GNU Scientific Library (GSL)

Http://www.gnu.org/software/gsl/

GSL Team

M.Galassi, J. Theiler, G. Jungman (LANL)
Brian Gough (GNU)

What is GSL ?

- Numerical library for the GNU system
- Development started in 1996
- Written in ANSI C
- Free software under the GNU GPL
- Relase 1.0 was in 2001
- About 1000 functions (rngs, special fns, ...)
- Currently at release 1.3

What's GNU?

- Project to create free Unix operating system
- Started in 1984
- In use today: GNU/Linux
- Demonstration of the viability of free software model

What is Free Software?

- **Four freedoms**
  - **0) to run the program**
  - **1) to study the program**
  - **2) to modify the program**
  - **3) to share the program**

Why Free Software?

- Copyright system invented several centuries ago
- Designed for books
- Software is different
  - Difference between source and object files
  - Development through incremental improvements
- Different system appropriate

Software Model

- **Software should be considered as a field of applied mathematics/computer science**
  - **Everyone works together to solve problems**
  - **Everyone benefits from the results**

GSL Motivation

- Needed a numerical library that could be used in free software (GPL'd) applications
- Existing Libraries
  - Proprietary: NAG, IMSL
  - Numerical Recipes (not free)
  - ....
- Proprietary licenses incompatible with large-scale scientific collaboration

Functionality (Ported Packages)

- **Ports of well known public domain Fortran packages**
  - **FFTPACK**
  - **MINPACK**
  - **QUADPACK**
  - **MISCFUN**
  - **VEGAS / MISER**
  - **BLAS (CBLAS)**

Functionality

Complex Numbers	**Roots of Polynomials**	**Special Functions**
Vectors and Matrices	**Permutations**	**Sorting**
BLAS Support	**Linear Algebra**	**Eigensystems**
Fast Fourier Transforms	**Quadrature**	**Random Numbers**
Quasi-Random Sequences	**Random Distributions**	**Statistics**
Histograms	**N-Tuples**	**Monte Carlo Integration**
Simulated Annealing	**Differential Equations**	**Interpolation**
Numerical Differentiation	**Chebyshev Approximation**	**Series Acceleration**
Discrete Hankel Transform	**Root-Finding**	**Minimization**
Least-Squares Fitting	**Physical Constants**	**IEEE Floating-Point**

Example: Special Functions

An example of a Bessel function $J_0(5)$,

```
#include <stdio.h>
#include <gsl/gsl_sf_bessel.h>
int
main (void)
{
  double x = 5.0;
  double y = gsl_sf_bessel_J0 (x);
  printf ("J0(%g) = %.18e\n", x, y);
  return 0;
}
```

```
Output J0(5) = -1.77596771314338292e-01
```

DESIGN

- One language: C
- Object oriented
- Algorithm Components
- Layered (BLAS)
- Reliable error estimation
- Testing

See: GSL Design Document http://sources.redhat.com/gsl

C Language

- **GNU's universal language / interface**
  - **support any platform with ANSI C compiler**
  - **Compatible with GNU software, GNOME, GTK, ...**
- **Easy for binding to other languages**
  - **Python**
  - **Scheme (GNU GUILE)**
  - **C++ (extern "C")**
- **Well established standard by 1996**

Object Oriented Design

- **Represent class of algorithm by a C struct with internal state** (Kiem-Phong Vo "An Architecture for Reusable Libraries" - VMALLOC, CDT, SFIO)
- **Example: random number generator:**

```
gsl_rng_type * T = ....
gsl_rng * r = gsl_rng_alloc (T);
double x = gsl_randist_gaussian (r, sigma) ;
```

- **'T' contains function pointers to implementation**

```
struct { void (*set) (void * state, ...);
         int (*get) (void * state, ...); }
..
```

Object Oriented Design (cont)

- **Use with**
  - **RNGs (and Quasi-RNGs)**
  - **Root finding (1d and Multidimensional)**
  - **Minimisation (1d and Multidimensional)**
  - **Non-linear least squares fitting**
  - **Differential Equations**
  - **Interpolation / splines**

Algorithm Components

- Algorithms broken down into components
  - Initialise
  - Iterate
  - Test
- User drives the algorithm (no callbacks)

```
gsl_multroot_fsolver * s;
s = gsl_multiroot_fsolver_alloc (T, &f, x);
do {
  iter++;
  status = gsl_multiroot_solver_iterate(s);
  if (status) break;
  status = gsl_multiroot_test_residual(s->f, 1e-3);
} while (status == GSL_CONTINUE && iter < 1000);
```

BLAS

- Library built over BLAS for efficiency
- GSL supplies default BLAS (-lgslcblas)
  - written in C
  - supports all operations (Level 1, 2, 3)
  - portable, no machine specific optimisation
- Recommend ATLAS for performance
  - automatically tuned BLAS
  - free software

Example: Error Estimation

- **Reliable error estimates are required**
- **Example $J_0(5)$ with error estimate,**

```
#include <stdio.h>
#include <gsl/gsl_sf_bessel.h>
int main (void) {
  double x = 5.0;
  gsl_sf_result r;  /* { result.val, result.err } */
  int status  = gsl_sf_bessel_J0_e (x, &r);
  printf ("J0(%g) = %.18e +/- %g\n", x, r.val, r.err);
  return 0;
}
```

```
J0(5) = -1.77596771314338264 2e-01 +/- 1.93021e-16
```

Licensing

- GPL vs LGPL ("Lesser/Library GPL")
- LGPL gives too much away
- Advantages of the GPL
  - contributors can retain ownership (dual licensing)
  - encourages users to release free software

Future

- **Need better organisation of users and developers**
  - **Editorial board**
  - **Consortium of users**