GNU Scientific Library – 設計と実装の指針

マーク・ガラッシ (Mark Galassi)
Los Alamos National Laboratory
ジェイムズ・ティラー (James Theiler)
Astrophysics and Radiation Measurements Group, Los Alamos National Laboratory
ブライアン・ガウ (Brian Gough)
Network Theory Limited
とみながだいすけ訳 (translation: Daisuke Tominaga)
産業技術総合研究所 生命情報工学研究センター

Table of Contents

GSL について

GNU Scientific Library は科学技術計算のためのサブルーチンを集めたライブラリである。一般の科学技術研究分野において、よく行われる (およびさほどでもない) 各種の計算を労力をかけずに行うためのものである。さらにソースコードも公開することで、計算法を改変したり、目的にあわせて調整したり、API やアルゴリズムそのものを改良したりすることができる。

GNU Scientific Library は C あるいは FORTRAN 用の既存の商用ライブラリ (NAG、IMSL CNL、IBM ESSL、SGI SCSL など) の代替となる、フリーの実装となることを目的としている。

利用環境として、安価なデスクトップタイプのワークステーションを想定している。汎用で、かならずしも専門家でなくても使えるライブラリとなるよう意図している。

1 目標

科学技術計算サブルーチン集に要求される条件には、以下のようなものがある。

- フリーであること (「無料」ではなく「自由」の意味の free。詳細は GNU General Public License を参照のこと)。誰もが自由に利用、再配布、改変などができること。
- C 言語で書かれており、コーディング規約、API、変数のスコープの決め方などが時代遅れでないこと。
- 教育的配慮のある、はっきりと書かれた解説文書があること。info、WWW、TeX の形式に変換してオンラインにしやすいので TeXinfo で書かれているのが望ましい。
- 洗練されたアルゴリズムが実装されていること。
- *autoconf* と *automake* に対応し、移植性に優れ、利用するシステムにあわせた設定が行いやすいこと。
- GNU の精神に正しく則っていること。

これらの観点からすると、既存のライブラリには一長一短がある。

Netlib (http://www.netlib.org/) は AT&T が管理している、おそらくはもっとも先進的な数値計算アルゴリズム集であり、オンラインで公開されている。惜しむらくは、ほとんどのコードは FORTRAN で書かれており、ライブラリ関数の呼び出し方が多くの場合、風変わりである。また、うまくまとまっていないので、利用するのに手間がかかる。

GAMS (http://gams.nist.gov/) は非常によくまとまった、科学技術計算のリンク集である。しかし netlib 同様、それぞれのルーチンの品質や文書化の度合いにはばらつきがある。

Numerical Recipes (http://www.nr.com, http://cfata2.harvard.edu/nr/) は優れた本で、非常に明確に各アルゴリズムを解説している。著者は、本に載せているコードを読者が利用できるようにライセンスを定めているが、その再配布は認められていない。Numerical Recipes のコードは、*freedom* の意味での *free* ではない。そもそも、その実装は FORTRAN からくる制約などに縛られている。[http://www.lysator.liu.se/c/num-recipes-in-c.html]

SLATEC は 1970 年代に米 DoE (Department of Energy) のプロジェクトにより FORTRAN で書かれた著作権フリー (public domain) のサブルーチン集である。コードは枯れていて設計も (その時代としては) よい。GSL は SLATEX の近代的な実装となることも想定している。

NSWC (Naval Surface Warfare Center numerical library) は著作権フリーの大きな FORTRAN ライブラリで、質の良いコードが数多く収録されている。しかし解説文書が非常に少なく、現在ではマニュアルのハードコピーがごく少数流通しているだけである。

NAG および *IMSL* は高品質の商用ライブラリで、ソースは非公開である。NAG の方がより先進的で網羅的である。IMSL は簡便で、変数のサイズを示す引数を導入することで「デフォルト引数」的な機能を実現している。

ESSL および *SCSL* はそれぞれ、IBM と SGI が販売している商用ライブラリである。

Forth Scientific Library [http://www.taygeta.com/fsl/sciforth.html 参照]. Forth を使っている人にはいいかもしれない。

Numerical Algorithms with C は G. Engeln-Mullges と F. Uhlig によるテキストの付録として ANSI C で実装されたライブラリである。ソースコードは入手できるが、ライブラリはフリーソフトウェアではない。

NUMAL (A C version of the NUMAL library) は H.T. Lau が自著のディスク付き書籍 "A Numerical Library in C for Scientists and Engineers" のために書いたライブラリである。ソースコードは入手できるが、ライブラリはフリーソフトウェアではない。

C Mathematical Function Handbook は Abramowitz と Stegun による "Handbook of Mathematical Functions" の function approximations の節を Louis Baker が実装したものである。ソースコードは入手できるが、ライブラリはフリーソフトウェアではない。

CCMATH は Daniel A. Atkinson が実装した、GSL と同様の範囲をカバーした C 言語の数値計算ライブラリである。コードは非常に簡潔である。当初は GPL で公開されていたが、その後 LGPL に変更されてしまった。

CEPHES C 言語で書かれた便利で高品質な特殊関数のルーチン集。GPL ではない。

WNLIB Will Naylor による小規模な C 言語の数値計算ライブラリ。著作権フリーである。

MESHACH C 言語で行列・ベクトル演算を網羅的に実装した、線形代数のライブラリである。自由に利用できるが、GPL ではない (商用利用に制限がある)。

CERNLIB は CERN によって長年開発されてきた大規模で高品質な Fortran ライブラリである。元々はフリーソフトウェアではなかったが、後になって GPL になった。

