From Technical Change Team

Selecting an LSST-approved Math Library

A proposal for DM to use GSL and FFTW as scientific math libraries was accepted by the TCT on 18 November 2008 with the caveat that suitable matrix functions might need to be acquired from a different source.

Case for a Math Library

from RL

Please suggest where to get C++ source code for the following classes:

1. A class to generate Gaussian deviates. The constructor requires the following inputs: mean, standard deviation, and random-number-generator seed. The class should have a function with void inputs that returns a deviate (double).

2. A class to generate Poisson deviates. The constructor requires the following inputs: mean and random-number-generator seed. The class should have a function with void inputs that returns a deviate (int).

from RHL

Steve needs to work on the background estimation (and thus sims for test cases etc.)

from SMM

I'm not sure if unit testing is relevant to this discussion, but I did end up incorporating a Mersenne twister code into the association pipeline in order to generate synthetic test data for the spatial cross match unit tests. There's very primitive support for getting deterministic behaviour: each unit test run prints out the PRNG seed it used, and there's a compile time option for forcing use of a particular seed. It's certainly not urgent, but it would be nice to replace that with calls to an agreed upon (and well tested) library.

from AJC

We are indeed using random number generators heavily in image sims and have discussed handling the seeds (as we need to be able to reproduce the images). We haven't reach a conclusion about how to manage them; we had to table it until we have the DC3 image generation running.

from TSA

I do agree that converging on a set of math libraries is urgent.

Case for Standardizing on C++ Gnu Scientific Library (GSL)

TBD

from RAA

> Why choose GSL over any other C++ scientific math library?
> Fill in this section...

Comment by jmyers on Fri 14 Nov 2008 03:16:32 PM CST

GSL is already in use by Pan-STARRS, so using GSL for LSST will make sharing of MOPS code easier to manage.

GSL should be sufficient for MOPS, including the new Deep Stack code I'm working on (we really just need least-squares fitting, I believe)-jmyers

Comment by Robert Lupton Mon, 17 Nov 2008 17:58:42 -0500

I'm not ready to adopt a standard for matrix algebra yet. The GSL wraps blas but not lapack, and the interface is clunky. Also, I see that they use cblas about which I have heard bad things.

The special functions and such like in GSL should be OK --- we don't have very tough demands, and there's useful stuff there (although not as much as I had thought)

So I'd like to allow users to use GSL and I vote YES on GSL. This is NOT to say that I think that we should be calling gsl routine directly; I'd rather see them wrapped into the lsst::afw::math namespace. This is partly tidiness and partly the realisation that we may want to change our minds --- my canonical worry is Gaussian variables and random seeds; the classic Box-Muller transform caches a value and setting the seed may or may not clear the cache [Numerical recipes doesn't] making things non deterministic.

I also vote Yes on FFTW, with the proviso that we'll want to hide it from the users within an LSST class --- I wouldn't want to simply use Image (for one thing, you can't use fftw to FFT an image in situ).

Comment by Andy Becker Mon, 17 Nov 2008 15:37:36 -0800

I think this is OK, especially if we hide the implementation in lsst::afw::math as Robert has suggested. This will also allow us to throw exceptions based upon the error code return. GSL also has the option to use an optional tuned ATLAS library instead of the standard CBLAS that comes with it - are there any arguments against using this?

Add comment

Your email or username:

GSL Licensing

from GPDF

The GSL is under the GPL, not the LGPL (licenses). That means if we want to release software that uses it, we have to release it under the GPL. Is that compatible with the decisions that have been made already on licensing of LSST software?

from RP

Our working understanding is that using GSL is compatible with our licensing plans.

I will note that GSL uses GPLv3 which is means that Apache is "compatible" with it.

It remains to be determined if our licensing plans are flawed. That said, our licensing plans--specifically the goal of allowing commercial use of our software--should NOT dictate whether we use an otherwise good, open-source library. I will repeat for the record that I am ready to agree to drop the plans to release Apache and go with GPL on the grounds that the licensing issues are taking up to much attention.

Python PyGSL

from RO

I have similar questions about how we will perform math operations in Python if we adopt GSL.

To do numerical operations in Python will we be sticking to numpy, or will be creating GSL vectors and matrices and operating on them with GSL functions? (There is a PyGSL as Andy Connolly determined during our group meeting today, but it is apparently not complete).

from FP

My understanding is that PyGSL uses numpy arrays directly. Also while not complete, it implements most of GSL:

?http://pygsl.cvs.sourceforge.net/viewvc/pygsl/pygsl/README?view=markup

Of course, I could be wrong...

Side issues raised

Random-Seed Control

from GPDF

Separate from the GSL question itself, this ticket leads me to ask: is there already a random-seed control policy in place for LSST software development? A policy on where in LSST software it is permissible to use computed random numbers at all?

from AJC

Not sure it impacts the immediate gsl needs but I'd be happy if we came up with a project statement on seeds etc (as well as which random number generators are optimal/portable)

from RHL

No, there's no policy, and no requirement for demonstration of deterministic use of any random numbers, which can be tricky due to e.g. caching on Box-Muller transforms.

Vector/Matrix? wrapper class

from AB

In a non-gsl specific question, are we going to have a LSST vector/matrix class that hides the implementation details, or would us programmers be using gsl_vector/gsl_matrix directly? Whatever we use it would be optimal if we could find a way to share the data instead of having to do copies (e.g. putting Images into matrices).

from RHL

After some private discussion with Andy, it seems that what he has in mind is setting a row of a matrix to the values of an Image. I think that this is out of scope --- it can't be done without copies unless you assume that the data is contiguous in memory which, in general, it isn't.

The question of constructing Images from numpy arrays and vice-versa is different, and could be handled via an Image constructor (-> C++) and a smallish piece of C in swigland (-> python). We could do this, but the memory management is tricky if we don't copy data.