.build” file and invoke meson to build a shared library from the generated C extension module.

3.3.4. Technology Roadmap: I will continue to use the Python internal “argparse” module to design the CLI. “pytest” will be used to add tests for the CLI and modernise the existing test suite. Meson will be required to build C extension modules with F2PY.

1. Post GSoC:

The following are some topics I would like to continue working on after the end of GSoC: -

- **Add pyf file support to Meson.** Meson does not natively support or understand “pyf” files. Cython was added very recently. I would like to add F2PY support to meson to natively parse “pyf” files.
- **Derived types support.** Over the course of my conversations with the F2PY development team in NumPy, I felt that the lack of modern derived type support holds F2PY back. I want to get involved in adding this support and make F2PY more general purpose.

- **Improve docs, add tests and fix bugs.** I shall remain active in the Scientific Python community. This project will tremendously increase my programming skills and knowledge. I will keep maintaining F2PY and want to extend my support to Numpy and Scipy as much as I can.
- **Maintaining Scipy.** SciPy was the first open-source software to which I contributed. I learned so much from its community, and I will continue to help maintain and improve SciPy. I will continue my work on the "stats" module as detailed in my discussion with Matt Haberland.

2. Timeline

2.1. Community Bonding Period (May 20 – June 12)

- Introduce myself and this project on NumPy, SciPy and GSoC mailing list.
- Remain in constant touch with my mentors using Slack. Set up user requirements and discuss the design details with mentors.
- Discuss with mentors the implementation plan.
- Try to fix bugs to understand F2PY's source code.
- Thoroughly study Meson's documentation and understand its usage.
- Set up dev environment and my blog page for TODO list and weekly report.

2.2. Official Coding Period (June 13 – September 4)

Week 1 - 2 (June 13 – June 25)

- Complete the existing f2pyarg.py. Implement missing functionalities like compile, link etc.
- Refactor the "run_compile" method for the "*np.distutils*" backend.
- Discuss linking issues and to-be-deprecated flags with the mentor and implement solutions.

Week 3 - 4 (June 27 – July 9)

- Start working on existing test_f2py2e.py. Implement remaining tests, enhance existing failing tests and increase test coverage.
- Communicate with the mentor and maximise tests for f2py CLI.

Week 5 (July 11 – July 16)

- Add developer documentation for F2PY. Will contain files *"f2py-developer.rst"* and *"f2py-test.rst"* explaining file structure, the functionality of F2PY and its test suite, respectively.
- Start refactoring F2PY's test suite in a modern fixture-based style.

Week 6 (July 18 – July 23)

- Finish refactoring the test suite.
- Fix bugs and update documents.

*** First evaluation period (July 25 – July 29)**

- Deliver the implemented f2py CLI, implemented tests and developer guide.
- The F2PY CLI will now be more open to study and contributions by developers.

Week 8 - 9 (Aug 1 – Aug 13)

- Create a “meson.build.src” file. (see sec. 3.3.3)
- Update “run_compile” method to add meson building option.

Week 10 (Aug 15 – Aug 20)

- Fix bugs and update documentation.

Week 11 -12 (Aug 22 – Sept 4)

- Pull request for code review and merge.
- Buffer time for any unexpected delays.

*** Final evaluation period (Sept 5 - 12):**

- Deliver the working implementation of the Meson build system for F2PY.
- Wrap up the project and submit the final evaluation of my mentor.

3. Acknowledgement

I want to thank Mr Rohit Goswami for helping me iterate this proposal and for all the suggestions he provided me for this project.

4. GSoC participation

This is my first application for Google Summer of Code.