Improving usability and sustainability for NumPy and OpenBLAS

1 Goals

This proposal aims to enable the continuation of recent efforts by NumPy and OpenBLAS towards improved usability, sustainability and maintainability for both projects, and improve NumPy's integration with Fortran tools. We aim to achieve this through:

1. improvements in `numpy.f2py` code and documentation;
2. continue improving NumPy's documentation aimed at new users and contributors;
3. addressing key technical issues in OpenBLAS which regularly impact the NumPy release process.

NumPy [1] is a numerical library for Python that provides the key data structure - an N-dimensional array - and numerical algorithms on which much of the scientific Python ecosystem is built. NumPy is used in virtually every area of science and engineering, including in many biomedical tools and applications like QIIME [2], Qiita [3], MNE [4], scikit-bio [5], and GALA [6]. It is used in the data analysis pipelines of large collaborations like the Human Cell Atlas [7] and the International Brain Laboratory [8]. It is used directly by millions of scientists analyzing their own data, and is a key dependency of statistical, image analysis and visualization tools they use like Matplotlib [9], SciPy [10], Scikit-image [11], Scikit-learn [12], Pandas [13], Statsmodels [14], PyMC3 [15], Tensorflow [16], and PyTorch [17]. As a part of NumPy, the `numpy.f2py` tool [18] is used by numerous projects in the Python scientific ecosystem including SciPy itself to automate the compilation of Fortran code into Python modules, making it an important piece of the infrastructure for these projects. However, while Fortran has evolved, `numpy.f2py` does not support many of the current features of the language. In addition, due to a lack of testing and maintenance, regressions have been introduced in `numpy.f2py` code that may result in a propagation of bugs for many other projects. We propose to address the backlog of maintenance and to extend the tool as necessary to support modern use cases, in particular adding support for Fortran 90 derived types.

OpenBLAS [19] is a library that provides accelerated versions of the BLAS [20] and LAPACK [21] linear algebra algorithms. It is a key dependency of NumPy and SciPy, as well as of R and Julia. So while most users will not realize they are using it, it is essential for the three leading scientific open source programming ecosystems. Moreover, there are few alternatives for OpenBLAS today – ATLAS [22] is significantly slower and dormant, BLIS/Flame [23] isn't yet ready as a replacement, and Intel MKL cannot be used by default for licensing reasons. While the OpenBLAS repository lists 160 contributors (over a timespan of 10 years and covering several CPU architectures), the project effectively only has a single maintainer and less than ten semi-regular participants. This presents a challenge. OpenBLAS issues regularly impact the NumPy release process, and most of the NumPy maintainers do not have the specific technical expertise needed to address such issues in the OpenBLAS code base. Therefore we aim to use part of the funding requested in this proposal to improve the sustainability and reduce technical debt of OpenBLAS.

This proposal aims to provide part-time funding for Melissa Mendonça and Martin Kroeker,

as well as for Pearu Peterson. Mendonça is a Software Engineer at Quansight, currently involved with the NumPy Documentation Team and `numpy.f2py`. She has worked previously as a university professor and, as a long-time member of the Brazilian Python and open source communities, has worked in popularizing open source software in academia through teaching and outreach. Kroeker has been the lead OpenBLAS maintainer for 5 years, and runs his own software business in Germany. Peterson is the original author of the `numpy.f2py` tool and is a Senior Software Engineer at Quansight.

The complete proposed team and their responsibilities are:

- Melissa Mendonça (PI, 20 hrs/wk for 1 year); managing the documentation team for NumPy and development on NumPy and f2py;
- Martin Kroeker (10 hrs/wk for 1 year); OpenBLAS technical improvements.
- Pearu Peterson (8 hrs/wk for 1 year); development on f2py;
- Student intern (3 months); OpenBLAS technical improvements.

References

[1] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: a structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, p. 22, 2011.

[2] J. G. Caporaso, J. Kuczynski, J. Stombaugh, K. Bittinger, F. D. Bushman, E. K. Costello, N. Fierer, A. G. Pena, J. K. Goodrich, J. I. Gordon, *et al.*, "QIIME allows analysis of high-throughput community sequencing data," *Nature methods*, vol. 7, no. 5, p. 335, 2010.