COLT は CERN で Wolfgang Hoschek が開発した、Java で書かれたフリーの数値計算ライブラリである。BSD スタイルのライセンスで公開されている。

将来的に *GNU Scientific Library* は、数値計算の研究者 (やその学生) に貢献してもらえるようなフレームワークとなることを目標としている。

2 開発への参加

この設計指針は元々は 1996 年に書かれたものである。開発者としては 2004 年現在、GSL の機能は一通り揃っていると考えている。これから新しい機能を追加しようとはあまりしていない。

現在は、ルーチンの動作をより安定にし、整合性を取り、いくつか残っている問題を整理し、報告されているバグを修正することに重点が置かれている。GSL の CVS レポジトリにある 'BUGS' に記入してあるバグについて、詳細の調査や修正案を考えることで GSL の開発に参加してくれればありがたい。

ある程度の量のコードを新たに追加すると、その部分だけ GSL の他の各ルーチンのコードと「枯れ具合」が大きく異なってしまう。ライブラリ全体の安定性を管理するため、新しい機能は (たとえば Perl の CPAN や TeX の CTAN のように) パッケージ として、GSL 本体と並べて置くことにし、各人で管理してもらいたい。

2.1 パッケージ

GSL は、既存のライブラリを単にリンクするだけで拡張機能として利用できるように実装されている。以下に、別のライブラリの乱数発生器を利用することを例として挙げる。

```
$ tar xvfz rngextra-0.1.tar.gz
$ cd rngextra-0.1
$ ./configure; make; make check; make install
$ ...
$ gcc -Wall main.c -lrngextra -lgsl -lgslcblas -lm
```

以下に、パッケージの実装の仕方を示す。考え方としては、利用方法が混乱しないようい GSL の実装と整合性がとれていて、将来的には GSL の一部として、枯れたパッケージとして公開できるようにするということである。

- この文書に示す GSL と GNU のコーディング規約にしたがう。

  これは Automake のような GNU のパッケージ・ツールを利用し、TeXinfo 形式で文書化し、テスト環境 (test suite) を用意するということである。'make check' で GSL のテスト関数を使ったテストが実行されるようにしておき、その結果を PASS: または FAIL: で表示するようにする。パッケージは一般にあまり大きくなく、スタティックにリンクされるようにしていてもあまり問題はないと思われるので、libtool を使うことは必ずしも要求される訳ではない。

- 他とぶつからない、独自のプレフィックスをつける。

  'gsl_' は GSL の内部で利用するため、これを使ってはいけない。たとえば乱数発生器なら、rngextra といったものをつける。

  ```
  #include <rngextra.h>

  gsl_rng * r = gsl_rng_alloc (rngextra_lsfr32);
  ```

- 開発段階を示す、わかりやすいバージョンをつける。

  普通、0.x は動作を保証できないアルファ版につけるバージョンである。そして 0.9.x がベータ版で、おおよそ完成しているが、細かい修正やバグフィックスを行うためのバージョンである。その後の最初のリリース (メジャーリリース) 版が 1.0 である。バージョン 1.0 は API の修正がもう必要なくなってからつけるべきである。

メジャーリリース後は API を変更してはいけない。また (バグフィックスを除いて) 後方互換性を保つべきであり、リリースされているコードの修正はないはずである。API はすべてライブラリの外から参照でき、またパッケージ特有のデータ形式も構造体 `struct` として外から参照できるようにする。パッケージが提供する API の変更が必要になったときは、新しくメジャーリリースをやりなおすべきである (バージョン `2.0` などとして)。

- GNU General Public License (GPL) で公開する。

  将来 GSL に含まれるようにするためには、パッケージのライセンスを GSL にするように調整しておかねばならない (see ⟨undefined⟩ [著作権について], page ⟨undefined⟩)。

パッケージのリリース・アナウンスを `gsl-discuss@sourceware.org` に投稿してくれれば、GSL のウェブページに載せる。

安全保安の意味から、パッケージには GPG で署名をつけておいてほしい (`gpg --detach-sign` *`file`*)。

例示したパッケージ ‘`rngextra`’ には二つの乱数発生器が実装されているが、`http://www.network-theory.co.uk/download/rngextra/` で公開されている。

3 実装指針

3.1 実装に用いる言語

C 言語のみ。

シンプルで、非常に幅広く用いられているから。

3.2 他の計算機言語とのインターフェイス

ラッパーは外部 (extra) パッケージとして、ラッパーが提供され、GSL の一部とはしない。各管理者が各自で管理するものとする。

ラッパーの実装には、swig, g-wrap を使うものとする。

3.3 実装するルーチン

既存のライブラリにあるものは実装対象とする。重要な分野の関数ほど早期に実装するものとする。

3.4 実装しないルーチン

- GPL で公開されている高品質なライブラリですでに利用できる機能。
- 大きすぎるもの。たとえばサブルーチンではなく、一つのアプリケーションであるようなもの。

  例としては、偏微分方程式の数値解を求めるような機能は、非常に大きなプログラムになりがちであり、また多くの場合、解く問題に特化したルーチンになってしまう (PDE ソルバーは非常に多くのアルゴリズム、解の種類、グリッドの種類がある)。この種のものは、GSL 本体とは分けておく。そういった機能を利用したいときは、既存の適切なアプリケーションを使うべきである。
- 分けて使った方がよいもの。

  日付と時間をあつかう機能、金融関係の機能などを「科学技術計算」のライブラリに収録するかどうかは、議論の分かれるところである。しかしこういった機能は他のプログラムで GSL とは独立に利用しても問題ないため、GSL とは分けることとする。

