B. Summary: Support for the Generic Mapping Tools

The Generic Mapping Tools (GMT) has served marine scientists for almost 25 years. Thousands of federally funded US scientists have been trained to use GMT, count on GMT for their day-to-day research productivity, and expect rapid responses to their inquiries about bugs, updates, new features, installation, or general usage. GMT's power to process data and produce publication-quality graphics has made it a *de facto* standard for a large segment of geoscientists, often in combination with other open-source (e.g., Python) or commercial (e.g., MATLAB®) tools. GMT is a UNIX command-line tool set but developers can now design new tools using the GMT5 C/C++ Application Program Interface (API). Two such prototypes are the GMT/MATLAB API (funded by NSF/OCE/MGG) and an in-development GMT/Python API (funded by NSF/EAR/Geoinformatics). The intertwined use of GMT, MATLAB® and Python encapsulates the working environment of a large group of marine geoscientists. A cost/benefit analysis of the GMT enterprise will reveal that it is very frugal compared to other comparable national and international geoscience software projects. For instance, the Australia-US-Norway collaborative GPlates project is ~10 years old and has so far cost Australian tax payers A$2 million, with comparable costs among their US and Norwegian partners. In contrast, GMT has cost less than US $1 million spread over 25 years and our customer base is almost a magnitude larger. Furthermore, international volunteers contribute an inordinate amount of time and expertise to improve GMT at no salary cost to NSF. The GMT team provides their vast user-base with numerous services: The gmt executable (and all its ~120 modules), the C/C++ API library for customization, the MATLAB® (and soon) Python APIs, bug and new feature reporting and tracking capabilities, access to user forums, and a complete and ever-expanding documentation. However, as the fourth decade of GMT is approached the team must begin to plan for the succession, i.e., the transition of GMT maintenance from being a PI-centric undertaking to a community-driven infrastructure.

Seven outstanding tasks are proposed that will simplify and stabilize GMT, help make it less dependent on the current PI, and better position the GMT enterprise for the inevitable transition:

- Task 1: Succession planning. Two steps are proposed for a seamless transition: a) Constitute a GMT Steering Committee to oversee GMT activities and to guide the succession planning, and b) assemble a GMT Maintainer's Handbook to guide current and future maintainers of all tasks involved in maintaining GMT.
- Task 2: PROJ4 dependency. Replace GMT's native but limited map projection library with the industry-standard PROJ.4 library for improved interoperability, access to additional map projections within GMT, and to reduce the GMT team's maintenance burden.
- Task 3: OGR Bridge. Like the GDAL "bridge", the GDAL's OGR library will be leveraged to read and write any file format for discrete geometries, similar to the existing implementation of the GDAL bridge to read any grid file format in GMT 5.
- Task 4: MGG focus. Add specific tools requested by the MGG community, such as more potential field and isostasy tools, as well as improve the documentation for the existing MGG tools.
- Task 5: GSHHG 3. Derive the next-generation GSHHG from OpenStreetMap (or similar) to avoid having to maintain a separate coastline database for which no time or long-term funding is available.
- Task 6: Code hardening. Dramatically increase the GMT test suite from the current ~500 scripts to several thousands by actively soliciting working examples from users, and continue the optimization by adding OpenMP support for multi-core processors to speed up intensive algorithms.
- Task 7: *PostScript* Light. Separate the *PostScript* low-level plotting library PSL from the GMT repository in order to simplify the GMT code base for future maintainers.

Intellectual Merit: GMT must be positioned for a future transition from a PI-centric to a community-centric business and operating model. The proposed planning is required to ensure continuation of GMT services, and given the complexity that planning must start now. Core enhancements to GMT must also start now due to their long time horizon of complete implementation given the reliance on international volunteers. Broader Impact: GMT serves thousands of US scientists daily and while its "home base" is marine geology and geophysics it has an impact across the entire Geoscience directorate.

TABLE OF CONTENTS

For font size and page formatting specifications, see GPG section II.B.2.

	Total No. of Pages	Page No.* (Optional)*
Cover Sheet for Proposal to the National Science Foundation		
Project Summary (not to exceed 1 page)	1	
Table of Contents	1	
Project Description (Including Results from Prior NSF Support) (not to exceed 15 pages) **(Exceed only if allowed by a specific program announcement/solicitation or if approved in advance by the appropriate NSF Assistant Director or designee)**	15	
References Cited	2	
Biographical Sketches (Not to exceed 2 pages each)	2	
Budget (Plus up to 3 pages of budget justification)	9	
Current and Pending Support	1	
Facilities, Equipment and Other Resources	1	
Special Information/Supplementary Documents (Data Management Plan, Mentoring Plan and Other Supplementary Documents)	1	
Appendix (List below.) **(Include only if allowed by a specific program announcement/ solicitation or if approved in advance by the appropriate NSF Assistant Director or designee)**		
Appendix Items:		

*Proposers may select any numbering mechanism for the proposal. The entire proposal however, must be paginated. Complete both columns only if the proposal is numbered consecutively.

D. Project Description: Support for the Generic Mapping Tools.

Preamble: This is very likely the last GMT maintenance proposal we will submit. After 25 years of development and maintenance using a PI-centric model with help from numerous volunteers, we propose to transition the GMT enterprise to a loosely connected federation of interested parties, guided by a GMT Steering Committee. Considerable work will be required to prepare this succession and to position GMT for this future, hence this proposal.

1. Introduction

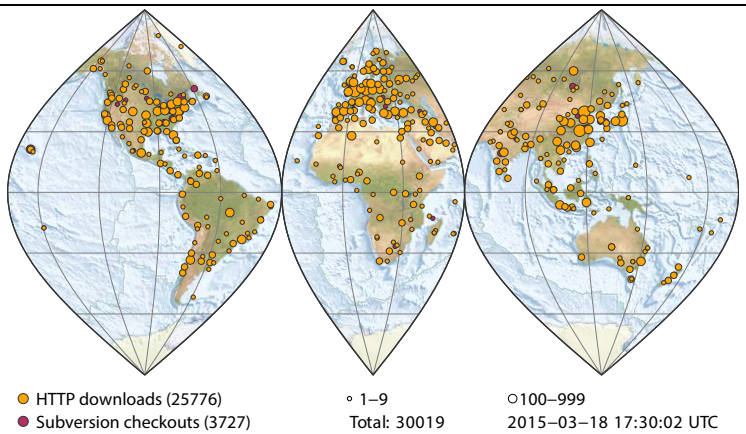

Fig. 1. *Graphical download statistics of GMT 5.1.1 for ~1 year (since its release on March 15, 2014 until March, 2015).* Most users prefer to install point-releases as only ~13% install from the active subversion repository. Separately, there have been ~20,000 additional downloads from the SOEST ftp server. These numbers equate to ~140 downloads every single day during the last year.

1.1 Overview of GMT

The Generic Mapping Tools (GMT; gmt.soest.hawaii.edu) is a widely used software infrastructure tool set for analyzing and displaying geoscience data [*Wessel and Smith*, 1991; 1995; 1998]. It has been NSF-supported since 1993 yet leverages considerable support from its national and international user community in terms of voluntary programming contributions [3 of the 5 core team members are presently located in Europe]. Its power to analyze and process data and produce publication-quality graphics has made it one of several standard processing toolsets used by a large segment of the Earth and Ocean Sciences. GMT's strengths lie in superior publication-quality vector graphics (from page-size to wall-size), geodetic-quality map projections, robust data processing algorithms scalable to enormous data sets and grid sizes (e.g., used to make global bathymetry maps from 440 million randomly-distributed soundings constraining grids having sizes up to 43,200 by 86,400 [*Olson et al.*, 2014]), and ability to run under all common desktop and laptop operating systems; furthermore, it is Open Source and freely available under a flexible license (Lesser GNU Public License). Consequently, GMT permeates work published in Earth science journals such as *Nature Geoscience*, the *Journal of Geophysical Research* and *Geophysical Journal International*, and citations of our GMT *EOS Trans. AGU* publications are closing in on 10,000. The GMT tool chest offers over 120 modules sharing a common set of command options, file structures, and documentation. GMT modules are designed as UNIX filters, i.e., they accept input and write output, and this design allows users to write scripts in which one module's output becomes another module's input, creating highly customized GMT workflows. This unlimited flexibility is one of the reasons why GMT is so broadly used: Figure 1 shows ~30,000 webserver downloads over a one-year

period following the release of version 5.1.1 (in addition there were ~20,000 ftp downloads), although many users yet to upgrade from GMT 4 are not represented (we are still seeing hundreds of downloads of GMT 4.5.13 per month as users are slow to upgrade). About 13% of all downloads are from US-based institutions; we consider these our core constituents as they predominantly are scientists at universities within the US. Current version is now 5.1.2, released during the spring of 2015.

