Volume Title
ASP Conference Series, Vol. **Volume Number**
Author
© **Copyright Year** Astronomical Society of the Pacific

Space Missions: Long term preservation of IDL-based software using GDL

Alain Coulais[1], Marc Schellens[*] , Sylwester Arabas[2], Maxime Lenoir[3,1], Léa Noreskal[1], Stéphane Erard[3]

[1]*LERMA CNRS and Paris Observatory, 61 Av. de l'Observatoire, 75014 Paris, France*

[*]*GDL project leader*

[2]*Faculty of Physics, University of Warsaw, Poland*

[3]*LESIA, Observatoire de Paris, CNRS, UPMC, Université Paris-Diderot, 5 place Jules Janssen, 92195 Meudon France*

Abstract.
GNU Data Language (GDL) is an Open Source clone of IDL[1], an interactive language widely used in Astronomy and space missions since decades. Proprietary status, license restrictions, price, sustainability and continuity of support for particular platforms are recurrent concerns in the Astronomy community, especially concerning space missions, which require long-term support.

In this paper, we describe the key features of GDL and the main achievements from recent development work. We illustrate the maturity of GDL by presenting two examples of application: reading spectral cubes in PDS format and use of the HEALPix library. These examples support the main argument of the paper: that GDL has reached a level of maturity and usability ensuring long term preservation of analysis capabilities for numerous ground experiments and spaces missions based on IDL.

1. Presentation of GDL

GDL is free software released under the GNU GPL v2/v3 license. All the key dependencies of GDL have GNU GPL-compatible licenses and are mainstream libraries, which shall guaranty of long term support: Plplot, ImageMagick, GNU Scientific Library, FFTw ...

The GDL interpreter has full compatibility with IDL 7.1 and includes support for selected features introduced in IDL 8 (e.g. negative array indexing, FOREACH statement, automatic garbage collection). Except for the widget support (GUI programming) which exists but is still limited, GDL covers most core functionalities of IDL and has reached a state where it becomes a viable alternative to IDL. It can be used to run large pipelines based on codes and libraries written in IDL syntax, reading back data, doing complex computations and delivering accurate results.

[1]RSI then ITTVIS and now EXELIS

Reviewed papers in various fields[2] (Solar Physics, Data processing, Cosmology...) are based on computations done with GDL, using well known third party libraries written using IDL syntax, reading same input data files and giving the same results than IDL.

Widely used libraries written in IDL syntax (e.g. AstroLib (Landsman 1993), HEALPix (Górski et al. 2005), MPfit (Markwardt 2009)) are fully or to a large extent compatible with GDL. Several formats, including FITS (Wells et al. 1981), internal IDL Save format[3], Planetary Data System (PDS), and several other file formats used in other disciplines such as geosciences and medical imaging (HDF, HDF5, NetCDF, DICOM, ...) are supported to both reading and writing in GDL.

Thanks to many packagers, pre-compiled and pre-configured versions of GDL are regularly shipped for most main Linux distributions (Debian, Ubuntu, Fedora, Gentoo, ArchLinux...), FreeBSD as well as Max OS X (via Macports and Fink). A large test suite helps to avoid regressions, assure reliability for computations and simplify deployments on new systems, where customization may be required. GDL is open to contributors and it does receive notable feedback from the scientific and open-source communities. That helps to trace bugs, to extend the functionality, and to ensure compatibility with IDL-based libraries.

2. Progress and Status

Two years ago, during the XIX ADASS conference (Coulais et al. 2010), we listed some key weakness of GDL, including performance issue, delay between introduction of new features by the developers and shipment of binary packages, regressions, unreliable graphical output features (Postscript in particular). We did achieve a notable progress.

• Packaging: As a result of intensified campaign among packagers the most recent 0.9.2 version of GDL was packaged for seven different packaging systems within a week after the release!

• Support and documentation: A new GDL website was created and filled with updated content. A mailing-list was created[4], you can subscribe to receive sporadic informations (serious bugs, announces of new versions, etc). A draft of users' guide[5] was created and have since been step-by step updated.

• The PostScript output and the so-called direct-graphics routines: A series of improvements in the quality of PostScript output as well as in the completeness of the direct-graphics interface was introduced in the most recent version of GDL (0.9.2). One could say that the overal status has changed from "alpha" to "pre-beta" – it is still not perfect, but it gradually improves. Feedback and help are very welcome and indeed crucial to keep the work progressing.

• Performance: Thanks to numerous optimisations and an efficient use of the OpenMP paralellisation available in newer compilers (e.g. GCC ≥ 4.2) GDL perfor-

---

[2] some are listed here http://gnudatalanguage.sourceforge.net/resources.php

[3] http://www.physics.wisc.edu/~craigm/idl/cmsave.html Due to licences incompatibilies with GNU GPL, CMSV lib. cannot be included directly in GDL software

[4] https://sympa.obspm.fr/wws/info/gdl-announces

[5] http://gnudatalanguage.sf.net/gdl.pdf

mance improved significantly, in particular on the ubiquitous multi-core machines. The benchmarks (e.g. IDL's TIME_TEST3) reveal that the calculations with GDL 0.9.2 are as fast (in some cases faster) as with IDL 7.1 and 8.0. We need to recode SMOOTH using C++, which makes a perfect application work for a student...

• Widgets: A limited support for GUI programming is now available in GDL. Despite numerous requests, we received few (but positive) feedback.

• Regression tests: The coverage of the test suite has increased. It was also extended to verify compatibility with external IDL-written libraries such as MPfit, HEALPix, AstroLib (FITS I/O) etc. As a result, we are much more confident in the overall quality of the code, and we experience much less regressions.