3.5 基本的な設計の指針

数値計算ライブラリを書く時、完全性 (completeness) と簡潔さ (simplicity) を両立させることは非常に難しい。完全性とは、どれだけ幅広い問題に適用できるかであり、そしてそのライブラリが「閉じている (closed)」ことである。数学的には問題の数は無限である。たとえば、一次元空間内でベクトルに関して導関数を計算することも、逆にベクトル空間内で導関数を (ある経路に沿って) 一変数で計算することもできる。

数値計算ライブラリとしては、そういった可能性をできるだけ尽くすには、機能を一つ一つ実装して追加していくしかない。そう考えておけば、追加する必要のある機能は常に一つしかない、と言える。

壮大な完成予想図を前にして「可能な限りあらゆる数学のデータ構造と演算を C 言語の構造体で実装したい」といきなり考えるのは無謀であり、たいていは失敗する運命にある。C 言語などで実装が可能な程度の複雑さ、というものがある。数学一般の複雑さを C 言語で実

装しようとしても、泥沼にハマり、管理できなくなるのがオチだろう。時間を気にしなくていいなら、そうする手もあるが。

一般に、完全性を目指すよりも簡潔さを重視した方がよい。GSL の一部として新しい機能を設計する場合は、可能な限りモジュールの独立性を保つこととする。モジュール間の相互依存性が生じそうな場合は、その線引きをどこにするのか、明確にせねばならない。

3.6 コードの再利用

GSL から、特定の関数のソースファイルだけを抜き出して、他のプログラムにインクルードするだけで使うことができれば、非常に便利である。適切と思われる場合には、このような再利用が可能となるように配慮するものとする。その場合、コンパイル時にたとえば `GSL_ERROR` のようなマクロを定義する必要が生じるかもしれないが、それは構わない。乱数発生器の例を参照のこと。

3.7 コーディング規約

プロジェクトの開始に当たっては、コーディング規約を定めるべきである。GSL では、

- GNU のコーディング規約 (coding standard)
- ANSI C 標準
- `glibc` (GNU C Library) のコーディング規則 (coding convention)
- `glib` (GTK のライブラリ) のコーディング規則

にしたがっている。

これらの規約は、GNU の文書 *GNU Coding Standards* および *GNU C Library Manual* (version 2)、Harbison と Steele の *C: A Reference Manual*、および Glib のソースで参照できる。

数式は常に Abramowitz と Stegun による *Handbook of Mathematical Functions* を参照している。これは決定版といえる本であり、著作権フリーになっている。

GNU Scientific Library プロジェクトになにか哲学があるとすれば、それは「C 言語で考える (think in C)」である。GSL は C で実装されているので、GSL の開発では C 言語で無理なくできることをやっており、他言語の機能をシミュレートするようなことはしない。そういった機能を使うことは GSL の開発においては避けられている。避けることによってある機能を GSL の外部に出すことになったり、あるいは GSL 内に実装する機能が制限されてしまう場合は、あえてそうしている。実装が不必要に複雑になることに、価値は見いだされない。C 以外の言語でも数値計算ライブラリは提供されており、その言語の機能を利用したい場合に C 言語で実装されたライブラリを無理に使ってもいいことはない。

また、C 言語の処理系は「マクロ・アセンブラ」であることに常に留意せねばならない。やっていることがちょっと複雑になってきたな、と思ったら、「同じことをマクロ・アセンブラで書く気になるか？」と自問自答してみてほしい。躊躇するようなら、それは GSL に取り入れるべきことではない、ということである。

以下の論文を読むことを勧める。

- Kiem-Phong Vo, “The Discipline and Method Architecture for Reusable Libraries”, Software - Practice & Experience, v.30, pp.107–128, 2000.

この論文は `http://www.research.att.com/sw/tools/sfio/dm-spe.ps` あるいは technical report Kiem-Phong Vo, "An Architecture for Reusable Libraries" として `http://citeseer.nj.nec.com/48973.html`. で見ることができる。

また移植性の高い C ライブラリの設計に関する論文が、Vmalloc、SFIO、CDT について書かれている。

- Kiem-Phong Vo, “Vmalloc: A General and Efficient Memory Allocator”. Software Practice & Experience, 26:1–18, 1996.

  `http://www.research.att.com/sw/tools/vmalloc/vmalloc.ps`

- Kiem-Phong Vo. “Cdt: A Container Data Type Library”. Soft. Prac. & Exp., 27:1177–1197, 1997

  `http://www.research.att.com/sw/tools/cdt/cdt.ps`

- David G. Korn and Kiem-Phong Vo, “Sfio: Safe/Fast String/File IO”, Proceedings of the Summer ’91 Usenix Conference, pp. 235-256, 1991.

  `http://citeseer.nj.nec.com/korn91sfio.html`

ソースコードは GNU のコーディング規約にしたがって、タブではなくスペースでインデントされなければならない。`indent` コマンドを使う場合は以下のようになる。

```
indent -gnu -nut *.c *.h
```