[3] A. Gonzalez, J. A. Navas-Molina, T. Kosciolek, D. McDonald, Y. Vázquez-Baeza, G. Ackermann, J. DeReus, S. Janssen, A. D. Swafford, S. B. Orchanian, *et al.*, "Qiita: rapid, web-enabled microbiome meta-analysis," *Nature methods*, vol. 15, no. 10, p. 796, 2018.

[4] A. Gramfort, M. Luessi, E. Larson, D. A. Engemann, D. Strohmeier, C. Brodbeck, L. Parkkonen, and M. S. Hämäläinen, "Mne software for processing MEG and EEG data," *Neuroimage*, vol. 86, pp. 446–460, 2014.

[5] "Scikit-bio, http://scikit-bio.org." Accessed: 2019-07-26.

[6] J. Nunez-Iglesias, R. Kennedy, S. M. Plaza, A. Chakraborty, and W. T. Katz, "Graph-based active learning of agglomeration (gala): a python library to segment 2D and 3D neuroimages," *Frontiers in neuroinformatics*, vol. 8, p. 34, 2014.

[7] A. Regev, S. A. Teichmann, E. S. Lander, I. Amit, C. Benoist, E. Birney, B. Bodenmiller, P. Campbell, P. Carninci, M. Clatworthy, *et al.*, "Science forum: the human cell atlas," *Elife*, vol. 6, p. e27041, 2017.

[8] L. F. Abbott, D. E. Angelaki, M. Carandini, A. K. Churchland, Y. Dan, P. Dayan, S. Deneve, I. Fiete, S. Ganguli, K. D. Harris, *et al.*, "An international laboratory for systems and computational neuroscience," *Neuron*, vol. 96, no. 6, pp. 1213–1218, 2017.

[9] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in science & engineering*, vol. 9, no. 3, p. 90, 2007.

[10] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İlhan Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa,

P. van Mulbregt, and S. . Contributors, "SciPy 1.0 – fundamental algorithms for scientific computing in python," 2019.

[11] S. Van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu, "scikit-image: image processing in python," *PeerJ*, vol. 2, p. e453, 2014.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[13] W. McKinney *et al.*, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, vol. 445, pp. 51–56, Austin, TX, 2010.

[14] S. Seabold and J. Perktold, "Statsmodels: Econometric and statistical modeling with python," in *Proceedings of the 9th Python in Science Conference*, vol. 57, p. 61, Scipy, 2010.

[15] J. Salvatier, T. V. Wiecki, and C. Fonnesbeck, "Probabilistic programming in python using PyMC3," *PeerJ Computer Science*, vol. 2, p. e55, 2016.

[16] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283, 2016.

[17] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[18] P. Peterson, "F2py: a tool for connecting fortran and python programs," *Int. J. Computational Science and Engineering*, vol. 4, no. 4, pp. 296–305, 2009.

[19] Q. Wang, X. Zhang, Y. Zhang, and Q. Yi, "AUGEM: automatically generate high performance dense linear algebra kernels on x86 CPUs," in *SC'13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pp. 1–12, IEEE, 2013.

[20] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, "An extended set of FORTRAN basic linear algebra subprograms," *ACM Transactions on Mathematical Software (TOMS)*, vol. 14, no. 1, pp. 1–17, 1988.

[21] E. Anderson, Z. Bai, J. Dongarra, A. Greenbaum, A. McKenney, J. Du Croz, S. Hammarling, J. Demmel, C. Bischof, and D. Sorensen, "Lapack: A portable linear algebra library for high-performance computers," in *Proceedings of the 1990 ACM/IEEE conference on Supercomputing*, pp. 2–11, IEEE Computer Society Press, 1990.

[22] R. C. Whaley and J. J. Dongarra, "Automatically tuned linear algebra software," in *SC'98: Proceedings of the 1998 ACM/IEEE conference on Supercomputing*, pp. 38–38, IEEE, 1998.

[23] F. G. Van Zee and R. A. Van De Geijn, "BLIS: A framework for rapidly instantiating BLAS functionality," *ACM Transactions on Mathematical Software (TOMS)*, vol. 41, no. 3, p. 14, 2015.

[24] "F2PY G3, https://github.com/pearu/f2py." Accessed: 2020-07-23.