While we are able to demonstrate both strong growth (Fig. 2) and a large user base, this task may become more complicated in the future. Already, unknown numbers of OS X users install GMT4 or GMT5 directly from one of the OS X package managers fink, port, or brew as clearly commands like "sudo port install gmt5" is a lot simpler than following a somewhat lengthy installation procedure. Likewise, the leading Linux distributions such as Ubuntu and Fedora have had GMT4 packages for a long time and as we speak both are in the process of preparing to offer GMT5 packages. Thus, while Figures 1 and 2 are not exactly showing us the tip of an iceberg yet, the popularity of GMT and its increased availability from other sources than those we directly control will eventually undermine the easily accessible user numbers and render them minimum estimates. During the 2010–2015 project period the GMT EOS publications have been officially cited ~3,000 times.

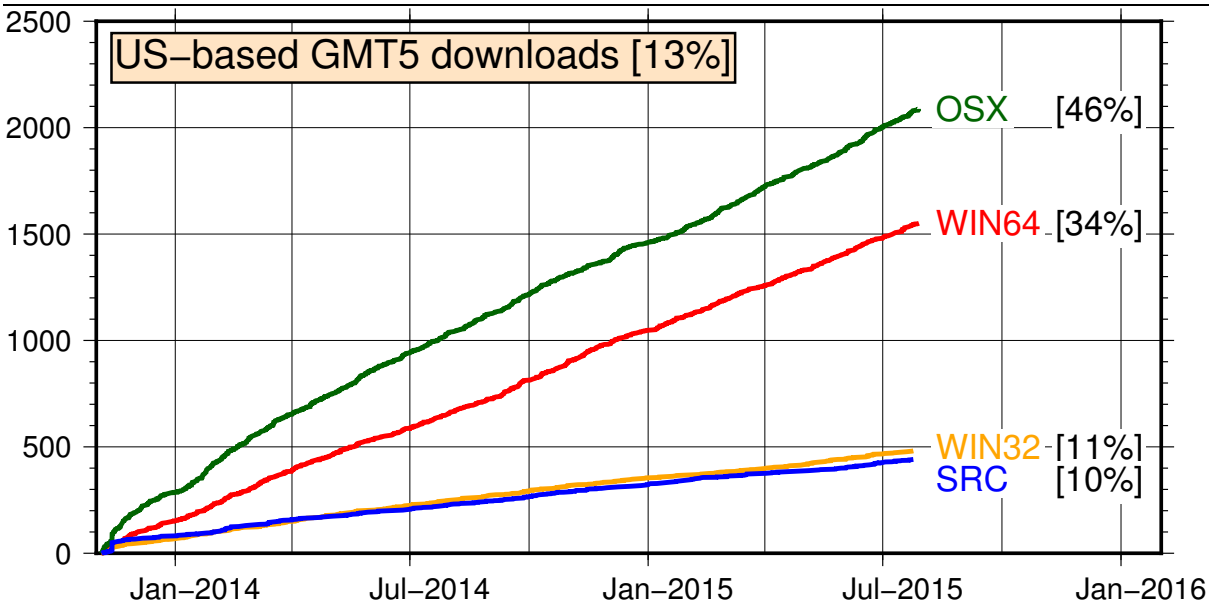

Fig. 2. *Time-series of US-only downloads of GMT5*. Cumulative download counts just from the SOEST webserver since the Nov. 5, 2013 release, restricted to show only US users broken down by platform (OSX 45%, Windows 45%, Linux 10%). We have a steady adaption rate with potential for strong growth as many have yet to migrate from GMT 4. The non-US numbers reflect 87% of users worldwide. Non-US platform proportions change, with Windows 67%, OSX 23%, and Linux at 10%. However, we caution that the SRC and OS X numbers may be artificially low since installs from OSX and Linux package managers are not included in the totals. The OS X numbers refer to the ready-to-use bundle that comes with all prerequisites bundled in (i.e., netCDF, GDAL, PCRE, and their prerequisites as well.)

1.2. The GMT Team

This team consists of the developers of the GMT infrastructure. Unlike many other software activities, the GMT team is largely composed of Earth and Ocean scientists with long-standing interests in data analysis and processing, algorithm development, and their implementation. PI Wessel (U. of Hawaii) is a marine geophysicist specializing in plate tectonics and scientific software development. He

has been the leader of this team since our first NSF grant in 1993, is largely responsible for most of the code base and manages the GMT project. The core team currently consists of four other volunteer developers: Walter Smith (Laboratory for Satellite Altimetry, NOAA) is a geophysicist specializing in satellite altimetry, marine geodesy and numerical algorithms, Remko Scharroo (EUMETSAT, Germany) is trained as a geodesist specializing in satellite orbits and remote sensing, Joaquim Luis (U. of Algarve, Portugal) is also a marine geophysicist involved in plate tectonics and is the author of Mirone [*Luis*, 2007], and Florian Wobbe (U. of Bremen, Germany) rounds out the team as a geoscientist with background both in geoinformatics and geophysics. In addition, we maintain contact with numerous other individuals via our website and accept submissions of code patches, benefit from extensive external testing, receive kernels of ideas for new GMT examples, features, or modules, and are provided with general improvements to our documentation by a wide range of contributors from around the globe.

1.3 The GMT Business Model

Unlike most NSF projects (we believe), the GMT enterprise relies heavily on volunteers from the global user community. We leverage the skills and dedication of accomplished scientists and developers from around the world in order to produce and maintain a high-quality, high-demand product in a relative short time, greatly magnifying the impact of the funding we receive from NSF. We have been very successful using this business model for over 20 years. While it is essential that NSF funding predominantly benefit US scientists, it is worth pointing out the benefit of having a huge global user base and why NSF should care about and appreciate this fact. This large user base means GMT is much more frequently exercised than it would be if restricted to US scientists and we in turn benefit by (a) a larger number of bug reports – allowing us to fix problems faster, (b) a much more nuanced set of feature requests – making GMT more relevant to a broader user base, and (c) an ability to recruit volunteers from a large group to help improve GMT. This labor force comes at no salary cost to NSF and US taxpayers and their contributions represent a large component of the work being performed. Specifically, of the five major GMT developers, three of the most active participants are European scientists [and the fourth is at a US Government lab and hence his time and the support of his IT team come free to NSF]. Only PI Wessel has salary funding from NSF but as Professor he is mostly supported by the State of Hawaii.

A cost/benefit analysis of the GMT enterprise reveals that we are very frugal compared to other comparable national and international infrastructure projects. For instance, the innovative Australia-US-Norway collaborative GPlates project for plate kinematic reconstructions [www.gplates.org] is ~10 years old and is estimated to have cost about $4 million (D. Müller, 2015, pers. comm.). In contrast, GMT has a cumulative price tag of ~$900k over 25 years, yielding an annualized cost ratio between these two projects of 10:1. In addition, our customer base is almost a magnitude larger. Another perhaps more relevant comparison is with MB-system, a vital MGG software package for processing and display of swath mapping sonar data [*Caress and Chayes*, 1996]. MB-System has a similar time history to GMT and a cumulative cost of ~$1.6 million, it includes GMT as one of its prerequisites for installation, and as specialized software it has a much smaller user base than GMT. Consequently, we believe we have been and will continue to be a solid and smart infrastructure investment for NSF in general and for OCE/MGG in particular.

The “guided volunteer” model has worked well for GMT but we do not think it necessarily would work as well for all projects. One consequence of this model is that it imparts more uncertainty to the actual timelines stipulated for product development. This is complicated further by disruptive developments such as OS upgrades, critical user feedback, and discovery of serious bugs, all leading to unplanned tasks of various magnitudes. Yet on the whole, we have been able to provide our vast user-base with numerous services: The gmt executable (and all its >100 modules), the C/C++ API library for customization, the MATLAB (and soon) Python APIs, bug and new feature reporting and tracking capabilities on our wiki, access to user forums, and a complete and ever-expanding documentation.