この例の `-nut` オプションで、タブがスペースに変換される。

3.8 実装の準備

実際に何か実装する前に、目標をはっきりさせておいてほしい。長い目で見ると、これがもっとも時間を節約することになる。これには二段階の大事なステップがある。

1. フリーのライブラリ (GPL かその互換ライセンスの) がすでにあるかどうかを見極める。もしあるなら、新たに実装し直すことはない。Netlib、GAMs、na-net、sci.math.num-analysis に加えて、web 上でよく探してみるべきである。その過程で商用ソフトウェアについてもリストを作れるだろう。そのリストはステップ 2 で使う。
2. 既存の商用、あるいはフリーの、機能の競合するライブラリについての網羅的サーベイを行う。各ライブラリの API の形式、プログラムやサブルーチン間の通信方法を調べ、実装しようとしている/していない機能についてよく理解して、各ライブラリを実装している/していない機能によるトレードオフがどうなっているか、で分類する。既存のライブラリに付属する文書はいい参考資料なので目を通す。
3. 実装の目標を研究し、最新の実装には何が必要とされるかを決める。最新のレビュー記事を探す。下記の雑誌には少なくとも目を通すべきだろう。
   - ACM Transactions on Mathematical Software
   - Numerische Mathematik
   - Journal of Computation and Applied Mathematics
   - Computer Physics Communications
   - SIAM Journal of Numerical Analysis
   - SIAM Journal of Scientific Computing

GSL は、研究プロジェクトではないことにも留意してもらいたい。新たにアルゴリズムを考案する必要性を生じずにいい実装を行うのは、難しいことである。GSL の開発においては、既知のアルゴリズムが使える場合にはそれを実装することにしている。細かい改良は構わないが、それに時間を割くべきではない。

3.9 アルゴリズムの選択

可能な限り、スケーラビリティがあって、数値が漸近的な挙動をするような場合を制御できるアルゴリズムを選択する。特に整数引数の場合に重要である。関数の定義において計算量が $O(n)$ のシンプルなアルゴリズム、たとえば Abramowitz & Stegun に多数載っている再帰的な定義を使いたくなるが、これは $n = O(10 - 100)$ のような場合にはいいが、その関数を $n = 1000000$ に対して使おうとすると問題が生じる。

同様に多変量データに対して、各要素が同じ大きさである、または O(1) であるようにスケーリングされているというような仮定をおいてはならない。実装するアルゴリズムの方で、適切な (|x| そのものではなく、スケーリングのための係数行列を D とするとき |Dx| で表されるような) ノルムを使って、スケーリングや平衡化 (balancing、変数値の桁を近づけること) を行うべきである。

3.10 文書化

文書化: プロジェクトの管理者/統括者は、文書化の例を示すべきである。質の高い文書化が要求されるため、解説文書では、何についての文書であるかを明示し、パッケージが提供する機能についての参考文献を挙げる。実装するそれぞれの各機能について、文献を挙げるようにする。入門書的な文書 (チュートリアル) は、必須ではない。

文書の作成には、フリーソフトウェアを、たとえばグラプのプロットには GNU plotutils のようなものを使うものとする。

プロットには gnuplot を用いることもあるだろうが、これは真の意味でのフリーソフトウェアではない。また、他の商用ソフトウェアによるプロットもあるかもしれないが、そういったものは GNU plotutils で置き換えるべきである。

参考文献には、その分野で広く知られている決定版で、もっともよい書籍を挙げ、知名度の低い教科書や入門書 (大学学部の教科書類) は避けるものとする。たとえば、アルゴリズムの参考文献ならクヌース (Knuth)、統計ならケンドールとスチュワート (Kendall & Stuart)、特殊関数ならアブラモウィッツとステグン (Abramowitz & Stegun) による Handbook of Mathematical Functions AMS-55、などである。アブラモウィッツとステグンの本は著作権フリー (public domain) なので誰でも自由に参照できる。可能な限りこれを参照するものとする。

これらの書籍が参考文献であれば、いつでも自由に参照することができる。もしオンラインで閲覧できず、書籍として刷られているものを購入せねばならないとしても、これらの文献は非常に質が高く、また他の多くの文献と比べて GSL マニュアルの内容をもっとも幅広くカバーするものと考えられる。参照すべき文献が多数にわたっているような場合、アルゴリズムの詳細を知ろうするとコストがかかるだけでなく、非効率的でもある。教科書類は数年で増刷されなくなることがあるが、参考文献には、長く発行され続けるものを選ぶ。

同様に、可能な限りオリジナルの論文を挙げる。実装の際には、実際にその論文を参照しながら (たとえばバグの修正作業時など)、将来的に開発者が交代する可能性を想定して開発を行うこと。

文献のたどっていく上で困ったことがあったら、`gsl-discuss` メイリング・リストに投稿してほしい。論文や別刷りを閲覧できる図書館に行ける人たちが対応できるかもしれない。

Texinfo 形式で数式を書くときは、`@math` コマンドを使えば、その中で いくつかの TeX の命令が使える。以下の例のように、数式モードは自動的に `$...$` で囲まれる。

```
to calculate the coefficient @math{\alpha} use the function...
```

上の例は、オンラインにする形式でも TeX 形式でも正しく整形される。

{ と } は TeX と texinfo の両方で特殊文字として扱われるので、@math コマンドの中で使うと不都合が生じる。\sqrt{x+y} といった式を書きたい場合には不便を強いることになる。

これは TeX の命令の前に texinfo のマクロ @c を置き、数式を英数字で模式的に表したもの (波カッコは @{ と @} で書ける) を @math コマンドの中に書くことで回避できる (書ける TeX の命令には特に制約はない)。こうすると TeX の命令は TeX の入力になり、@math の中に書いたことは info 形式に整形される。

@c{} の後に書かれていることは texinfo 形式ではまったく無視されるため、これは行の終わりにおかねばならない。これは texinfo における「コメント」とは異なっており、このコメントコマンド @c は、その直後の @math コマンドの出力を TeX の命令と置き換える機能がある。一般的な意味でのコメントには @comment コマンドを使う。

例を挙げると、

```
this is a test @c{$\sqrt{x+y}$}
@math{\sqrt@{x+y@}}
```

これは plain TeX で this is a test $\sqrt{x+y}$ と書き、info で this is a test @math{\sqrt@{x+y@}} と書くことになる。

