Generic Mapping Tools: Improved Version Released

Generic Mapping Tools (GMT) is an open-source software package for the analysis and display of geoscience data, helping scientists to analyze, interpolate, filter, manipulate, project, and plot time series and gridded data sets. The GMT toolbox includes about 80 core and 40 supplemental program modules sharing a common set of command options, file structures, and documentation. Its power to process data and produce publication-quality graphic presentations has made it vital to a large scientific community that now includes more than 25,000 individual users. GMT’s website (http://gmt.soest.hawaii.edu/) exceeds 20,000 visits per month, and server logs show roughly 2000 monthly downloads.

Since 1993, the U.S. National Science Foundation (NSF) Directorate for Geosciences has supported the development, maintenance, documentation, and distribution of GMT. On 5 November, the GMT developers are releasing GMT version 5, the first major new release in 9 years. Although GMT 5 brings many new functionalities to scientific computing and visualization, the key change is the repositioning of GMT as an application programming interface (API). The API, a new software library that users can access (Figure 1a), will allow GMT developers to serve general users better and will provide software developers direct access to GMT’s high-level functionality from their own programs.

GMT 5’s Application Programming Interface

In prior versions, the bulk of GMT functionality—such as the ability to grid data and make plots—was coded directly in individual programs. The GMT 4 library offered access only to low-level functions. From these, high-level GMT programs, which could do complex things like interpolation and contouring, were built. Unfortunately, this design prevented developers from leveraging GMT functionality from within other programming environments because access to GMT could be achieved only via system calls or low-level library functions. Consequently, calls to GMT from within another environment—for example, from MATLAB or custom C programs—usually required use of temporary files to handle input and output (I/O). The old design also prevented GMT itself from taking advantage of its own functions directly. For instance, the tool `pslegend` needed to make extensive use of system calls to `psxy` and `pstext` to plot the lines, symbols, and text that make up a map legend, making it an awkward program to maintain, especially across platforms with different conventions for I/O redirection and system calls. Furthermore, relying on external system calls made the use of GMT other than from shell scripts cumbersome, slow, and a security risk. Attempting to program custom functionality (e.g., extend GMT capabilities) via an undocumented low-level GMT 4 library was tedious and required multiple iterations because the library was subject to frequent change.

In GMT 5, each executable program has been redesigned to be a self-contained program module in a new C/C++ high-level GMT API. In addition, the I/O routines for the five standard GMT objects (grid, image, data table, text table, and color palette table) have been standardized across all modules. These high-level functions are now accessible to users via the API, and for the first time, this new API is fully documented. These changes will allow for the development of custom modules, such as performing discipline-specific analysis outside the scope of a “generic” tool set.

For the API to be as flexible as possible, the GMT 5 developers generalized the notion of I/O. In addition to the familiar mechanism of passing file names or using standard I/O, users can specify the source of input data and/or the location of output data in several different ways, including data already loaded into memory from an application invoked previously, file pointers to open files or standard streams, and file descriptors.

Thus, for a typical table of data, the GMT API takes care of the task of assembling any combination of files, pointers, and memory locations into a single virtual data set from which the GMT module may read all records at once into memory or read one record at a time. Likewise, GMT’s program modules may write their output to a virtual destination, which might be a memory location in the user’s application, a data stream, a file descriptor, or an output file. The GMT modules themselves are unaware of these details and simply read from a “source” and write to a “destination.” This design provides an easy access mechanism from within interpreted languages, such as Julia, that know how to call functions and exchange data directly from shared libraries.

Executable Programs in GMT 5

The separate executable programs—for example, gridding, imaging, and contouring commands—from version 4 no longer exist. Instead, GMT 5 builds a single executable, called `gmt`. Its first argument is the name of the module of interest. For example, `gmt pscoast` would launch the `pscoast` module, which draws coastlines on maps. This single-executable approach is the recommended way of using GMT 5 (it solves namespace problems for package managers because several GMT tools have generic names that clash with those of other packages, e.g., `triangulate`, `surface`, and `pстext`). Examples and documentation present the new syntax.

Consequently, GMT modules can now be accessed in several ways. For compatibility with existing scripts, users can choose a “classic” mode that will also honor the names of the previous executables (e.g., `blockmean` and `pscoast`). These legacy commands are links that point to the single executable `gmt`, and the link name tells `gmt` which module to launch. Because symbolic links are not supported on legacy Windows operating systems that are still widely in use, such as XP, classic executables for each of the old-style modules are provided instead.