2. Accomplishments during last project period [2010–2015]

2.1 Major revisions

Since its initiation, GMT has been a UNIX command-line tool set. However, the release of GMT 5 [*Wessel et al.*, 2013] introduced three key changes:

- We provided a fully documented C Application Program Interface (API) for building new tools and libraries, containing basic functionality for handling GMT data objects (i.e., the input/output of data), manipulating module options, reporting of errors and warnings, and accessing any of the ~120 modules via a flexible GMT_Call_Module function. Unlike the situation in the previous GMT 4 version, these modules are no longer stand-alone UNIX programs but are high-level API functions. This means, for example, that a scientist writing custom C/C++ or Fortran code can call high-level GMT functions and quickly develop GMT-like custom programs such as new discipline-specific supplements.
- We addressed issues with "namespace pollution", i.e., how to manage lots of different program executables in a standard installation directory (such as /usr/bin) when there might be many other tools with exactly the same names. For instance, there are already other software packages that distribute generic modules called "surface" or "triangulate" and that means they are competing for installation in the same binary directory as gmt via standard software package managers. By building just a single executable called **gmt** (that can access all modules) and requiring users to run "gmt triangulate" we avoid this conflict, while a few shell functions implement backwards compatibility with older GMT 4 scripts.
- We generalized the notion of input sources and output destinations. In addition to the familiar mechanism of passing file names or using standard input/output, developers using the API can specify the sources of input data and/or the destinations of output data in several different ways, including memory locations, file pointers to open files or standard streams, and file descriptors. The GMT modules themselves are unaware of these distinctions since this flexibility is implemented in the API input/output functions.

Consequently, the new GMT API has made it possible for developers to build additional tools or external APIs on top of the core C API, allowing for rapid development of new and complex functionality, perhaps discipline-specific processing not presently available in the main GMT tool chest. An example of such development is the GMT5SAR supplement for radar interferometry processing, being developed by a team lead by David Sandwell [*Sandwell et al.*, 2011].

At present, there are six such developments (some funded, some supported by volunteers) that are building new infrastructure on top of the GMT C API:

1. The GMT/MATLAB API (supported by our NSF/OCE/MGG grant with supplement that expired this summer) has been released to the community for beta testing (via gmt.soest.hawaii.edu/projects/gmt-matlab-api) and targets users of MATLAB who want access to GMT modules while working in the MATLAB environment. It is important to note that while GMT 4 had very basic support for MATLAB going back more than a decade, this limited support only included the ability to read and write GMT grids from the MATLAB command line; no access was provided to the GMT modules themselves except via cumbersome system calls and separate input/output processing. We will discuss the GMT/MATLAB API in much more details later in this proposal.
2. A prototype GMT/Python API is presently being designed (separately funded by NSF/EAR/Geoinformatics), with activity and availability to be reported via a similar project subpage on the main GMT site (gmt.soest.hawaii.edu/projects/gmt-python-api). We will design this API using lessons learned from the GMT/MATLAB API development as well as an experimental GMT/Julia API (below), but first we will interact with the GMT/Python community

in shaping the design of the user interface to ensure the broadest possible functionality and standardization.

3. An experimental GMT/Julia API is being developed in parallel with the GMT/MATLAB interface; it uses the same concepts and thus benefits from the experience gained while developing the GMT/MATLAB API. Julia is a young, high-performance dynamic programming language for technical computing (julialang.org) that shares a great deal of syntax with the MATLAB language. In its current form the Julia API is almost a literal translation of the GMT/MATLAB API (which is C code) written in the Julia language. Besides the functions written to provide the same functionality as the other two APIs, numerous Julia wrapper functions exist that provide access to many other functions of the GMT API library. In principle, these can be used to write new GMT modules in Julia by taking advantage of the entire GMT infrastructure (In the remainder of this proposal most of what is referred to regarding the GMT/MATLAB API applies nearly one-to-one to the GMT/Julia API.) This work is an unfunded project headed by international collaborator J. Luis.
4. The high-level GMT C API was designed so that external software developers could leverage the core functionalities of GMT, such as table and grid input/output and data processing, including fast Fourier transforms, in order to build new tools rapidly. The most recent 5.5 release of MB-System now links with the GMT5 API. As part of a separate Geoinformatics grant, co-PI David Sandwell (Scripps, UCSD) is leading an effort to redesign the GMTSAR infrastructure for processing of Synthetic Aperture Radar Interferometry (InSAR) imagery (topex.ucsd.edu/gmtsar) to be using the new GMT C API. GMTSAR contains a mix of C programs linked with GMT and a suite of scripts, and these scripts (written using the C shell) are currently being ported to GMT 5 (gmt.soest.hawaii.edu/projects/gmt5sar). We expect that these scripts will eventually be ported to Python and thus rely on the GMT/Python API instead.
5. The Custom GMT project allows developers to add additional custom modules to GMT. This project depends on the full gmt-devel package, which provides access to lower-level GMT functions that are not part of the official high-level API but nevertheless are typically required to build new GMT modules. Custom modules can be bundled into separate shared plugin libraries that are accessed via the main gmt executable simply by placing the plugins in a special GMT plugin directory. The official GMT supplements are examples of such modules and are being treated as loadable plugins via a shared supplements library. The Custom project provides templates and examples of a few working GMT modules that can serve as starting points for developers targeting a specific workflow or data processing used by a particular community. One such plugin [*Wessel et al.*, 2015] providing access to custom processing and extraction of data supplied by the Global Seafloor Fabric and Magnetic Lineation project [*Seton et al.*, 2014] has been released via www.soest.hawaii.edu/PT/GSFML.
6. Finally, there is work being done by volunteers on designing a GMT/FORTRAN API as well (gmt.soest.hawaii.edu/projects/gmt-fortran-api). While the GMT C API presently has a few functions targeting F77 for basic grid input/output, the GMT/FORTRAN API will extend the official API to more modern versions of FORTRAN. At present, this is an unfunded project fully dependent on the contributions of volunteers. We expect it to slowly move forward using this development model.

2.2 GMT-MATLAB Interoperability

From WikiPedia, "**MATLAB®** (**mat**rix **lab**oratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. Developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran". Both scientists and engineers rely on MATLAB for technical work. As shown in Figure 3, access to

GMT from MATLAB can now be accomplished via a MEX (MATLAB Executable) function (also called **gmt**), which offers further access to all of GMT's 120+ modules as well as fundamental input/output of GMT data objects. Internally, the GMT/MATLAB C API defines five high-level composite data objects that handle input and output of data via GMT modules. These are data tables (representing one or more sets of points, lines, or polygons), grids (2-D equidistant data matrices), text tables (free-form text/data mixed records), color palette tables (i.e., color maps), and raster images (1–4 color bands). Correspondingly, we have defined five data objects that we use to facilitate the interface between GMT and MATLAB via the **gmt** MEX function. The GMT/MATLAB API is responsible for translating between the five GMT objects and native MATLAB objects, and references to data arrays are passed if transposing of matrices is not required. The five data objects are:

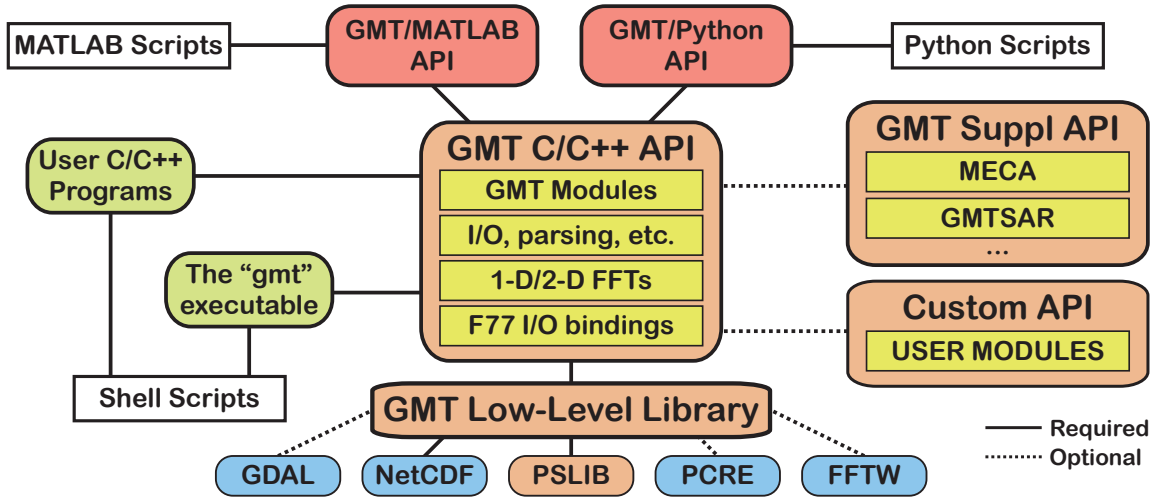

Fig. 3. *Conceptual block diagram of GMT 5 dependencies*. In GMT5, the high-level functionality resides in the API and any module may be called via the single **gmt** executable. Supplemental and custom APIs may also be accessed this way. The GMT/MATLAB and GMT/Python APIs will provide scripts in those languages direct access to all GMT modules, similar to how UNIX shell scripts access modules via the **gmt** executable [modified from *Wessel et al.*, 2013].

