Space Missions: long term preservation of IDL-based softwares using GDL

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

[1]LERMA CNRS and Paris Observatory, France

[*]GDL project leader

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

[3]LESIA CNRS and Paris Observatory, Section de Meudon, France

ADASS XXI – November 9, 2011

sorry for my English

What is GDL ? Motivations

GDL : http://gnudatalanguage.sourceforge.net/

Free clone of IDL
under GNU GPL v2 or later
fully syntax compliant with IDL 7 version
few IDL 8 new syntax features (negative index ...)

Motivations:

- smart language and accessible for Scientists
- large amount of existing codes, pipelines, libraries in IDL syntax
- perennity ; mid term and long term existence of IDL (which OS ?)
- to get rid of licensing and pricing and *jetons* on the web/cloud/cluster (ha, firewalls and *jetons* ! ha, summer (students) time and *jetons*)

We think the IDL users community in Astronomy is (still) wider than the Python one.

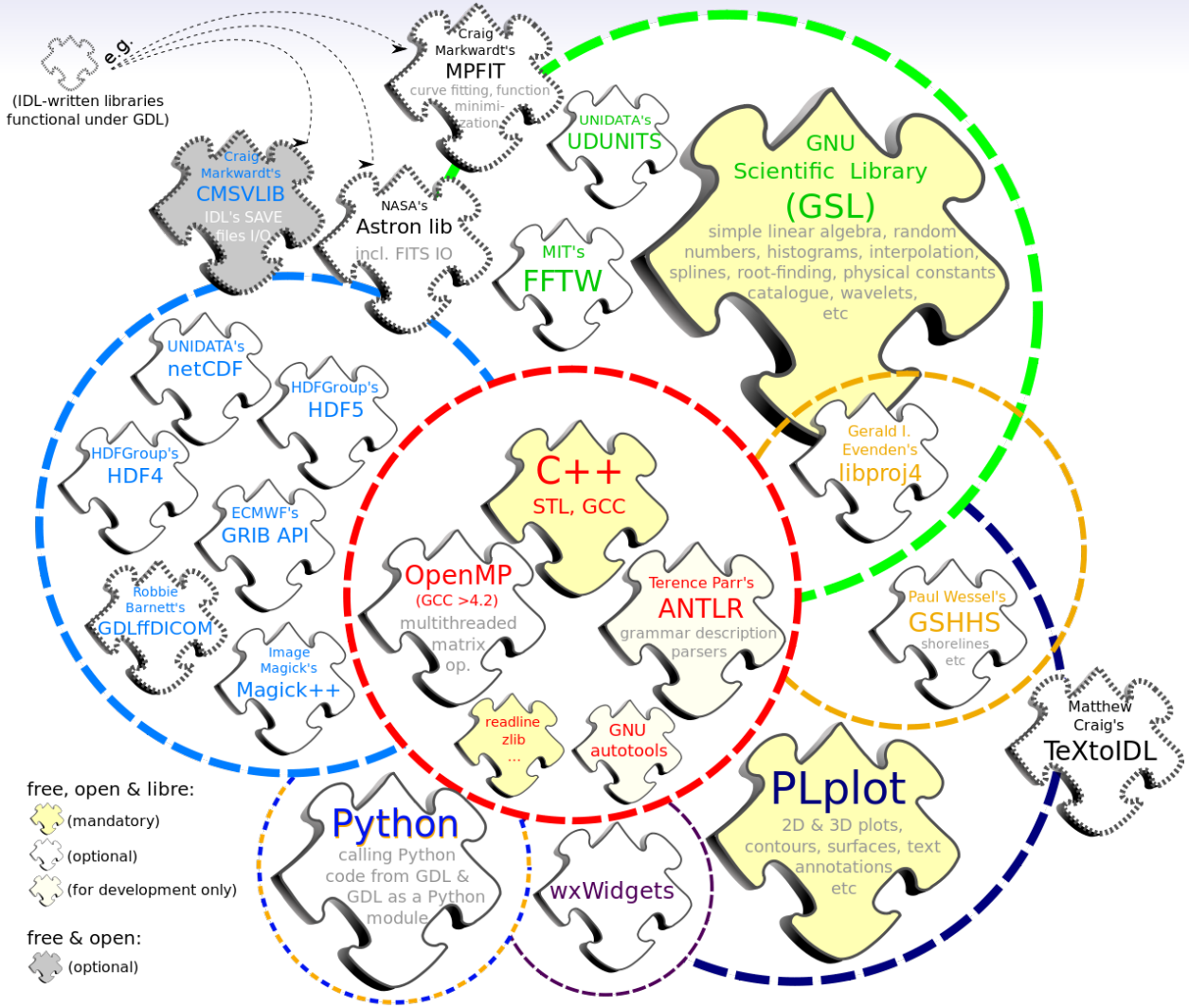


Figure: Mandatory and optional dependencies for GDL (fig. by Sylwester)

OS: Unix; packets vs compilation

Tested on: OSX (most recent flavours), Gentoo, Debian, Ubuntu, CentOS, FC, SL, Mandriva, Mageia, various *BSD flavours, SunOS ...

