1 Introduction

The GNU Scientific Library (GSL) is a collection of routines for numerical computing. The routines have been written from scratch in C, and present a modern Applications Programming Interface (API) for C programmers, allowing wrappers to be written for very high level languages. The source code is distributed under the GNU General Public License.

1.1 Routines available in GSL

The library covers a wide range of topics in numerical computing. Routines are available for the following areas,

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

The use of these routines is described in this manual. Each chapter provides detailed definitions of the functions, followed by example programs and references to the articles on which the algorithms are based.

Where possible the routines have been based on reliable public-domain packages such as FFTPACK and QUADPACK, which the developers of GSL have reimplemented in C with modern coding conventions.

1.2 GSL is Free Software

The subroutines in the GNU Scientific Library are "free software"; this means that everyone is free to use them, and to redistribute them in other free programs. The library is not in the public domain; it is copyrighted and there are conditions on its distribution. These conditions are designed to permit everything that a good cooperating citizen would want to do. What is not allowed is to try to prevent others from further sharing any version of the software that they might get from you.

Specifically, we want to make sure that you have the right to share copies of programs that you are given which use the GNU Scientific Library, that you receive their source code

or else can get it if you want it, that you can change these programs or use pieces of them in new free programs, and that you know you can do these things.

To make sure that everyone has such rights, we have to forbid you to deprive anyone else of these rights. For example, if you distribute copies of any code which uses the GNU Scientific Library, you must give the recipients all the rights that you have received. You must make sure that they, too, receive or can get the source code, both to the library and the code which uses it. And you must tell them their rights. This means that the library should not be redistributed in proprietary programs.

Also, for our own protection, we must make certain that everyone finds out that there is no warranty for the GNU Scientific Library. If these programs are modified by someone else and passed on, we want their recipients to know that what they have is not what we distributed, so that any problems introduced by others will not reflect on our reputation.

The precise conditions for the distribution of software related to the GNU Scientific Library are found in the GNU General Public License (see [GNU General Public License], page 467). Further information about this license is available from the GNU Project webpage *Frequently Asked Questions about the GNU GPL*,

```
http://www.gnu.org/copyleft/gpl-faq.html
```

The Free Software Foundation also operates a license consulting service for commercial users (contact details available from `http://www.fsf.org/`).

1.3 Obtaining GSL

The source code for the library can be obtained in different ways, by copying it from a friend, purchasing it on CDROM or downloading it from the internet. A list of public ftp servers which carry the source code can be found on the GNU website,

```
http://www.gnu.org/software/gsl/
```

The preferred platform for the library is a GNU system, which allows it to take advantage of additional features in the GNU C compiler and GNU C library. However, the library is fully portable and should compile on most systems with a C compiler.

Announcements of new releases, updates and other relevant events are made on the `info-gsl@gnu.org` mailing list. To subscribe to this low-volume list, send an email of the following form:

```
To: info-gsl-request@gnu.org
Subject: subscribe
```

You will receive a response asking you to reply in order to confirm your subscription.

1.4 No Warranty

The software described in this manual has no warranty, it is provided "as is". It is your responsibility to validate the behavior of the routines and their accuracy using the source code provided, or to purchase support and warranties from commercial redistributors. Consult the GNU General Public license for further details (see [GNU General Public License], page 467).

2 Using the library

This chapter describes how to compile programs that use GSL, and introduces its conventions.

2.1 An Example Program

The following short program demonstrates the use of the library by computing the value of the Bessel function $J_0(x)$ for $x = 5$,

```
#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;
}
```

The output is shown below, and should be correct to double-precision accuracy,[1]

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

The steps needed to compile this program are described in the following sections.

2.2 Compiling and Linking

The library header files are installed in their own '`gsl`' directory. You should write any preprocessor include statements with a '`gsl/`' directory prefix thus,

```
#include <gsl/gsl_math.h>
```

If the directory is not installed on the standard search path of your compiler you will also need to provide its location to the preprocessor as a command line flag. The default location of the '`gsl`' directory is '`/usr/local/include/gsl`'. A typical compilation command for a source file '`example.c`' with the GNU C compiler `gcc` is,

```
$ gcc -Wall -I/usr/local/include -c example.c
```

This results in an object file '`example.o`'. The default include path for `gcc` searches '`/usr/local/include`' automatically so the `-I` option can actually be omitted when GSL is installed in its default location.

2.2.1 Linking programs with the library

The library is installed as a single file, '`libgsl.a`'. A shared version of the library '`libgsl.so`' is also installed on systems that support shared libraries. The default location of these files is '`/usr/local/lib`'. If this directory is not on the standard search path of your linker you will also need to provide its location as a command line flag.

[1] The last few digits may vary slightly depending on the compiler and platform used—this is normal.

To link against the library you need to specify both the main library and a supporting CBLAS library, which provides standard basic linear algebra subroutines. A suitable CBLAS implementation is provided in the library '`libgslcblas.a`' if your system does not provide one. The following example shows how to link an application with the library,

```
$ gcc -L/usr/local/lib example.o -lgsl -lgslcblas -lm
```

The default library path for `gcc` searches '`/usr/local/lib`' automatically so the `-L` option can be omitted when GSL is installed in its default location.

2.2.2 Linking with an alternative BLAS library

The following command line shows how you would link the same application with an alternative CBLAS library '`libcblas.a`',

```
$ gcc example.o -lgsl -lcblas -lm
```

For the best performance an optimized platform-specific CBLAS library should be used for `-lcblas`. The library must conform to the CBLAS standard. The ATLAS package provides a portable high-performance BLAS library with a CBLAS interface. It is free software and should be installed for any work requiring fast vector and matrix operations. The following command line will link with the ATLAS library and its CBLAS interface,

```
$ gcc example.o -lgsl -lcblas -latlas -lm
```

If the ATLAS library is installed in a non-standard directory use the `-L` option to add it to the search path, as described above.

For more information about BLAS functions see Chapter 13 [BLAS Support], page 122.

2.3 Shared Libraries

To run a program linked with the shared version of the library the operating system must be able to locate the corresponding '`.so`' file at runtime. If the library cannot be found, the following error will occur:

```
$ ./a.out
./a.out: error while loading shared libraries:
libgsl.so.0: cannot open shared object file: No such
file or directory
```

To avoid this error, either modify the system dynamic linker configuration[2] or define the shell variable `LD_LIBRARY_PATH` to include the directory where the library is installed.

For example, in the Bourne shell (`/bin/sh` or `/bin/bash`), the library search path can be set with the following commands:

```
$ LD_LIBRARY_PATH=/usr/local/lib
$ export LD_LIBRARY_PATH
$ ./example
```

In the C-shell (`/bin/csh` or `/bin/tcsh`) the equivalent command is,

```
% setenv LD_LIBRARY_PATH /usr/local/lib
```

The standard prompt for the C-shell in the example above is the percent character '`%`', and should not be typed as part of the command.

[2] '`/etc/ld.so.conf`' on GNU/Linux systems.