782
doi:10.1017/S1431927613005904
Microsc. Microanal. 19 (Suppl 2), 2013

Open-Source Python Scripting and Analysis with Nion Swift

Matthew F. Murfitt[1], Christopher E. Meyer[1], Gwyn Skone[1], Niklas Dellby[1] & Ondrej L. Krivanek[1]

[1]Nion Co., 1102 8th St, Kirkland, WA 98033, USA

The amount of information produced by ever-improving scientific instruments continues to grow exponentially [1]. Advanced software plays a central role in processing the information, and it is important to ensure that its results can be independently verified and reproduced. This is only possible if the experimental and computational procedures are fully described and documented [2].

Presently, closed-source commercial applications are often used for both acquiring and processing scientific data. Standard examples are Gatan's DigitalMicrograph™ (DM), which is capable of both acquiring and processing data, and MATLAB™ and Mathematica™, which are used for post-acquisition processing. Although DM offers the benefit of simultaneous acquisition and analysis, it lacks the extensive mathematical libraries and processing options offered by the other two applications. The applications are often easy to use and provide a polished user interface, but they fail to expose internal code that can be checked for correctness. Additionally, they often need expensive licenses to run, and they may not be available on a wide range of platforms.

Fortunately, alternatives exist, such as Python, which is an open-source scripting language with a large active community [3]. The Python source code is freely available for multiple platforms and offers a vast library of advanced routines (over 19,000 currently, although not all actively maintained), including numerical analysis, data visualization, a rich selection of image processing routines used in astronomy and scanning probe microscopy, and even molecular biology [4]. Python also allows users to create native compiled modules through a C API (application programming interface).

Bearing these factors in mind, we have developed Nion Swift: software for microscope control and data acquisition, processing and analysis. Swift combines an intelligent, high-level control of Nion electron microscopes with an attractive, easy-to-use user interface (UI), and advanced data processing. It includes functions such as efficient autotuning and multi-dimensional data handling, 64-bit addressing and multi-platform support. It will be described more fully elsewhere.

As one of its key features, Swift allows the acquired data to be processed and analyzed in an embedded Python environment. Using Swift means that the processing code can be published and scrutinized in a fully documented and verifiable way, an option that does not exist with closed-source alternatives. We intend to make Swift freely available for non-commercial off-line use, in a form that can run on both PCs and Macs. The Python processing routines are supported on an even wider variety of platforms.

As an example of the benefits that Swift can offer, consider the case of needing to implement a novel principal-component analysis (PCA) algorithm for analyzing EELS data. DM on its own does not provide a full-featured math library to rely on. Combining DM with Mathematica or MATLAB would require users to learn two programs (and potentially two scripting languages), and having to worry about transferring data between them. Moreover, the finished project would only be available to users who have the licenses to run the two programs.

Swift offers an alternative: developing algorithms that combine a graphical user interface provided by

Swift with processing provided by Python, through the open-source Python SciPy module [5]. If more speed is needed, native C plugins can be written and imported live without having to restart or lose any data. Once a solution in Swift is created, it can be run freely without a license, with minimized training. At the same time, proprietary plugins developed by users who do not want to distribute their source code are also supported.

As a practical example, Figure 1 shows a STEM image captured by Swift and its live Fourier transform, plus the phase surface of the sample-level electron wavefront and the corresponding probe shape, both calculated using open-source Python routines.

In summary, Swift provides a user-friendly environment for the acquisition and standard analysis of scientific data, plus a powerful framework for developing new data processing and analysis procedures. It gives access to a wide range of existing open-source mathematical and scientific routines, and a standard method of writing native C plugins. By creating an environment well suited to exploring, testing and sharing different processing solutions in an open and reproducible manner, Swift may well cause a paradigm shift in the way scientific data is treated in electron microscopy[6].

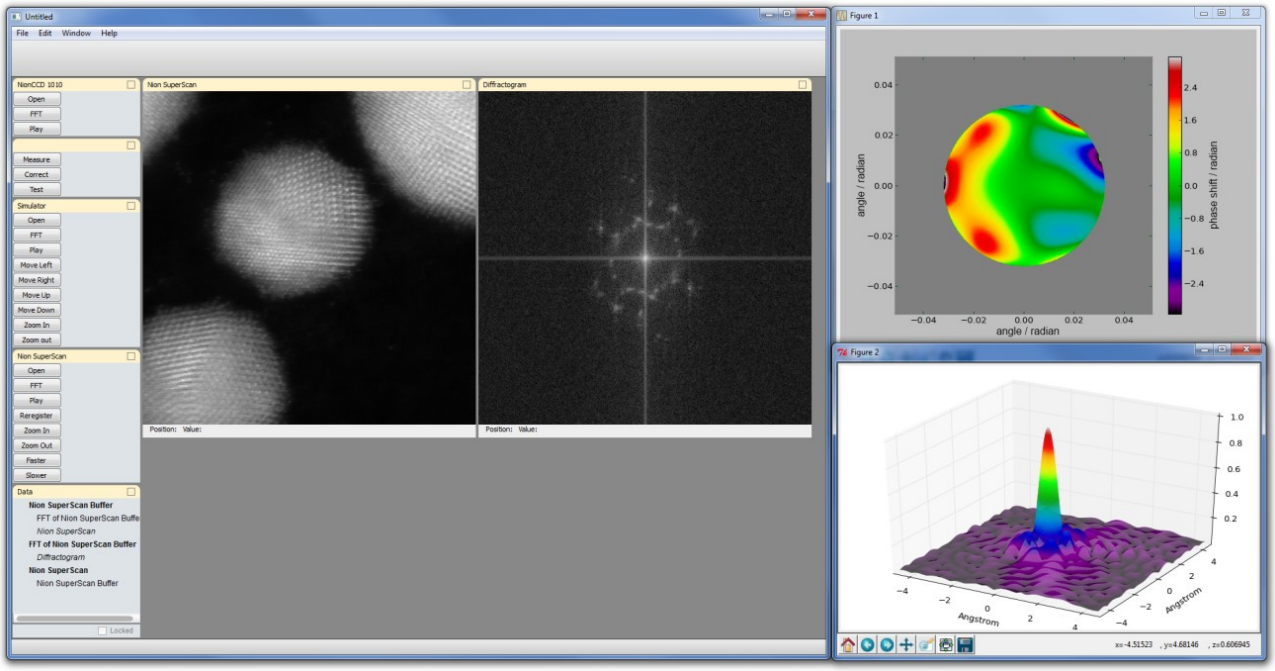

Figure 1. A screenshot of Swift. The window on the left shows a STEM image and a live diffractogram. The right side shows the electron wavefront phase variations across the objective aperture, calculated from measured aberrations, and the estimated probe shape.

[1] As a practical example, spectrum-image data sets >4 GB are now common. See also Gray, J. *et al*., "Scientific data management in the coming decade", ACM SIGMOD Record 4, **34**. pp 34-41 (2005).
[2] Donoho, D.L. *et al.*, "Reproducible research in computational harmonic analysis", Computing in Science & Engineering 1, **11** (2009)
[3] http://www.python.org
[4] http://pypi.python.org/pypi
[5] http://www.scipy.org/
[6] More information available at http://www.nion.com/swift

https://doi.org/10.1017/S1431927613005904 Published online by Cambridge University Press