No plans at all for M$ systems (but who used that in XXI century ?)

Despite good support from packagers, packets are often in late. We are in GDL 0.9.2 today !

Two ways for compilation: `configure` OR `CMake` (less than 5 minutes compilation on multicores machines)

- $\sim$ 120 000 lines in C++
- $\sim$ 10 downloads per day (TGZ, not packets)

Performances : thanks to OpenMP and Marc !

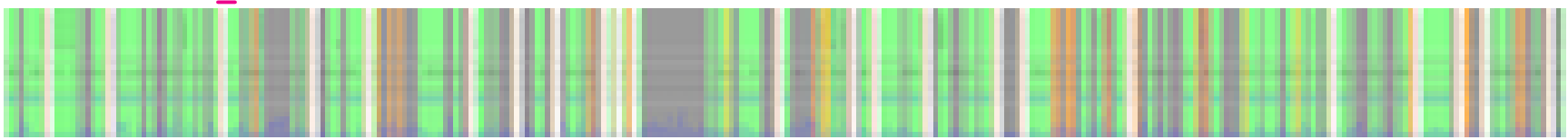
TIME_TEST3 on Debian 16 cores (idl-8; Xeon L5520 @ 2.27 GHz) and CentOS 8 cores (idl-7; Xeon X5450 @ 3.0 GHz) [output with TIME_COMPARE]

Time	GDL_16c	GDL_8c	idl7_8c	idl8_16c	
0.03	124*	112	118*	100^	Empty For loop, 2000000 times
0.01	138*	100^	119*	151*	Call empty procedure (1 param) 1000
0.01	150*	147*	100^	198*	Add 200000 integer scalars and stor
0.01	154*	134*	100^	181*	50000 scalar loops each of 5 ops, 2
0.00	130*	100^	318*	436*	Mult 512 by 512 byte by constant an
0.01	126*	100^	124*	164*	Shift 512 by 512 byte and store, 30
0.01	127*	100^	252*	303*	Add constant to 512x512 byte array,
0.01	141*	100^	234*	154*	Add two 512 by 512 byte arrays and
0.00	100^	116*	433*	320*	Mult 512 by 512 floating by constan
0.01	128*	119*	100^	107	Shift 512 x 512 array, 60 times
0.00	100^	138*	650*	330*	Add two 512 by 512 floating images,
0.01	149*	100^	118*	128*	Generate 1000000 random numbers
0.01	238*	182*	100^	125*	Invert a 192^ 2 random matrix
0.02	265*	158*	100^	162*	Transpose 384^ 2 byte, FOR loops
0.01	166*	100^	143*	168*	Transpose 384^ 2 byte, row and colum
0.04	102	104	101	100^	Transpose 384^ 2 byte, TRANSPOSE fun
0.02	217*	144*	100^	105	Log of 100000 numbers, FOR loop
0.00	100^	147*	515*	196*	Log of 100000 numbers, vector ops 1
0.01	158*	100^	160*	171*	131072 point forward plus inverse F
0.03	741*	513*	100^	116*	Smooth 512 by 512 byte array, 5x5 b
0.01	694*	600*	100^	127*	Smooth 512 by 512 floating array, 5
0.02	103	146*	356*	100^	Write and read 512 by 512 byte arra
0.39	173*	135*	101	100^	Total Time
0.01	118*	100^	118*	118*	Geometric mean

^ = fastest. * = Slower by 15% or more.

Completeness of GDL vs IDL intrinsic

Visualization of IDL intrinsic Pro/Func recoded in GDL
http://michaelgalloy.com/page/10
from http://aramis.obspm.fr/~coulais/IDL_et_GDL/Matrice_IDLvsGDL_intrinsic.html



Theoretical percentage of completeness for intrinsic Pro/Func

color code	syntax	number	%
green	recode in C++	214	52.6 %
orange	recode in GDL syntax	32	7.9 %
green + orange	(total recoded)	246	60.4 %
gray	still missing	161	39.6 %
	total	407	100.0 %

BUT, as a 20 years old IDL coder (processing data or simulations with the following instrument or satellites : NRH, PdBi, ALMA, ISO, Spitzer, AstroF, Planck ...), today, in my whole IDL codes, I miss TWO: USERSYM and the famous WMENU !
Do you know/use: BLAS_AXPY, FILE_READLINK, QHULL, XDXF ?!

The real problem is more on the keyword side, and especially the graphical keywords.

Contributions welcome !