1. **Matrices**. This is the fundamental data type in MATLAB so data sets (other than equidistant grids) are always passed and received via the gmt MEX function in the form of a double-precision *nrows* by *ncolumns* matrix. Consequently, it is very easy to pass tables directly from MATLAB into those GMT modules that process data tables as well as to receive data matrices from GMT modules that process and produce data tables as output.
2. **Grids**. Many tools (GMT, but also others software) consider equidistant grids a particular data type and numerous file formats exist for saving such data. Because GMT relies on GDAL (Geospatial Data Abstraction Layer; www.gdal.org) we are able to read and write almost all such formats in addition to a native netCDF format that complies with both the COARDS and CF netCDF conventions (www.unidata.ucar.edu/software/netcdf/conventions.html). In the context of this API we have designed a native MATLAB grid structure that holds header information from the GMT grid as well as the data matrix representing the gridded values. These structures may be passed into GMT modules that expect grids and are returned from GMT modules that produce such grids.
3. **Color maps**. GMT uses its flexible Color Palette Table (CPT) formats to describe how the color (or pattern) of symbols, lines, polygons or grids should vary as a function of a state variable. In MATLAB, this information is provided in another native structure that holds the color map as well as an optional *alpha* array for transparency values. Like grids, these structures may be passed to GMT modules that expect CPT files and will be returned from GMT modules that normally would produce CPT files on standard output.

4. **Text cell array**. Because of their variable record lengths, text tables are best represented in MATLAB by a cell array. All GMT modules expecting text tables (or producing them) will use a cell array to communicate this information with MATLAB.
5. **Image**. The fifth GMT object, the raster image, shares many characteristics with the grid object except the bytes representing each node reflect gray shade, color bands (1, 3, or 4 for indexed, RGB and RGBA, respectively), and possibly transparency values. We therefore represent images in another native MATLAB structure that among other items contains three components: The image matrix, a color map (present for indexed images only), and an alpha matrix (for images specifying transparency on a per-pixel level).

The GMT and MATLAB combination is extremely flexible, letting the user harvest the general numerical and graphical capabilities of both systems, and represents a giant step forward in interoperability between GMT and other software package (i.e., beyond mere file format compatibility). It is difficult to do justice to the range of operations possible with this combination; in Figure 4 we present just a very simple MATLAB script that illustrates a mix of both MATLAB and GMT capabilities. All calculations and imaging is done at the MATLAB prompt; there is no longer any need to read or write data to or from the outside in order to communicate with GMT modules. The same capability is also available using Octave, a free MATLAB clone. Here we simply grid some data using two different algorithms and compare the two results using contour maps (MATLAB) and color images (GMT).

```
% Example of GMT/MEX mixed-use script
gmt ('create'); % Initialize a GMT session
load Jahns.txt; % Load in the x,y,z data set
% Perform spatial median decimation in GMT, get cleaned decimated data back
D = gmt ('blockmedian', '-R20/2180/30/3030 -I20', Jahns);
% Grid using GMT's surface and greenspline modules, with modest tension
G1 = gmt ('greenspline', '-R20/2180/30/3030 -I10 -St0.2 -D1 -V', D);
G2 = gmt ('surface', '-R20/2180/30/3030 -I10 -N5000 -C0.01 -T0.2 -V', D);
% Display grids and data points via 2 contour plots in MATLAB as Figure 1:
figure(1)
subplot (2,1,1)
v = 40:5:100;   % Contour grid every 5 meters
contour (G1.x, G1.y, G1.z, v); hold on; axis equal
plot (Jahns(:,1), Jahns(:,2),'r.')
subplot (2,1,2)
contour (G2.x, G2.y, G2.z, v); hold on; axis equal
plot (Jahns(:,1), Jahns(:,2),'r.')
% Display grids and data points via 2 GMT color maps
gmt ('grdimage', '-Jx0.001i -P -K -Ba -BWSne -Ctopo > map.ps', G1);
gmt ('psxy', '-R -J -O -K -Ss0.1c -Gblack >> map.ps', D);
gmt ('grdimage', '-J -O -K -Ctopo -Ba -BWsne -Y3.5i >> map.ps', G2);
gmt ('psxy', '-R -J -O -Ss0.1c -Gblack >> map.ps', D);
% Convert GMT figure to PNG and display in MATLAB as Figure 2:
I = gmt ('psconvert', '-TG -E200 -P -F -A map.ps');
figure (2)
imshow (I.image);
gmt ('destroy') % Terminate the GMT session
```

Fig. 4. *Example of GMT interoperability within a MATLAB script.* Various GMT modules (in **bold**) accept and return data matrices and grids, produce a *PostScript* plot, convert it to a transparent PNG image and view the image within the MATLAB session. While this is a simple example, we note that the **blockmedian** and **surface** combination powers the creation of many global data sets [*Becker et al.*, 2009] and that our gridding module **surface** is widely used across all sciences [the citations of our splines-in-tension algorithm [*Smith and Wessel*, 1990] exceed 1,000]. Notice how data objects (tables, grids, and images) are passed in and out of GMT.

Figure 5 shows the result of running this script in MATLAB. We show a screenshot of the resulting windows. MATLAB reads in a *x,y,z* dataset, uses GMT for spatial decimation using median statistics and performs two types of gridding, then displays contours and data in MATLAB as well as a color map plus data points in GMT, the latter *PostScript* plot being rasterized and loaded back into MATLAB for display.