C/C++ coders seeking to manipulate data in specific ways can call any module via the API function `GMT_Call_Module` and may develop complex custom programs that use GMT modules as building blocks. In addition to the approximately 80 core modules and about 40 supplemental modules, users may also use the API to read and write GMT objects, process command line options,
inquire about default parameters, and carry out other common tasks.
All modules are called dynamically and loaded on demand from shared libraries. The main GMT (core) modules are now decoupled from the supplement modules, the latter being found in a separate and optional plug-in group. This division simplifies maintenance, eliminates the need to distribute supplements to those who do not require them, and keeps the core truly generic.

Users may write applications or design their own plug-in modules (stored in one or more extensions) that may be loaded by the main gmt executable in exactly the same manner as the official supplemental modules. They can thus leverage the GMT framework for I/O, default and option processing, optimal fast Fourier transforms, generic data processing, and plotting.

In addition, developers have built a prototype MATLAB/Octave API (to be released later), which allows access to GMT modules from within those environments. MATLAB/Octave API calls pass arguments to a GMT module via strings that resemble those used to run gmt on the command line. In this API, however, primary input and output file names may be omitted, in which case data derive from (or will be written to) MATLAB/Octave variables (Figure 1b). A Python API is also being considered for release in the future.

Finally, the GMT API also contains three low-level functions to support FORTRAN. These allow reading and writing of any GMT-supported grid format from legacy FORTRAN programs still widely used within the sciences and include the option to transpose between C-style row-major order arrays and FORTRAN column-major order arrays.

**Improved, Customizable Operations**

Apart from the API, GMT 5 also has several other improvements. These include new modules (11 in the core for miscellaneous vector and geographical operations, 5 via a new potential field supplement, and more that are being planned), numerous improvements to existing tools (almost all modules have new options for enhanced capabilities), advanced access to several fast Fourier transform libraries, increased grid read/write performance, enabled “wildcard” support in data table selections, better integration with geographical information systems, and interactive program documentation and a gallery of examples.

In addition, the new version has 40 revised core examples, a unified development environment, better uniformity of GMT options and parameter names while retaining optional backward compatibility with version 4, extended feature testing via more than 400 test scripts covering all programs, and a formal process for reporting bugs via a new wiki with tracking of bug reports and feature requests. The top-level GMT documentation discusses all improvements (more than 200), including several not listed here, in detail; see http://gmt.soest.hawaii.edu/.

With all these changes, GMT 5 is positioned to enable rapid development of custom applications and to provide dramatic speed-up of scripted workflows in MATLAB and Octave while retaining legacy support.

The GMT 5 source code, examples, user and developer documentation, and Windows and OS X installers can be obtained from the GMT website. Official installation packages will soon be added to third-party Mac OS X distributions (e.g., Fink, MacPorts, and Homebrew distributions), and over time, package managers will likely add GMT 5 to various software distributions for the Linux and Solaris operating systems as well.

Please use the issue tracking features via the GMT website (rather than email) to report any issues encountered in the new (or previous) release. As during earlier transitions, GMT developers will continue to maintain GMT 4 until GMT 6 is introduced sometime in the future.

**Acknowledgments**

We thank the U.S. National Science Foundation for ongoing support. Paul Wessel gratefully acknowledges a Green Scholar Fellowship from the Institute of Geophysics and Planetary Physics at the University of California, San Diego’s Scripps Institution of Oceanography. GMT 5 developers also acknowledge the extensive help from all contributors of code, bug fixes, and bug reports. We thank editors and reviewers for helpful suggestions. The views expressed here are solely those of the authors and do not necessarily constitute a statement of policy, decision, or position on behalf of the National Oceanic and Atmospheric Administration or the U.S. government. This is SOEST contribution 9023.

—Paul Wessel, School of Ocean and Earth Science and Technology, University of Hawaii at Mānoa, Honolulu; email: pwessel@hawaii.edu; Walter H. F. Smith, Laboratory for Satellite Altimetry, National Oceanic and Atmospheric Administration, College Park, Md.; Remko Scharroo, European Organisation for the Exploitation of Meteorological Satellites, Darmstadt, Germany; Joaquim Lujis, Instituto Do Algarve, Faro, Portugal; and Florian Wörbe, Alfred Wegener Institute, Bremerhaven, Germany