Preparing your code for GDL

Please distinguish *pipeline/computation* vs *preparing camera ready output*

- compiling the last GDL version : we delivered GDL 0.9.2 today !
- collecting the needed dependencies (IDL Lib., AstronLib, CMSVlib (XDR files)) in the GDL_PATH
- preparing a basic end-to-end pipeline
- do you have a way to check that the reading of data is OK ?
- do you have a way to check that the final result is OK ?
- which null test can you do ?
- test-and-trial procedure to locate missing functionalities
- real audit (missing pro/func, missing keywords ...)
- building a deterministic test suite

Two examples: Virtis/PDS and HEALPix

Virtis/PDS

Adaptation of the Virtis/PDS *LecturePDS* (*reading PDS*) library (http://pds.nasa.gov/)
About one month of work for a student (Maxime)
Main problems:

- few bugs in STRING-related procedures
- missing transparent support for GZIP files
- big/small endian tricks
- were missing DIALOG_PICKFILE and FILE_SEARCH
- few code cleaning in Virtis/PDS lib.

and that was all ! All examples (big data files) were processed identically by IDL and GDL.

Thanks to Léa, Maxime and Stéphane

HEALPix

Can the HEALPix library (http://healpix.jpl.nasa.gov) been used within GDL ? See `test_healpix.pro` in `testsuite/`.

- Yes for the computations
- Yes for most of the graphical outputs (PNG and PS), except one PS output

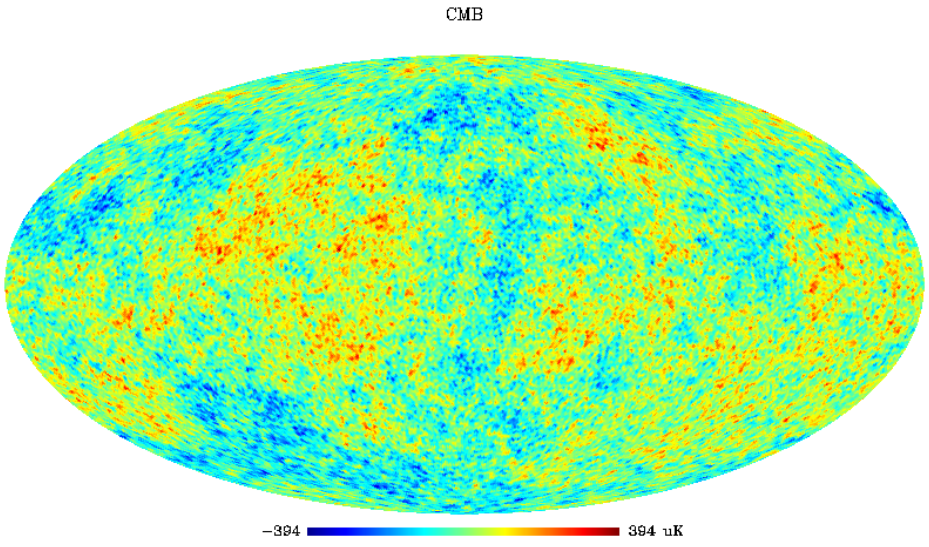


Figure: Reading FITS file and display CMB map

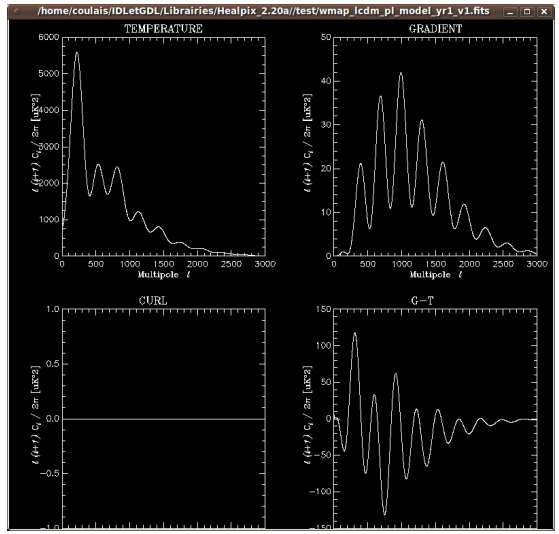


Figure: Simulation (large computations) then C_l plotting

Thanks to Éric Hivon

What you can do for GDL ?

- Testing GDL on your box (make check). We are facing problems related to Plplot and ImageMagick tricks. Wider base welcome.
- Testing your code with last GDL version.
- Report bugs (if any) (please report on SF bug tracker)
- Report missing Pro/Func and keywords you really need.
- Providing code (in C++ or in GDL syntax). Due to the simple API it is quite easy to add to GDL your Pro/Func coded in C++.
- Packaging !

- We are ready to consider collaborations.
- We may help you (at least during audit)
- Very good subjects for stagiaires/students.

When coming with adequate test code, each progress is irreversible !

Conclusion

GDL is a mature free clone of IDL:

- syntax OK
- good performances
- most of the key procedures/functions are available
- efficient regression test suite

but still

- few *useful* pro/func are missing (e.g. STRMATCH, LUDC, ...)
- some keywords are missing, especially in graphical functions
- may have un-expected bugs

If your concern is mainly preserving IDL based pipelines, it is easy to check whether you can live with the subset we provide !
Some large libraries in IDL syntax can be used now in GDL, e.g.: PDS/Virtis, ULySS, HEALpix, TexToIDL, MPfit, AstroLib, Cappellari' lib. ...
Sci. Refereed papers are written now using GDL.