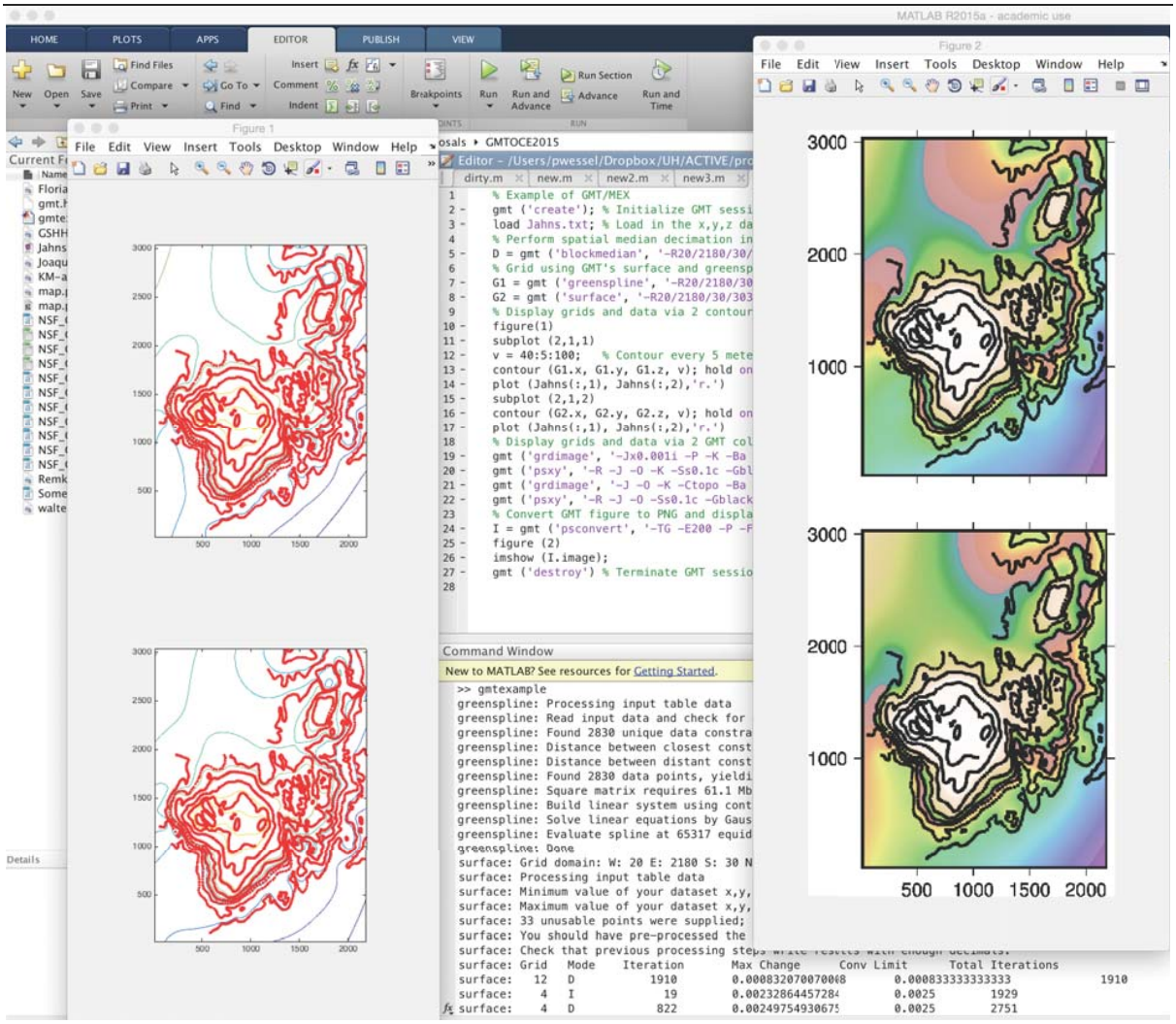

Fig. 5. *Screenshot of GMT interoperability with MATLAB*. The example script in Fig. 4 has run, completed all calculations and opened up the two figures: Figure 1 generated by MATLAB and Figure 2 generated by GMT, then rasterized to PNG and returned to MATLAB for display.

There are numerous other improvements to GMT that were implemented during the last proposal period; see **Results from Prior Support** for some details as well as the release notes on our website.

3. Proposed Work

We propose to tackle seven outstanding issues that will help position GMT to become less dependent on the current PI and to be better positioned for the inevitable transition to a community-driven model for infrastructure maintenance. The keywords for the next period are continuity, stability, simplification, interoperability and documentation. While our goals will require considerable work, the aim is to reach a state where further work will be much more manageable. It is vital that we simplify our code base,

strengthen interoperability and improve documentation across all aspects of GMT, both on the user and developer side. Consequently, we have seven major project tasks that we wish to complete during the proposal period; these are directly tied to the keywords listed above:

3.1 Task 1: Succession Planning [continuity, documentation]

Based on current projections, it is very likely that PI Wessel will retire within the next 7–10 years, and during this period he will have many other demands on his time that will require a tapering off of GMT involvement. It thus becomes important to consider how 30+ years of collective investment in GMT infrastructure can be maintained in a post-Wessel era. While this new era is far from imminent, we are under no illusion that it will be a simple transition. Over time, Wessel should relinquish his day-to-day leadership and retreat to play a supporting role in the maintenance of the GMT infrastructure but no longer be the essential participant that he is now. Moving from a PI-centric model to a loose federation of interested scientists and technicians will require a commitment to a focused and well-organized transition period. For this reason we propose to invest considerable time and energy into two key activities during the coming 5-year period:

Task 1a: We will establish a GMT Steering Committee of peers who will be charged with the task of overseeing and guiding the practical aspects of the transition. Our collaborator David Sandwell, Scripps Institution of Oceanography, has kindly agreed to initiate this committee and his key task is to assemble a technical group of GMT-aware scientists, across a broad range of ages and disciplines, to guarantee a competent and pro-active committee that has the fortitude to contribute to this vital planning. To avoid adding new costs, we propose that this committee should meet annually with PI Wessel and other available GMT developers attending the Fall AGU Meetings, with other business to be conducted over Skype and by email. To encourage their involvement, PI Wessel will provide quarterly reports to the committee. By involving outside scientists with a stake in the continuation of GMT we expect to explore a variety of options for how to solve this critical transition. This is uncharted territory for us and we wish to do it properly and for our procedures to serve as a model for similar endeavors. Long-term, it is likely the GMT source base may need to move to either a US government site (e.g., USGS, NOAA) or to an established Open Source site (e.g., SourceForge, GitHub) as it is unlikely the University of Hawaii will be able to serve as repository in a post-Wessel era.

Task 1b: We will start the assembly of a GMT Maintainer's Handbook. This "book" (which in the current era most likely will be a set of wiki pages) will explicitly detail all steps involved in the day-to-day upkeep of our web services as well as the testing, producing, and releasing of new GMT versions. Currently, too many of these steps are no more than mental bookmarks in various GMT team members' brains, and the lack of such a resource makes it exceedingly difficult for a new curator (or a group of curators) to take over these activities. We currently have completely undocumented aspects of the GMT Wiki site management, low-level details of the cmake setup, the evolution of the RST documentation, test suite scripting, etc. To date, we have strongly prioritized our documentation efforts on GMT usage but we will now have to dedicate resources toward crafting a dynamic handbook. The GMT Steering Committee will ensure that the Handbook is completed during the project period and its evolution will be a key aspect of the quarterly reports. As we develop the handbook we will let the various GMT developers take turns in going through the steps and run both simulated and actual executions of a GMT point release. We believe a thorough and orderly documentation of all these tasks is essential and will put us on a path towards a manageable transition.

3.2 Task 2: PROJ.4 dependency [simplification, interoperability]

While this task has been introduced as potential work in earlier proposals it has always taken a back seat to the major GMT 5 development and in particular the GDAL integration [www.gdal.org]. Reviewers may recall that the GDAL "bridge" allowed us to address much more pressing features, such as the ability to read and write any grid and image format. With those aspects now well implemented we

wish to turn our attention in earnest to PROJ.4. From Wikipedia, "PROJ.4 is a library for performing conversions between cartographic projections. The library is based on the work of Gerald Evenden at the USGS but is now an OSGeo project maintained by Howard Butler and Even Rouault". This library is the industry standard for map projections and offers far more projections than GMT currently is capable of. It also processes standard projection meta-data used by GIS packages that are essential for interoperability between GMT and other software, both commercial and open source. Specifically, this task will involve replacing all GMT calls to our low-level projection library (these functions are all conveniently isolated in the gmt_proj.c file) with equivalent functions in PROJ.4. An unintended side-effect of this transition will be access to a slew of new map projections in GMT by virtue of PROJ.4's larger offerings. Reviewers may wonder why we did not use PROJ.4 in GMT from the get-go and instead developed our own mapping sub-library. In fact, both GMT and PROJ.4 originated in the 1980s and were largely unaware of each other until around 2000, and more formal emails between Evenden and Wessel did not take place until 2008 (and even later with Karney). By that time we already had a fully functioning (albeit limited) projection library tuned to GMT and it was not perceived as a weakness that needed to be addressed immediately. There will be significant but relatively straightforward work involved in replacing gmt_proj.c functions with PROJ.4. However, it needs to be done delicately. Over the years, GMT has added improvements to the algorithms presented in Snyder [1987] as we discovered that Snyder's implementations of transverse Mercator could not go over the pole and down the other side, and so we had to rebuild our own, and in the course of that process we introduced auxiliary latitudes to facilitate conformal, equal area, and true distance developments of the ellipsoid. Thus, we must ensure we do not loose any capability in making the transition. Nevertheless, the benefits of this simplification are huge: (1) We will no longer need to maintain a (duplicate) set of complicated projection codes, (2) we will instantly gain access to additional map projections, (3) we will greatly improve our interoperability with other GIS software, and (4) PROJ.4 itself may be strengthened if GMT has capability that it is missing, as well as by adding so many new PROJ.4 users as new testers. These will have long-lasting and positive consequences for GMT users.