[25] "NumPy Team Gallery, https://numpy.org/gallery/team.html." Accessed: 2020-07-13.

[26] "Zhang Xianyi's homepage, https://xianyi.github.io." Accessed: 2019-07-26.

Form application misc sections

Short description of project (200 words maximum)

NumPy

NumPy is the fundamental package for numerical and scientific computing with Python. It provides an N-dimensional array data structure and a large set of numerical functions that operate on it. Its array computing concepts – vectorization, broadcasting, indexing, and universal functions (ufuncs) – are used throughout the scientific Python ecosystem, and have also inspired similar libraries in other programming languages. The NumPy API and concepts have been adapted to other Python libraries, such as Dask and Xarray for distributed computing and PyTorch and CuPy for GPU computing.

NumPy furthermore provides some essential algorithms for numerical applications, such as Fourier transforms, random number generators and linear algebra routines. It also provides many of the utilities and standards for the scientific Python ecosystem: documentation guidelines and the NumpyDoc package, testing tools, and f2py and numpy.distutils for building scientific packages containing compiled code.

NumPy's is used in all areas of science and engineering, and its user base is estimated to be in the range of 5,000,000 – 15,000,000.

OpenBLAS

OpenBLAS is a highly optimized library for linear algebra computations. It provides accelerated versions of BLAS (Basic Linear Algebra Subroutines and LAPACK (Linear Algebra Package), which are the two universal interfaces to perform linear algebra with. OpenBLAS provides automatic parallelization and optimized code for a large set of CPU architectures. Its performance is state-of-the-art, comparable to the leading commercial alternatives and one to two orders of magnitude faster than the reference BLAS and LAPACK implementations.

Proposal Purpose (1 sentence, max 255 chars)

We aim to improve the robustness and usability of NumPy, continuing to work in documentation and community building, modernize its integration with Fortran tools via `numpy.f2py` and ensure the sustainability of both NumPy and OpenBLAS.

Abstract/Summary (max 250 words)

NumPy is a library for Python that provides the key data structure for numerical and scientific computing with Python. It also provides f2py, a tool to automate the compilation of Fortran code into Python modules, making it an essential piece of the infrastructure for many projects in the scientific Python ecosystem.

We propose to expand the work of the NumPy Documentation Team, building a comprehensive set of high-level educational content involving NumPy, while at the same time diversifying opportunities for community contributions. In addition, we propose to address the maintenance backlog

and extend the functionality of f2py, adding support for modern Fortran features, increasing its test coverage, and improving the documentation.

Finally we will address key technical issues in OpenBLAS, which is the high-performance linear algebra library that is a critical NumPy dependency. The main topics we plan to address are issues with Thread Local Storage, performance at certain problem sizes, implementation of BLAS extensions already provided by similar libraries, and improvements to the Reference LAPACK implementation, which OpenBLAS relies on for most of its LAPACK functionality.

Short narrative biography of the applicant (max 100 words)

Melissa is a Software Engineer at Quansight, currently involved with the NumPy Documentation Team, f2py, and consulting. She has a Ph.D. in Mathematical Optimization and has worked previously as a professor at the Federal University of Santa Catarina (Brazil), where her main focus of research has been computational methods for mathematical optimization and their implementation.

Melissa is a long-time member of the Brazilian Python and open source communities and has worked in popularizing open-source software through teaching and outreach. She has served as co-chair of Diversity and Inclusion for the SciPy Conference in 2019 and 2020.

2 Work plan

2.1 Maintaining and improving `numpy.f2py`

Currently, `numpy.f2py` supports wrapping Fortran 77 and Fortran 90 code, with the exception of derived types, which were introduced in Fortran 90. We aim to extend the functionality of `numpy.f2py` by adding support for derived types and working towards full Fortran 2003 compatibility, including type-bound procedures. While there have been attempts at resolving these issues in the past [24], they were never integrated into NumPy, and at the moment this would require a larger refactor of `numpy.f2py`, which we propose to do. At the same time, features that were supported at some point have suffered regressions through a lack of testing. This may cause issues in any of the numerous packages that depend on `numpy.f2py`, and need to be addressed. Finally, the documentation for the package is outdated and should be expanded to include detailed descriptions of applications and features, with examples and a clearer guidance of which features from the Fortran standard are supported.