3. Exemples of long term preservation of pipelines within GDL

Space missions and large telescopes often use a particular file format (e.g. FITS or PDS), develop a pipeline for pre-processing (flat fielding, despiking, ...), and transform raw data into scientifically validated data (calibration, non linearity corrections, pointing, ...). After intense developments, the software is frozen but it is still required for reading back raw data, perhaps testing new processing methods, porting to new computers, delivers data through the Internet... We do not discuss here the quality of the graphical outputs or interactive widget-based interfaces, but focus on the capability to read back input data, process them and write correct outputs. We consider that the capabilities currently available and tested in GDL should suffice for most pipelines written in IDL for past space missions, after a quick audit of the code. In recent years, we have accumulated experience in ingesting large codes in GDL, because scientists wanted to check to which extent GDL is able to run specific libraries. Although GDL only covers ~60% of the intrinsic pro/func of IDL 6.1 (54% in C++, 6% in IDL/GD syntax) this subset of functionalities is now large enough to ensure good overall support of standard codes[6].

• The AstroLib library: Most libraries written using IDL syntax and related to Astronomy use AstroLib (Landsman 1993), if only to read and write FITS files. In several years, we received no bug reports related to AstroLib, only requests for missing functionalities. In 2010, we made a deep audit of AstroLib on the IDL side and updated missing keywords in GDL (e.g. new keywords /Cumulative and /Integer in TOTAL and PRODUCT). GDL can also process with success the NRAO pg93 FITS test files.

• PDS format and Virtis library: the Planetary Data System (PDS) is a long term archive preservation format for NASA Solar System missions, now widely used by other agencies (ESA, JAXA...). Support of the PDS format is a long standing issue in Planetary Science. VirtisPDS (a.k.a LecturePDS) is an IDL library developed in support of the VIRTIS experiments on the Rosetta and Venus-Express ESA missions[7]. It includes a general purpose PDS reader able to access many PDS data sets, particularly focused on support of imaging spectroscopy. Originally, the library could not run under GDL, mostly because of missing features in GDL and some limitations in the library.

---

[6] A matrix based on IDL 6.1 and GDL 0.9 http://aramis.obspm.fr/~coulais/IDL_et_GDL/Matrice_IDLvsGDL_intrinsic.html was transformed in a very illustrative graphical way http://michaelgalloy.com/2009/12/01/routines-currently-available-in-gdl.html

[7] http://voparis-europlanet.obspm.fr/othertool.shtml

Some missing features were implemented, e.g. DIALOG_PICKFILE and wider support for endianness. A few changes were made in the library to avoid features still missing in GDL, in particular concerning numerical types and error catching. Less than a month work of a student without background in IDL was needed to fix all these issues. The current version was tested using a variety of files from data archives in many fields. The current VirtisPDS library can run with either IDL (≥ 5.5) or GDL (≥ 0.9). The licensing freedom of GDL also makes it a favourite solution to support access to PDS-formatted services in a Virtual Observatory framework. Such a VO system is being defined in the EuroPlaNet context.

• HEALPix: Widely used in cosmology, HEALPix is a library dedicated to projection of data on a sphere. It contains two main parts: numerical computations and graphical outputs. After few interactions with one of the HEALPix developers a few missing functionalities (e.g. managing gzip FITS files) were added to GDL, and a test suite including calls to the HEALPix internal test suite is now shipped in the GDL tarball (testsuite/test_healpix.pro). No discrepancies between IDL and GDL with regard to numerical computations were found. Except one PostScript output which cannot be done now in GDL, all other graphical outputs (PNG, PS) are very similar to the IDL ones. HEALPix in GDL was used for preparing a reviewed paper (Roukema 2010).

4. Conclusions

The existence of peer-reviewed research papers based on computations done using GDL supports the main message of this paper: GDL has become a mature software. The interpreter is stable and covers the whole IDL syntax except few features added in IDL 8. Computations in GDL 0.9.2 are as fast or even faster than IDL 7 and 8 on multi-cores.

It is time you give a chance to GDL for your past and future pipelines. We are ready to consider (1) the inclusion of dedicated test suites in our test suite (2) collaborating for custom developments for missing functionalities.

Today, we consider that GDL is ready for long term preservation of IDL-based pipelines as long as you take time to make sure that your codes are using the subset of IDL functionalities GDL provides.

Acknowledgments. The GDL team would like to thank all the known and anonymous contributors who report problems, bugs, and also provide pieces of code. We also especially warmly thank all the packagers (Non exhaustive nominative list). AC thanks Léa and Maxime, two excellent students who worked on GDL last summer, M.-F. Landréa and UFE team at Paris Observatory for their support, É. Hivon from IAP who reported tricky discrepancies and provided a full test suite for HEALPix and G. Duvert who provided USERSYM.

References

Coulais, A. et al., 2010, in ADASS XIX, vol. 434 of ASPCS, 187. arXiv:1101.0679
Górski, K. M. et al. 2005, ApJ, 622, 759. arXiv:0409513
Landsman, W. B. 1993, in ADASS II, vol. 52 of ASPCS, 246 ADS
Markwardt, C. B. 2009, in ADASS XVIII, vol. 411 of ASPCS, 251 arXiv:0902.2850
Roukema, B. F. 2010, A&A, 518, A34. arXiv:1004.4506
Wells, D. C., Greisen, E. W., & Harten, R. H. 1981, A&AS, 44, 363 ADS