3.3 Task 3: OGR Bridge [simplification, interoperability]

In GMT 5 we completed the GDAL bridge, which allows GMT to read all grid file formats known to GDAL. This implementation greatly improved GMT interoperability as suddenly all kinds of grid files and even images could be read directly into GMT. The equivalent infrastructure in GDAL for geospatial "Simple Feature" data (e.g., tables with vector data of points, lines, and polygons) is called OGR. With the help of GDAL's creator (Frank Warmerdam) and with funding to Warmerdam from NIWA (New Zealand, via longtime GMT contributor Brent Wood) the new file format GMT/OGR was designed and implemented in GDAL, allowing the GDAL tool **ogr2ogr** to convert from a variety of geospatial formats (such as ESRI shapefiles) to GMT/OGR, which is an ASCII-based file format that can be read by GMT, even the legacy GMT 4 programs (the geospatial metadata are simply seen as comments). Adding the OGR Bridge would eliminate the need for this intermediate step and allow GMT (via the OGR bridge) to read and write such files directly. We have received considerable feedback from users wishing to use shapefiles with GMT and the detour via **ogr2ogr**, while operational, is seen as a stopgap measure. Like the GDAL bridge before it, implementing the OGR bridge will greatly improve the interoperability of GMT, perhaps eliminate the need for the GMT/OGR transitional file format entirely, and simplify our own vector data i/o library codes. Consequently, the OGR bridge would strengthen GMT's position as a GIS Swiss Army knife immensely. The maintainer of GDAL is now Even Rouault and the GDAL community is very active.

3.4 Task 4: Finalize MGG tools [stability, documentation]

During our expired grant period we added several new tools and enhanced existing ones, all with an MGG-specific focus. In particular, we added a new supplement (potential) that now contains tools for

calculating geopotential fields (e.g., via Talwani [*Talwani and Ewing*, 1960; *Talwani et al.*, 1959], Okabe [*Okabe*, 1979], and Parker [*Parker*, 1972] methodologies) and both 2-D and 3-D isostatic (flexural) calculations for different rheologies and load configurations, as well as reduction to the pole for magnetic anomaly grids and the evaluation of seamount province bathymetry from model parameters of individual seamounts. We wish to finalize this supplement with a few additional thermal modules (these exist but need to be repurposed for GMT), but more importantly we would like to improve the documentation and add a series of tests and working examples. We also would like to improve the documentation for our existing MGD77 processing tools. While these tools have been implemented they lack complete documentation and in particular suitable test scripts (see task 5). The tools are only as good as the documentation explaining their usage. Related to MGD77, the crossover error analysis tools (in the supplement x2sys), so frequently used by marine scientists collecting data at sea, will be elevated to the GMT core since they were in fact implemented as a generic tools set (i.e., any data files could be used to assess crossings and develop systematic corrections of any type). Finally, with the help of the undergraduate student we will expand every module's documentation to include more working examples and additional explanation of all options, with cross-links to documentation, cookbook, and example gallery and test suite pages. These additional examples will be added as mini test-scripts as well and this will strengthen the documentation by virtue of having actual working examples that can be tried by rookie users using ""cut-and-paste".

3.5 Task 5: GSHHG 3 [simplification]

Several GMT modules rely on our data set of coastlines, political borders, and rivers for map making or for data masking. The underlying GSHHG data set [*Wessel and Smith*, 1996] is also made available to non-GMT users in the form of binary tables and ESRI shapefiles. Unfortunately, the GSHHG data derive from vintage databases originating in the 1970s and 1980s (i.e., World Vector Shorelines [*Soluri and Woodson*, 1990] and CIA's World Data Bank II [*Gorny*, 1977]) and these data have numerous problems that are beyond repair without Herculean editing efforts. Over the years we have spent much time processing and upgrading these data. Fig. 6 is representative of what we typically find when zooming in on any particular part of the world. (Note: we receive numerous emails each year about similar shortcomings in GSHHG for almost any part of the world). Here we see GSHHG's full-resolution coastline (yellow line) compared to our proposed successor source OSM (OpenStreetMap; www.openstreetmap.org) in red. There are two obvious differences between the lines, highlighting long-standing issues with both accuracy and precision: (1) there is a registration problem in GSHHG where features are shifted in some direction by several hundred meters. Based on our experience these shifts vary in amplitude and direction with geographic location and likely derive from the initial scanning, digitizing, datum confusion and processing when the aforementioned data were first released. While we have occasionally applied Band-Aid solutions to improve the accuracy (e.g., shifting all polygons in the Society Islands to fit Google Earth imagery, improve river mouths, delete duplicate features, etc.) it is not a scalable approach to handle the entire world; (2) the precision of the data is quite limiting, especially for maps of smaller areas with linear dimensions of a few tens of km, and many users simply have had to obtain local data in order to make a suitable map with GMT. Obviously, the precision can only improve by adding better data. We have examined a handful of other locations around the world and the lessons from Fig. 6 are typical. We propose to assemble a new generation (GSHHG 3) database for GMT that will derive from OSM's public offerings. Rather than curating and maintaining this database (as we currently are doing for GSHHG 2), we will simply use existing processing tools and scripts for extracting the desired features from the OSM database and reformat them for use in GMT. This will include our successive dumbing-down to lower-resolution versions for use with regional and global maps, a task for which we use line-simplification software [*Douglas and Peucker*, 1973; *Pallero*, 2013]. Because OSM uses a cloud/wiki-based approach, registered and trusted users are able to improve on the product by editing the OSM data online. We see this as a win-win situation for GMT and OSM since GMT users will gain access to a much better database and OSM maintainers will gain support from a new group of

potential data editors. We will also avoid spending time on the maintenance of obsolete data and focus on our core competencies of GMT software development and maintenance. Finally, OSM also contains additional data that will be of interest to many GMT users, such as cities, feature names, streets, and much more.

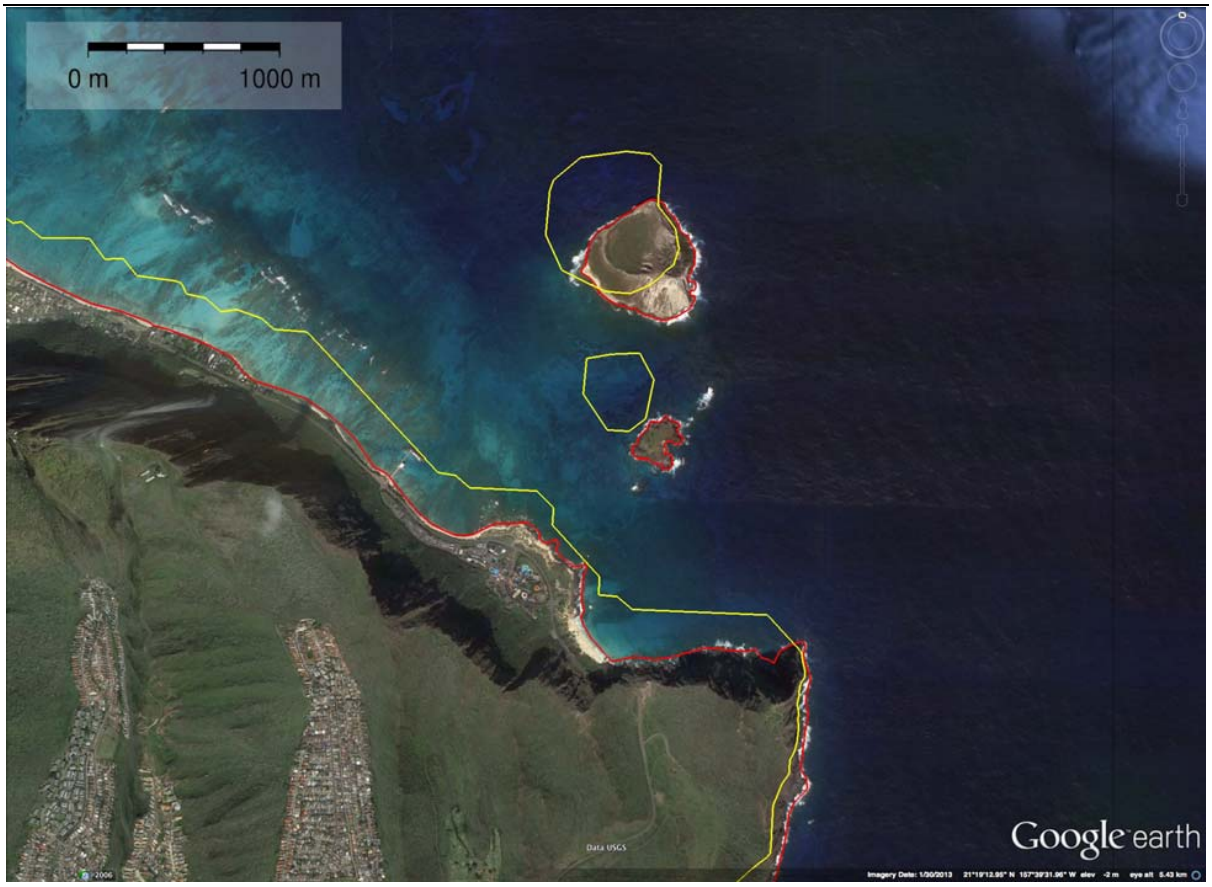

Fig. 6. *The limitations of GSHHG*. Easternmost point on Oahu, Hawaii, showing the uninhabited islets Manana and Kaohikaipu across from the tourist attraction Sea Life Park. The GSHHG full-resolution coastline is drawn in yellow, while the OSM coastline is shown in red. Note the lower precision of GSHHG as well as its lack of accuracy.

