

Report on LAL/LALApps

Jolien Creighton, Julien Sylvestre, Alberto Vecchio and Igor Yakushin
on behalf of the LSC Data Analysis Software Working Group *

March 10, 2004

Overview

LAL is a set of C libraries containing the core routines for gravitational wave data analysis. LALApps is a set of codes, including full-blown search codes, pipelines, and scripts, based on LAL functions. LAL is written in C; it is not based on any commercial software, but relies on open source packages, such as FFTW. Jolien Creighton is the software librarian. About 40+ authors have contributed so far to LAL/LALApps. The web page for LAL/LALApps is:

- <http://www.lsc-group.phys.uwm.edu/lal/>

LAL and LALApps documentation is available, respectively, at:

- <http://www.lsc-group.phys.uwm.edu/lal/lsd.pdf>
- <http://www.lsc-group.phys.uwm.edu/lal/lalapps.dpf>

There are two LAL mail lists/archives for general announcements, including release announcements, and general discussion forum:

- <http://www.lsc-group.phys.uwm.edu/mailman/listinfo.cgi/lal-announce>
- <http://www.lsc-group.phys.uwm.edu/mailman/listinfo.cgi/lal-discuss>

Bug reports can be found at:

- <http://www.lsc-group.phys.uwm.edu/lal/bugs.html>

LAL/LALApps source code is maintained in a CVS repository at

- `:pserver:anonymous@gravity.phys.uwm.edu:/usr/local/cvs/lal`

The library records information about version, tags, configuration arguments, and this is duplicated in the headers.

*In order to gather information and feedbacks on LAL/LALApps the authors of this report have distributed a questionnaire to the whole LSC and the chairs of the upper-limit working groups

Platforms and libraries

LAL/LALApps is supported for a broad range of platforms, including Solaris and Linux. Work is close to completion to support the library on the native M\$ Windows build system, but LAL works on cygwin. Testing has stopped on IRIX, HPUX and DEC/OSF1 (but Jolien still compiles on DEC/Linux with gcc and ccc). LAL/LALApps now only supports particular combinations of autoconf and automake – this, however, only applies if one is building LAL from CVS rather than a distributed tarball. For grid applications Scott Koranda has successfully built LAL on the TeraGrid machines, which are IA-64 running SUSE.

LAL/LALApps can hook in metaio, LAM, MPICH, and FrameL. Due to the recent introduction of FFTW3 in LAL, support for double precision FFTs will be provided very shortly. There is no built-in graphics support in LAL. There is a broad consensus among the users that GSL should be made a requirement. Standard data discovery tools are missing at present. Work in order to alleviate this problem is currently being done by Scott Koranda. It would be useful if LAL could provide the functionality to transparently access data in some local archive, given just the desired channel(s) and GPS time range.

In the context of Grid applications, LAL should be able to build against the Globus SDK in the near future.

Documentation, testing and validation

Authors are responsible for providing documentation, which, at present, is very uneven (some packages come with thorough documentation, others contain no documentation at all).

Unit tests are even more uneven. Some vital functions have undergone careful testing/validation (e.g. antenna beam patters, in-spiral binary waveforms, timing routines for pulsar searches); unit tests for a substantial fraction of functions mostly consist of just example code. The integrated tests of pipelines tend to be more thorough; however, the approach to and supervision of testing/validation are de facto left to the UL chairs and vary from group to group (e.g. the stochastic background data analysis pipeline in LALApps was validated using DSO and a MATLAB code).

Specifications

LAL specifications are described in the LIGO document T990030-E available at <http://www.lsc-group.phys.uwm.edu/lal/lalspec.pdf>. A revision of the specifications is worth considering at this stage with the goal of distilling down the existing specifications to a small(er) set of recommendations in order to reduce overheads in coding (a typical example is to loose the number-of-arguments rule). In fact, there is a general consensus that it is fairly “heavy” to code to the specs. Any substantial revision of the specifications at this stage would be extremely unpractical because of the inertia involved in altering stylistic conventions.

Summary and recommendations

LAL is C library containing GW specific routines. LALApps is a set of codes, full pipelines etc for data analysis that has been extensively used for LSC astrophysical searches. About 75% of all the upper-limits results are now obtained using LAL/LALApps. The package is far from “perfect” (it is essentially impossible to find enthusiastic fans of LAL/LALApps in the LSC), however has finally reached a point of being useful: the core GW specific routines are used on a daily basis by a broad community and have undergone thorough validation. The benefit of LAL being a library is that many different environments (e.g. LALApps, DMT, MATLAB) can link to it, and the LSC should take advantage of this feature.

It is our recommendation that the LSC continues support of LAL(/LALApps) in the *near future*. The LSC might consider the following recommendation to improve LAL:

- LAL specifications could be revised (but not radically changed) with the aim reducing coding overheads;
- If MATLAB is to become an LSC "official platform", it is important to ensure the ability of calling LAL functions from MATLAB (and vice versa) by providing the necessary wrappers/MEX-programmes – this applies, in general, to all the official data analysis platforms;
- GSL should be included into LAL, and other packages might be considered;
- Functionalities at present available in dacondAPI could be wrapped in LAL;
- An advanced debugger could be developed for LAL code in Matlab or in Root, for instance, in order to simplify and encourage the validation of LAL functions by the community, as opposed to just their authors (there has been some limited success in compiling LAL with cint for Root by K Rawlins, but this effort is still quite preliminary);
- The structure of LALApps should be re-discussed;
- LAL should be integrated into GRID tools for large scale analysis.

The *long term future* of GW specific routines is an important issue, whose solution is outside the charge of this sub-group. We simply point out that there are essentially three approaches: (i) rewrite these utilities for each framework (DMT, LALApps, or whatever the LSC happens to adopt), (ii) use a common cross-platform and cross-framework library (either LAL or something partially/totally different) or (iii) provide a common C++ interface to LAL among different environments (DMT, MATLAB, GEO++). Each approach has advantages and disadvantages. Approach (i) requires more effort in terms of re-implementation and verification, but could provide more flexibility and allow for cross-checking. It might have non-negligible implications for any "community-approach" to data analysis. Approach (ii) would save effort, is closest to the status-quo and would probably ease the issue of verification. It would also promote a community development. Clearly, if one sticks to LAL the effort is subject to the cumbersome LAL specifications; if a new library is re-written almost from scratch a substantial amount of effort is required – but likely not as much as in (i). Approach (iii) allows to leverage existing LAL functions (not the search codes) on a very short time scale and, therefore, provides a way of using LAL functions in other data analysis environments.