これは、分かりにくい TeX の命令を、info では C 言語風に英数字で書くのにも使える。

```
for @c{$x \ge y$}
@math{x >= y}
```

上の例は TeX でも info でも分かりやすい形式で表示されるだろう。

3.11 名前空間

ライブラリの外から参照できる関数、変数にはすべて gsl_ をつける。

ライブラリの外から参照できるマクロにはすべて GSL_ をつける。

ライブラリの外から参照できるヘッダファイルのファイル名にはすべて gsl_ をつける。

インストールされるライブラリには、libgslhistogram.a のような名前を付ける。

インストールされる実行ファイル (ユーティリティなど) にはすべて gsl- (ハイフン。下線ではない) をつける。

関数名、変数名はすべて小文字、マクロとプリプロセッサの変数は大文字とする。

他にもいくつか、変数、関数に関する命名規則がある。

p1 プラス 1 (+1)。関数 log1p(x) や変数 kp1 ($= k + 1$) など。

m1 マイナス 1 (-1)。関数 expm1(x) や変数 km1 ($= k - 1$) など。

3.12 ヘッダーファイル

インストールされるヘッダファイルは、複数回インクルードされてもエラーが出ないようにしておかねばならない。一般的には以下のようにプリプロセッサ・マクロを定義して、条件付きコンパイルをすることでそれができる。

```
#ifndef __GSL_HISTOGRAM_H__
#define __GSL_HISTOGRAM_H__
...
#endif /* __GSL_HISTOGRAM_H__ */
```

3.13 対応するプラットフォーム

C 言語の標準ライブラリを備えた ANSI C 処理系、IEEE 準拠の数値演算を行うプラットフォームを対象とする。

3.14 関数名

各モジュールには名前を付け、それをモジュール内の関数名のプレフィクスとする。たとえばモジュール名が gsl_fft なら関数名は gsl_fft_init、という具合である。GSL のソースツリー上では、各モジュールは各サブ・ディレクトリに対応する。

3.15 オブジェクト指向

アルゴリズムの実装はオブジェクト指向であるべきだが、それは ANSI C で容易に実装でき、移植性の高いものでなければならない。「継承」機能を実装するための型変換などのテクニックは使うべきではなく、プログラム作成時にそういったことに配慮する必要が生じることは、避けなければならない。そうすると多くの実装法が使えないことになるが、それらは GSL に実装するには複雑すぎる。

関数へのポインタを使うことで、C 言語でも抽象概念的なクラスを定義することができる。GSL ソースの rng ディレクトリを参照されたい。

著作権フリーの FORTRAN コードを再実装しようとする場合、データ構造として配列を使うのではなく、構造体を適切に定義してオブジェクトを表すようにしてほしい。構造体は、その利用が一つのファイル内で完結していて外部から参照できない場合は、便利に使うことができる。

たとえば以下の関数を繰り返し呼び出す FORTRAN プログラムがあったとする。

```
SUBROUTINE  RESIZE (X, K, ND, K1)
```

ここで X(K,D) が二次元配列で、そのサイズを X(K1,D) に変更したい場合、以下のように構造体を使えばより分かりやすくプログラミングできる。

```
struct grid {
    int nd;  /* number of dimensions */
    int k;   /* number of bins */
    double * x;   /* partition of axes, array of size x[k][nd] */
}

void
resize_grid (struct grid * g, int k_new)
{
...
}
```

同様に、一つのファイル内で同じコードが何度も出てくる場合、それをスタティックな関数、あるいはスタティックなインライン関数として定義するとよい。そうするとコンパイラが型チェックを行うようになり、すべて展開した形で書くよりも手間が軽減される。

3.16 コメント

GNU のコーディング規約にしたがう。その中の対応箇所を以下に引用する。

“各文は完結していて、文頭の単語は大文字とする。しかし小文字の識別子が文頭に来る場合は、それを大文字にしてはならない。一文字でも大文字にすると、異なる別の識別子になってしまう。文頭が小文字になるのを避けたいときは、語順の異なる文にすればよい (たとえば "The identifier lower-case is ..." といったように)。”

3.17 無駄のない構造体

構造体の定義は「最小限」が望ましい。たとえば対象としている問題を解くアルゴリズムが複数がある場合 (たとえば導関数を使う方法と使わない方法など)、各場合に対応するそれぞれの構造体を定義するのがよい。実行時に自動的に判別するような機能は望ましくない。

3.18 アルゴリズムの分割

繰り返し計算を行うアルゴリズムは、初期化、繰り返し計算の 1 ステップ、判定、の各部に分けて、繰り返しの状況を監視 (出力)、制御できるようにすべきである。コールバックやフラグによる判定よりも、状況を監視する方がよい。コールバックが必要に思えるときは、アルゴリズムをそれぞれ独立したコンポーネントにさらに分解できるかもしれない、ということである。分割することでアルゴリズム全体を制御できるようになる。

たとえば実時間で進行するプロセスと同期して微分方程式の数値解を求めるとき、求解のステッピングを制御する必要があるだろう。そのためには、繰り返し計算のアルゴリズムがステップ単位に分割されていなければならない。高レベルな求解ルーチンではこのような柔軟な使い方はできない。

3.19 メモリの確保と管理

関数内でヒープ上にメモリを確保する場合、_alloc を関数名の末尾につける (たとえば gsl_foo_alloc のように)。確保したメモリを解放する関数は、末尾に _free をつける (gsl_foo_free のように)。