3.6 Task 6: Code hardening [stability, simplification]

As is well understood, all software known to man has bugs. Our wiki and its bug/feature-tracking tool have greatly improved our ability to keep track of outstanding bugs and feature requests and to process them in an orderly fashion. No more are emails from users getting lost in the depths of our infinite inboxes since the issues now remain online and can be tracked until fixed. Our approach to manage bugs is complemented by our script database. Whenever a user discovers a bug we generate one or more test scripts (often with the user's help) that demonstrate the problems. These scripts are then added to our database of similar scripts and we run these scripts quite often, especially before a new release is being prepared. Depending on the severity of the problem we will put resources towards understanding the bug and correcting it. To date, we have over 500 scripts that exercise all the GMT modules and many of their options. Yet, with over 120 GMT modules this is actually not a sufficiently large number of test scripts, i.e., it is less than 5 tests per module on average. Given the number of ways

GMT modules can be invoked we need many more scripts per module. We wish to dramatically increase the number of test scripts (i.e., by an order of magnitude) by taking advantage of our large user base. We will ask users to submit scripts they have created to make figures or to process data and we will modify them to fit into our testing infrastructure. Adding new and more numerous scripts will test additional aspects of GMT and increase the likelihood of exposing (and thus allowing us to correct) weaknesses in the code base. We will utilize the help of an undergraduate student in computer science to streamline this process. The student will also overhaul all examples given in our documentation to make sure that (a) they use correct syntax and can be successfully run via copy/paste to the command line, and (b) that we give an adequate number of examples per module to cover the available options. Note that adding working scripts will prevent new bugs from going unnoticed, as our automatic procedure will detect changes in results or changes in plots compared to the stored originals. This is how we become aware of changes that break previously working parts of GMT. Even subtle problems like unintended shifts of annotations placements will show up as a failure when compared to the stored originals. We use **graphicsmagic** module **convert** to detect changes in *PostScript* plots between software revisions and our test procedure reports which of the plots have changed.

Another aspect of code hardening is optimization. With informal help from the San Diego Supercomputer Center we have made great strides by adding OpenMP to parallelize computing-intensive sections of GMT (such as convolutions and other slow and repetitive code sections), and we have begun replacing old linear algebra solvers with highly optimized LAPACK library functions. We wish to extend these efforts to all suitable candidate modules in the GMT core as well as the supplements. Again, the gains from this task are two-fold: (1) faster execution times, especially on multi-core processors, and (2) the elimination of hard-to-maintain linear algebra code in GMT by relying on externally maintained and rock-solid numerical libraries.

3.7 Task 7: PostScript Light Library [stability, simplification]

Our final task is rather simple but nevertheless an important one. At the bottom of the GMT library stack lies our *PostScript* plotting library, PostScript Light (PSL). This is where projected coordinates and preferences from GMT are translated into the *PostScript* graphics language instructions that printers understand. It is almost completely decoupled from GMT apart from system configuration include files that the GMT cmake installation prepares. We propose to isolate this library from the GMT repository and create a separate installation package for it. The PSL library would then become a required prerequisite in order to install GMT (much like netCDF, GDAL, and many other standard libraries are). Scharroo and Wessel currently maintain PSL and we expect to continue this work as a labor of love. The gain to current and future GMT maintainers lies in the simplification of having one less library to maintain, and as an added benefit it also makes life simpler for a small group of non-GMT users who need access to PSL but currently have to go through the much more involved GMT installation process. While presented last, this is probably the first task we will attack and complete within a few weeks.

4. Management Plan

PI Wessel will supervise the project and will be responsible for communication with the other GMT developers. He is also the direct supervisor of the undergraduate assistant and will ensure the student is used on tasks that commensurate with his or her expertise. Wessel will delegate tasks to optimize the use of our expert volunteers' skills and spare time but will also be carrying out much of the proposed work. He will also prepare quarterly reports to the GMT Steering Committee so that this group will gain a better understanding of our progress and can better act to address any difficulties that may arise during the course of the project. PI Wessel (and other GMT core developers) will meet with the committee annually, presumably during the fall AGU meeting, and via Skype every six months or as the situation dictates.

Most of the seven tasks will be pursued in parallel and will depend on availability of our volunteer's spare time. The longer time-horizon applied to all tasks makes it easier to manage swings in access to labor.

During our previous grant period we commenced our work with a weeklong GMT developer summit in Hawaii, with all four GMT developers participating (Dr. Wobbe had not yet joined the team at that time). This meeting allowed for extensive dialogues between the developers, something 12-hour time-zone differences between Hawaii and Europe and the back and forth of email trails make very cumbersome. The summit was extremely successful as it laid the groundwork for all of the changes we subsequently have implemented in GMT 5. We wish to replicate this success in order to ensure that consensus is reached after extensive debate and we will be using the agreed-upon approaches to plan the paths forward. For this proposal, we suggest that these summits instead take place at the Scripps Institution of Oceanography. This location will enable synergistic activities with the GMT Steering Committee (Sandwell is at Scripps) and will reduce the overall travel budget considerably. Given the willingness of our volunteer developers to dedicate time in their busy schedules for long travels and meetings to serve GMT, we propose a developer meeting of one full week to discuss and elaborate on specific plans to address all the proposed tasks. The summit will produce a written technical plan to address the above tasks but in much more detail and will be shared with the GMT Steering Committee. We will coordinate the scheduling of this week to allow Sandwell to first form the committee so we may interact with this body, either in person or via Skype. We propose two such summits during the 5-year period: One in Year 1 to stake out the course and another summit in Year 4 to address course corrections and the finalization of the project. Because of the complexity of the succession we have decided the second summit is vital and thus warranted.

5. Relationship of proposed work to current EAR/Geoinformatics grant

The NSF EAR/Geoinformatics program awarded a grant to a new collaboration between the University of Hawaii (Wessel), UCSD (Sandwell) and the University of Wisconsin (Feigl) to strengthen GMT in several EAR-specific areas. The key tasks of this collaboration are

T-1: Extend GMT to analyze and display more diverse types of data
T-2: Build an Application Programming Interface (API) between GMT and Python
T-3: Build tools for Interferometric Synthetic Aperture Radar (InSAR)
T-4: Develop scripts for deploying GMT on the Open Science Grid (OSG)

We wish to point out that the overlap between the tasks discussed in this proposal and those of the Geoinformatics grants are almost nonexistent. The Geoinformatics work is focused on expanding GMT capabilities in certain areas (e.g., InSAR processing, structural geology symbolism, ternary diagrams, seismology and geodesy symbolism, and high-performance grid computing) whereas the present proposal focuses on simplifications, reduces redundancies, improves documentation, and paves the way for the inevitable succession of GMT leadership.

6. Broader Impacts

The GMT scientific toolset enjoys exceptionally broad usage across numerous scientific fields and all continents, and its usage continues to expand, particularly to the Windows platform, making both the MATLAB and coming Python APIs very significant due to the relative poor support for a UNIX-like experience under Windows. The tasks proposed here, such as the PROJ.4 integration, the OGR bridge and GSHHG 3, will improve interoperability and therefore ensure that GMT will reach an even broader audience.