To improve the reliability and sustainability of `numpy.f2py`, we propose the following actions:

1. Writing a new `numpy.f2py` user guide with comprehensive examples and feature descriptions;
2. Increasing the test coverage for `numpy.f2py`;
3. Implementing support for Fortran 90 derived types;
4. Writing and implementing a NumPy Enhancement Proposal on a refactor of `numpy.f2py` that allows it to support modern Fortran features;
5. Ensure that the maintenance backlog is addressed and that packages that depend on `numpy.f2py` are supported.

Work will be performed by Melissa Mendonça and Pearu Peterson, using the standard collaboration methods of the project: GitHub, mailing list, and a weekly video call that is open to the whole community.

2.2 Documentation Team coordination

As a result of our current efforts, the NumPy Documentation Team [25] has been established and we have engaged new contributors of different experience levels and backgrounds. This is done through a bi-weekly documentation team call where, on average, nine people participate. The team has created two new tutorials, one how-to, and has worked on several improvements to the API Reference documentation on the NumPy GitHub repository. In addition, in order to promote the generation of narrative documentation for NumPy, a new repository has been created to accept contributions in the form of Jupyter Notebook documents generated by the community. We plan to continue that work through mentoring of new contributors and community engagement, aiming to diversify the content of the NumPy documentation.

To enable the continuation of this work, we propose the following:

1. Writing better *onboarding* guides for new contributors, including possible pathways for users and explicit expectations on how to become a maintainer, core developer, or member of the team.
2. Coordinating the creation of new documents focusing on educational or narrative documentation.
3. Creating at least five new pieces of documentation, including at least two aimed at experienced users. For example:
    - Best practices for using advanced indexing and vectorization
    - Integrating Fortran code in NumPy
    - Using typing with NumPy

Work will be performed by Melissa Mendonça, using the standard collaboration methods of the project: GitHub, mailing list, and a bi-weekly video call that is open to the whole community.

2.3 OpenBLAS technical improvements

Continuing the work started during the previous grant, we propose to:

1. Identify and fix any remaining bugs in the Thread Local Storage (TLS) code, in particular at high thread counts;
2. Identify and fix performance bottlenecks at small matrix sizes;
3. Implement common BLAS extensions like Hadamard product.

Furthermore, we propose to implement improvements (in particular to the performance of corner cases) in Reference-LAPACK that are needed for OpenBLAS. The development of the Reference-LAPACK appears to have become stagnant with a visible backlog of proposed improvements and experimental features that we plan on addressing.

Work will be done by Martin Kroeker and the student intern, and delivered via pull requests on GitHub. Kroeker will recruit and mentor the intern.

3 Existing support

The Berkeley Institute for Data Science has received two grants for the project "Improving NumPy for Better Data Science."

- $659,359 from the Alfred P. Sloan Foundation, Apr 2018 - Oct 2020.
- $645,020 from the Gordon and Betty Moore Foundation, Apr 2018 - Oct 2020.

The main focus of those grants is to reduce technical debt and implement architectures improvements that make NumPy easier to extend. It supports two full-time NumPy developers at Berkeley. These are the only grants NumPy has received to date.

Beginning May 2019 NumPy signed up with Tidelift. In exchange for maintenance commitments around NumPy releases and security vulnerabilities, Tidelift pays NumPy $1,000 per month until October 2020.

During 2009-2014 the creation of OpenBLAS was supported by several grants of Chinese institutional funders to the University of Chinese Academy of Sciences in Beijing, as listed on the homepage [26] of OpenBLAS's original developer Zhang Xianyi (who based OpenBLAS on GotoBLAS, which had funding before 2008).

NumPy and OpenBLAS have been awarded a joint grant of $195,000 from the Chan Zuckerberg Initiative through Cycle 1 of the Essential Open Source Software for Science program. The focus of that grant was a new website, documentation restructuring and new tutorials, governance deliverables and mentorship to grow the team for NumPy, and technical improvements (thread safety, AVX-512 vectorization and thread local storage) for OpenBLAS.