オブジェクトの初期化が完全には行われず、エラーを返すような場合には、初期化の過程で確保したメモリをすぐに解放せねばならない。

関数の中で、一時的な用途でメモリを確保してはならない (たとえ return する前に解放するとしても)。関数を呼び出す側でメモリの管理ができないからである。すべてのメモリについて、確保と解放がそれぞれの関数で行われ、その引き渡しを "workspace" 引数を通じて行うべきである。そうすればループの中で、何度もメモリの確保と解放が行われてしまうようなことを避けることができる。

確保したメモリがどのオブジェクトのものなのかをはっきりさせておくために、workspaces 構造体は他の構造体オブジェクトとメモリを共有したり、内部に他の workspaces オブジェクトを持つようなことをしてはならない。様々な状況でも簡便に使えるようにするため、workspace オブジェクトのメモリ確保は整数引数で行われるべきであり、他の構造体の要素を使って行うような方法は好ましくない。

3.20 メモリ配置

GSL では行列やベクトルは一次元のフラットなメモリ・ブロックに保持される。C 言語のポインタ、あるいはポインタへのポインタの配列ではない。行列は、行指向でメモリに格納される。たとえば、列番号をひとつづつたどると、メモリ上で一つずつ動いていくことになる。

3.21 線形代数演算のレベル

線形代数演算を行う関数は、以下の 2 レベルに分けられる。

"1d" の関数は C 言語の普通の引数 (double *, stride, size) を受け取り、普通の C プログラムで簡便に利用できる。gsl_vector を扱う機構を使う必要はない。

これは、学習曲線を最小化するためである。何かの理由でこの種の関数を、たとえば FFT など、一つだけ使う必要が生じたような場合に、gsl_vector の扱い方を調べる必要なく、手軽に利用できる。

すると、行列では同じようにしないのか？という疑問を生じるかもしれない。行列で同じことをしようとすると、引数リストが各行列について (size1, size2, tda) のように長くなって混乱しがちであり、行と列のサイズを取り違えやすい。この場合は gsl_vector と gsl_matrix を使ってプログラム側で把握しておくようにするのが、むしろよい。

そういうわけで GSL の線形代数機能として、上の二つのレベル (C 言語配列の 1d 演算、および高レベルの gsl_matrix と gsl_vector を使う演算) が用意されている。

低レベルの関数と同等の、高レベルのベクトル演算を行う関数を定義するのは可能だが、それは GSL 開発においてさほど重要な問題ではないと考えて、行っていない。また引数に v->data、v->stride、v->size と書き並べるよりも C 言語の普通の引数を書く方が簡便である。低レベル関数のベクトル版の実装は、利便性の向上にはなるかもしれない。

計算効率を重視するので、可能なら BLAS のルーチンを内部で使うようにする。

3.22 誤差推定

特殊関数の実装では、生じうる誤差の最大値をガウシアン誤差の 2 倍、たとえば 2-sigma として計算している。したがって解の真の値がその範囲に入っている確率が 98% ということである。その範囲は、計算結果 +/- エラーの値、である (1 sigma が 32% とは限らない)。計算結果の分布が必ずしも正規分布になるとは限らないが、この方法は現実的にはうまくいっている。

3.23 例外とエラー処理

基本的には、エラーが発生したらエラーコードを返すようにする (‘gsl_errno.h’ にエラーコードのリストがある)。エラーを示すには GSL_ERROR を使う。現状ではこのマクロの定義は理想的とは言えないが、コンパイル時に変更することができる。

エラーの発生時には単にエラーコードを返すだけではなく、GSL_ERROR マクロを使うべきである。それにより、デバッグ時にプログラム側でエラーをトラップできるようになる (gsl_error 関数にブレークポイントを設定することでそれができる)。

返り値が単なるエラーではなく、状況を示すような場合は、GSL_ERROR は使わない方がよいことがある。たとえば対話的に入力を求めるルーチンで、繰り返し計算が成功したかどうかを入力の前に表示するような場合である。一般的な繰り返し計算アルゴリズムでは、「失敗」(返り値は GSL_CONTINUE) を返すことがプログラムの実行中ではほとんどであり、そういった場合には GSL_ERROR を使う必要はない。

エラーが生じたときには、確保していたメモリを解放するようにしなければならない (特に、初期化処理中にエラーが発生した時)。

3.24 オブジェクトの保存と読み込み

メモリ・ブロックを使うようなオブジェクト (ベクトル、行列、ヒストグラムなど) を定義する場合、そのメモリ・ブロックを読み書きする関数を用意するとよい。

```
int gsl_foo_fread (FILE * stream, gsl_foo * v);
int gsl_foo_fwrite (FILE * stream, const gsl_foo * v);
int gsl_foo_fscanf (FILE * stream, gsl_foo * v);
int gsl_foo_fprintf (FILE * stream, const gsl_foo * v, const char *format);
```

書き出す関数は、メモリ・ブロックの内容を単にダンプするだけで、ブロックのサイズなど他の情報は何も出力しない。これは、これらの関数を使って高レベルの入出力ルーチンを書けるようにするためのものである。fprintf/fscanf 形式のものは、アーキテクチャの異なるプラットフォームでも正しく動作し、他のものはバイナリ形式で入出力する。それらの関数を実装するのには、以下の関数を使うとよい。

```
int gsl_block_fread (FILE * stream, gsl_block * b);
int gsl_block_fwrite (FILE * stream, const gsl_block * b);
int gsl_block_fscanf (FILE * stream, gsl_block * b);
int gsl_block_fprintf (FILE * stream, const gsl_block * b, const char *format);
```