7. Results from Prior NSF Support

(a) NSF Award Number: OCE 1029874; *Amount:* $275,042 [including supplement] *Period of Support:* 9/1/2010-08/31/2015
(b) Title of the project: Support of the Generic Mapping Tools (GMT)
(c.1) Summary of Results (Intellectual Merit): During the 5-year project 2010–2015 period we have made available 9 GMT 4.x releases and 4 GMT 5.x releases. GMT 5 represents an entire new branch of GMT and is decoupled from the old GMT4 source base. With version 5, GMT has become an API library with a simple driver program (**gmt**) that call upon the high-level GMT modules in the shared library. Over 200 new features have been implemented to address user requests, in particular from the large MGG/OCE community of users. The previous two build processes (separate ones for UNIX and Windows) have been replaced with a single cross-platform approach using **cmake**. We have successfully transferred both GMT 4 and 5 code bases from CVS to subversion control, retaining the change history from previous versions. We have also designed a brand new GMT website and wiki, with full bug tracking support, user forums (subscription required), source code browsing, online active documentation, and installation instructions. GMT file-level interoperability has been expanded: The OGR/GMT format has been finalized and documented on our site [Appendix P]; GDAL's implementation has been tested and found compliant. The GDAL tool **ogr2ogr** translates ESRI shapefiles and others to the new OGR/GMT format that GMT can read and access aspatial data for setting plot properties. Similarly, a new GMT common option (**-a**) allows users to reformat a GMT dataset into an OGR/GMT file with **gmt convert**, whose output can be converted via **ogr2ogr** back into an ESRI shapefile (or many other GIS formats). When GMT is built with GDAL support we can also read almost any grid format known to man, since GDAL is used to read grids that GMT cannot handle by itself (this is transparent to the user). The GMT and GIS worlds have moved much closer to each other, allowing the easy exchange of data between systems. Our coastline database GSHHG (previously GSHHS) continues to evolve and is now at version 2.3.4 (we have released 9 updates since 2010). We have replaced the entire Antarctica coastline and islands with newer and much more accurate data from the Atlas of the Cryosphere [*Bohlander and Scambos*, 2007]. We have also updated and added a version of the Digital Chart of the World (DCW) to allow painting, outlining, or clipping of individual countries or states. We have been building a GMT/MATLAB API on top of the GMT 5 C API and expect the beta-version to be available early this fall. Finally, we have fixed hundreds of bugs, large and small, over the 5-year project timeline.
(c.2) Summary of Results (Broader Impact Activities): We have, with international cooperation, strengthened the GMT website and made it easier for GMT users to install our latest versions. We are seeing broader adoption of GMT packages by software managers, which will result in wider use of GMT. Specific outreach includes improving GMT documentation, including the addition of video podcasts, more examples, and clearer instructions.
(d) Publications resulting from the NSF award; and (e) Research products:

Wessel, P. et al., The Generic Mapping Tools, http://gmt.soest.hawaii.edu.

Wessel, P. and W. H. F. Smith, GSHHG, A Global Self-consistent, Hierarchical, High-resolution Geography Database http://www.soest.hawaii.edu/pwessel/gshhg.

Wessel, P., W. H. F. Smith, R. Scharroo, J. F. Luis, and F. Wobbe (2013), Generic Mapping Tools: Improved version released, *Eos Trans. AGU*, *94*(45), 409–410, doi:10.1002/2013EO450001.

E. References

Becker, J. J., Sandwell, D. T., Smith, W. H. F., Braud, J., Binder, B., Depner, J,, Fabre, D., Factor, J., Ingalls, S., Kim, S.-H., Ladner, R., Marks, K. M., Nelson, S., Pharaoh, A., Trimmer, R., von Rosenberg, J., Wallace, G., Weatherall, P. (2009), Global Bathymetry and Elevation Data at 30 Arc Seconds Resolution: SRTM30_PLUS, *Marine Geodesy*, *32*, 355–371.

Bohlander, J., and T. Scambos (2007), Antarctic coastlines and grounding line derived from MODIS Mosaic of Antarctica (MOA), edited, National Snow and Ice Data Center, Boulder, Colorado.

Caress, D. W., and D. N. Chayes (1996), Improved processing of Hydrosweep DS multibeam data on the R/V/ Maurice Ewing, *Mar. Geophys. Res.*, *18*, 631–650.

Douglas, D. H., and T. K. Peucker (1973), Algorithms for the reduction of the number of points required to represent a digitized line of its caricature, *The Canadian Cartograhper*, *10*(2), 112–122.

Gorny, A. J. (1977), World Data Bank II General User Guide*Rep. PB 271869*, 10pp pp, Central Intelligence Agency, Washington, DC.

Luis, J. F. (2007), Mirone: A multi-purpose tool for exploring grid data, *Computers & Geosciences*, *33*(1), 31–41.

Okabe, M. (1979), Analytical expressions for gravity anomalies due to homogeneous polyhedral bodies and translations into magnetic anomalies, *Geophysics*, *44*(4), 730–741.

Olson, C. L., J. J. Becker, and D. T. Sandwell (2014), A new global bathymetry map at 15 arcsecond resolution for resolving seafloor fabric: SRTM15_PLUS, in *Eos Trans. AGU*, edited, pp. Abstract OS34A-03.

Pallero, J. L. G. (2013), Robust line simplification on the plane, *Computers & Geosciences*, *61*(0), 152-159, doi:http://dx.doi.org/10.1016/j.cageo.2013.08.011.

Parker, R. L. (1972), The rapid calculation of potential anomalies, *Geophys. J.*, *31*, 447–455.

Sandwell, D. T., R. Mellors, X. Tong, M. Wei, and P. Wessel (2011), GMTSAR: An InSAR Processing System Based on Generic Mapping Tools, *Scripps Institution of Oceanography Technical Report*.

Seton, M., Whittaker, J. M., Wessel, P., Müller, R. D., DeMets, C., Merkouriev, S., Cande, S., Gaina, C., Eagles, G., Granot, R., Stock, J., Wright, N., Williams, S. E. (2014), Community infrastructure and repository for marine magnetic identifications, *Geochemistry, Geophysics, Geosystems*, *15*(4), 1629–1641, doi:10.1002/2013GC005176.

Smith, W. H. F., and P. Wessel (1990), Gridding with continuous curvature splines in tension, *Geophysics*, *55*(3), 293–305, doi:10.1190/1.1442837.

Snyder, J. P. (1987), *Map projections – A working manual*, US Govt. printing office, Washington, DC.

Soluri, E. A., and V. A. Woodson (1990), World Vector Shoreline, *Int. Hydrograph. Rev.*, *LXVII*(1), 27–35.

Talwani, M., and M. Ewing (1960), Rapid computation of gravitational attraction of three-dimensional bodies of arbitrary shape, *Geophysics*, *25*(203-225).

Talwani, M., J. L. Worzel, and M. Landisman (1959), Rapid gravity computations for two-dimensional bodies with application to the Mendocino submarine fracture zone, *J. Geophys. Res.*, *64*, 49–59.

Wessel, P., K. J. Matthews, R. D. Müller, A. Mazzoni, J. M. Whittaker, R. Myhill, and M. T. Chandler (2015), Semiautomatic fracture zone tracking, *Geochem. Geophys. Geosyst.*, *16*, doi:10.1002/2015GC005853, doi:10.1002/2015GC005853.

Wessel, P., and W. H. F. Smith (1991), Free software helps map and display data, *EOS Trans. AGU*, *72*(41), 441, doi:10.1029/90EO00319.

Wessel, P., and W. H. F. Smith (1995), New version of the generic mapping tools released, *Eos Trans. AGU*, *76*(33), 329, doi:10.1029/95EO00198.

Wessel, P., and W. H. F. Smith (1996), A global, self-consistent, hierarchical, high-resolution shoreline database, *J. Geophys. Res.*, *101*(B4), 8741–8743, doi:10.1029/96JB00104.

Wessel, P., and W. H. F. Smith (1998), New, improved version of Generic Mapping Tools released, *Eos Trans., AGU*, *79*(47), 579, doi:doi:10.1029/98EO00426.

Wessel, P., W. H. F. Smith, R. Scharroo, J. F. Luis, and F. Wobbe (2013), Generic Mapping Tools: Improved version released, *Eos Trans. AGU*, *94*(45), 409–410, doi:10.1002/2013EO450001.