Milestones and Deliverables

The deliverables for the Cycle 1 of the EOSS grant included website and documentation work for Numpy. These deliverables are 100% and 90% complete, respectively. Since we managed to build an active Documentation Team, which meets bi-weekly in calls that include on average 9 people and has developed significant activity beyond the grant deliverables, we propose a follow-up for documentation to allow the team to capitalize on this momentum, in addition to the work proposed in `numpy.f2py` and OpenBLAS. In order to evaluate the efforts in this direction, and we propose the following metrics:

- A 75% increase in merged pull requests for `numpy.f2py`;
- An increase in the number of user-oriented documentation and educational material hosted in one of the NumPy repositories, to at least 5 new pieces of documentation (Tutorials or How-Tos);
- A decrease in OpenBLAS issues reported on the NumPy issue tracker for new NumPy releases.

`numpy.f2py` deliverables

One deliverable every three months:

1. Write a new user guide with comprehensive examples and feature descriptions.
2. Increase the test coverage to at least 80%.
3. Implement support for Fortran 90 derived types.
4. Address the maintenance backlog.

Stretch deliverables:

1. Attain 100% test coverage for f2py.
2. Engaging and onboarding a new maintainer for `numpy.f2py`

Documentation deliverables

One deliverable every three months, in order:

1. Write documents with clear guidelines on the expectations and how to meet the roles defined by the new governance structure
2. Create at least 5 new pieces of documentation, including at least two aimed at experienced users. For example:
    - Best practices for using advanced indexing and vectorization
    - Integrating Fortran code and NumPy
    - Using typing with NumPy
3. Have at least one maintainer dedicated to documentation work.

OpenBLAS deliverables

One deliverable every 3 months:

1. Identify and remove any issues with Thread Local Storage for improved performance;
2. Identify and fix performance issues at small matrix sizes;
3. Improve support for the new half-precision float format;
4. Algorithmic improvements, implementation of new BLAS extensions (GEMMT, Hadamard product)

Metrics

1. An increase in the number of high-level documents for NumPy.
2. A decrease in the time needed to review and merge pull requests related to `numpy.f2py`;
3. A decrease in OpenBLAS issues reported on the NumPy issue tracker.

Progress Report

Provide a short summary of progress towards the deliverables in your currently funded proposal (maximum of 250 words)

As part of the work developed with the support of the Cycle 1 EOSS grant, we list the following:

- The new numpy.org website has been released, with a complete redesign and new documentation content, with translations infrastructure in place;
- New website and documentation teams have been established;
- The user guide has been restructured, and a new guide on contributing to the NumPy documentation has been added;
- Two tutorials have been integrated into the NumPy documentation, with other documents in progress;
- The NumPy team has participated in the SciPy conference, presenting talks and organizing a sprint at the event;
- There has been considerable work done towards improving thread safety and support for the 512-bit Advanced Vector Extensions (AVX-512) code in OpenBLAS. Also, OpenBLAS now passes the entire SciPy test suite on both x86_64 and PowerPC architectures. The increased activity has also attracted several new contributors.

Landscape Analysis

Briefly describe the other software tools (either proprietary or open source) that the audience for this proposal is primarily using. How do the software projects in this proposal compare to these other tools in terms of size of user base, usage, and maturity? How do existing tools and the project(s) in this proposal interact? (maximum of 250 words)

In the scientific Python ecosystem, NumPy is the standard library for array computation and numerical functions. Many open-source packages used in scientific computing, machine learning and data science, such as Jax (https://jax.readthedocs.io/), xarray (http://xarray.pydata.org), PyTorch (https://pytorch.org/), Dask (https://dask.org/), TensorFlow (https://www.tensorflow.org/) and SciPy itself (https://scipy.org), to cite a few, have NumPy as a dependency. These packages and libraries are used by industry and academia in many contexts and applications, which makes NumPy - and OpenBLAS as a dependency of NumPy - unique in its position as a core library for the entire ecosystem. While other open-source programming languages, such as Julia, C/C++, or Fortran, are also used in the scientific computing space, each of these has its own focus and scope. In fact, they are often used in combination, as highlighted by this proposal's focus on the Python/Fortran integration provided by *numpy.f2py*. This allows a rich and broad set of applications to be addressed. Finally, one could consider closed-source solutions such as MATLAB in scientific computing, especially as it has been widely used in academia for many years. However, as the open science movement has been encouraging researchers and even the industry to improve openness and transparency in the scientific software world, open-source software solutions have been preferred in recent times.