または

```
int gsl_block_raw_fread (FILE * stream, double * b, size_t n, size_t stride);
int gsl_block_raw_fwrite (FILE * stream, const double * b, size_t n, size_t stri
de);
int gsl_block_raw_fscanf (FILE * stream, double * b, size_t n, size_t stride);
int gsl_block_raw_fprintf (FILE * stream, const double * b, size_t n, size_t str
ide, const char *format);
```

これらの関数を、実際のファイル入出力に使うことができる。

3.25 返り値の取り扱い

関数内では、呼び出した関数から返ってくる値は、変数を用意してそれに代入することとする。これによりデバッグがやりやすくなり、後になって関数の動作を調べたり修正したりするのが容易になる。その変数がごく限られた範囲でしか必要とならないようなときは、適切なスコープを作る。

例えば、

```
a = f(g(h(x,y)))
```

と書く代わりに、返ってきた値を保存する変数を用意して、

```
{
  double u = h(x,y);
  double v = g(u);
  a = f(v);
}
```

とする。こうするとデバッガで値を確認するのが容易になり、ブレークポイントもより細かく設定できる。このように書いたプログラムでも、コンパイル時に最適化オプションをつければ、一時的にしか使われない変数は自動的に省略される。

3.26 変数名

変数名には、以下の命名規約がある。

`dim`	次元数
`w`	workspace へのポインタ
`state`	状態を表す変数へのポインタ (文字数を少なくしたいときは `s` とする)
`result`	計算結果へのポインタ (関数の「出力」となる)
`abserr`	絶対誤差
`relerr`	相対誤差

epsabs	絶対誤差の許容範囲
epsrel	相対誤差の許容範囲
size	配列やベクトルの大きさ、たとえば double array[size] のような具合
stride	ベクトル要素の刻み幅
size1	行列の行数
size2	行列の列数
n	一般の整数、たとえば FFT におけるデータ配列の要素数など
r	乱数発生器 (gsl_rng)

3.27 データ型のサイズ

ANSI C の int 型は最低 16 ビットあることしか保証されない。処理系によってはそれ以上のサイズがあることもあるが、保証はない。したがって 32 ビットの整数を使いたい場合は long int を使わねばならない。そうすれば 32 ビット以上であることが保証される。実際には、多くのプラットフォームで int は 32 ビットであるが、GSL は ANSI C 準拠でなければならず、特定のプラットフォームに依存してはいけない。

3.28 size_t

すべてのオブジェクト (メモリのブロックなど) の大きさは、size_t を単位とするべきである。したがって for(i=0; i<N; i++) のような繰り返し処理のインデックスも size_t とするべきである。

int と size_t ははっきりと使い分けなければならない。相互に代入する事はしてはならない。

カウンタ (インデックス) の値を減らして行くようなループ処理をしたい場合は、size_t は符号なし整数であるため、以下のように

```
for (i = N - 1; i >= 0; i--) { ... } /* i が負になったらループを抜ける */
```

とする代わりに、

```
for (i = N; i-- > 0;) { ... }
```

として、i=0 よりも値が小さくなることを避けなければならない。デクリメント演算子を後置にすることで、条件確認は i が 0 になるまえに行われる。しかしループから抜けたときには i は 0 から 1 回デクリメントされている (符号なし整数については i>0 は i!=0 と同じであるため、この例は for (i = N; i--;) とも書ける)。

もっとはっきりと混乱を避けるためには、ループのカウンタと逆に増減する変数を用意するとよい。

```
for (i = 0; i < N; i++) { j = N - i; ... }
```

注 (BJG). わたしの勧める書き方は元々、

```
for (i = N; i > 0 && i--;) { ... }
```

として明示的に i>0 を確認しループから抜けたときに i=0 となるようにすることであった。しかしこれはループを展開するときに分岐が余計にできてしまい、実行速度が遅くなるやり方である。これは J. Seward の私的による。彼に感謝する。

コーディング・スタイルの問題ではあるが、インクリメント、デクリメントに関してはデフォルトで後置演算子 (i++ と i--) を使い、特に必要のある場合に限って前置演算子 (++i と --i) を使うようにしてもらいたい。

3.29 配列とベクトル

関数の引数としては、ポインタでも配列でもどちらでも宣言できる。C 言語の標準では、これらは同じものである。しかし使い方を明確に分けて、ポインタは渡した関数の内部で変更される可能性のあるオブジェクトを、配列は刻み幅が 1 の一組のオブジェクトを表すのに使うのが便利である (配列が渡した先で変更されるかどうかは、const の有無による)。ベクトルの場合は刻み幅は 1 である必要はなく、ポインタ形式の方が好ましい。

```
/* 計算結果を返すのに使う実数変数の場合 */
int foo (double * x);

/* 変更されうる実数ベクトルの場合 */
int foo (double * x, size_t stride, size_t n);

/* 変更されない実数ベクトルの場合 */
int foo (const double * x, size_t stride, size_t n);

/* 変更されうる配列の場合 */
int bar (double x[], size_t n);

/* 変更されない配列の場合 */
int baz (const double x[], size_t n);
```

3.30 ポインタ

式の右辺でポインタの間接参照演算子 (dereference operator) を使うことはできるだけ避けなければならない。それには一時的に変数を使うのがよい。そうするとコンパイラによる最適化も行われやすく、間接参照演算子と乗算のどちらの意味かで混乱するようなことも減らせる。以下の例、

```
while (fabs (f) < 0.5)
{
  *e = *e - 1;
  f  *= 2;
}
```

の代わりに、

```
{
  int p = *e;

  while (fabs(f) < 0.5)
    {
     p--;
     f *= 2;
    }

  *e = p;
}
```

とするのが望ましい。

3.31 Const

関数のプロトタイプ宣言で、変更されないことがはっきりと分かっているオブジェクトを指すポインタには `const` をつけることとする。また、関数内あるいは特定のスコープ内で定数として扱われる変数にも `const` をつけることとする。これにより、定数として扱われるものの値 (たとえば配列のサイズなど) が予期せず変更されてしまうことを防ぐことができる。コンパイラによる最適化も行われやすくなる。値渡しの引数についても、定数として扱われるものについては同様である。

3.32 疑似テンプレート

'`templates_on.h`' と '`templates_off.h`' に、擬似的にテンプレート機能を実現しているマクロがある。ソースツリーのディレクトリ '`block`' に使用例があるので参照されたい。なにか悪い夢に出てきそうなマクロになっているが、場合によってはこういう手法を使わざるを得ないこともある。あまり多用しないようにしてほしい。

テンプレートは「データ」(ベクトル、行列、統計、ソート) の操作にしか使わない、という規約を設けている。これは、なにか特定の一つの型で表される外部データを扱うプログラムを想定している。たとえば 8 ビットのカウンタから生成される巨大な文字配列などである。

他の関数はすべて、倍精度、単精度の浮動小数点実数、各種の整数型 (符号なし long int を乱数のために使うなど) を使うことができる。テンプレートを完備した開発を勧めている訳ではない。

それは "putting a quart into a pint pot" (不可能だと分かっていることをあえてやってしまうこと) である。まとめると、普通に使うのに適した「自然な型」を使うのが原則であって、テンプレートは、他のデータ型に対応する必要が出てくることに備えなければならないような、限られた場面を乗り切るために用意する、ということである。

浮動小数点実数については、`double` を使うのが自然であろう。こういった考え方が、C 言語の一部分をなしている。

3.33 恣意的な定数値

勝手な値の定数を使ってはならない。

たとえば、「小さな」値として '1e-30' や '1e-100'、`10*GSL_DBL_EPSILON` などといった値をハードコードしてはならない。こういったやり方は、ライブラリの汎用性を損ねる。

IEEE に規定されている演算を使って正確な値を導かねばならない。誤差が無視できない大きさになりそうなときは、解析的に導出されたアルゴリズムで適切に誤差の項の大きさを推定し、呼び出したプログラムに返すべきであり、憶測で値を返してはならない。

用心深く検討されたアルゴリズムでは、そういった恣意的な定数は必要ないことが多い。またはパラメータとしてプログラム側で設定できるようにする。

たとえば、以下の例を考える。

```
if (residual < 1e-30) {
   return 0.0;  /* residual は丸め誤差以下なので 0 とする */
}
```

このコードは、以下のように修正するべきである。

```
   return residual;
```

こうすることで、呼び出し側で residual の値について重視するかその必要はないかを判断することができるようになる。

GSL_DBL_EPSILON のような定数の使用が受け入れられるのは、関数の近似値を求める関数 (テイラー展開や漸近展開など) の場合だけである。こういった場合には、それは恣意的な定数ではなく、そのアルゴリズムに固有な部分ということである。

3.34 動作の検証

各モジュールを実装するにあたって、モジュールに含まれる各ルーチンの動作を検証するための、試験用のデータとツールを一式、用意しなければならない。

試験セットは実装されたライブラリを使い、既知の結果に対する再現性を確認し、または複数回実行することで結果のばらつきを統計的に解析するべきである (たとえば乱数発生器の場合など)。

理想的には、一つのディレクトリ (モジュール) に一つの試験セットがあり、モジュールのコード全体について試験、確認できるようにすべきである。しかし実際にそういった試験セットを作成するためには多大な労力を要する。そのため現実的には、重要な部分だけ動作を試験して、他の部分はコードを見て確認することになる。すべての種類のエラーについて、それが発生する状況を作って試験を行うこととする。パラメータの値が想定外のときに、エラーを発生させずに関数がリターンしてしまうような深刻な事態を避けるためである。特に、ヌル・ポインタを渡したときのテストを怠ってはならない。それは、セグメンテーション・フォールトを発生させなければならない状況だからである。

テストは確率的であってはならない (deterministic である必要がある)。GSL で用意してる gsl_test 関数を使って、モジュールの各機能についてそれぞれ、PASS か FAIIL かを出力させるものとする。これにより FAIL するのがどこかが特定できる。

「エントロピーの高い」、現実的なテストを行うようにしなければならない。0 や 1 のような単純な値についてだけテストを行っても、バグは洗い出されない。たとえば $x = 1$ という値でテストを行っても、x を乗じることを忘れているようなバグはわからない。同様に $x = 0$ でテストしても、x を含む項を加えるのを忘れているようなバグは分からない。そういったサイレント・バグを見つけるためには、たとえば 2.385 といった値を使うべきだろう。

複数の値でテストを行う場合は、それらの値の間に単純な関係が成り立っていないか、確認しておかねばならない。それらがキャンセルし合ってバグを見えなくしてしまう可能性がある。

実数の乱数をテストに使う必要があるときは、試験プログラム内で od -f /dev/random で乱数を発生させるとよい。

試験プログラムの出力に sprintf を使ってはならない。これを使うと、試験プログラム自体のバグが見つかりにくくなる。関数 gsl_test_... で文字列を引数として受け取れるようになっているので、これを使うこととする。

3.35 コンパイル

「きれいに」コンパイルできるように実装しなければならない。コンパイル・オプションで、各種のチェックを厳しく行うようにする。

```
make CFLAGS="-ansi -pedantic -Werror -W -Wall -Wtraditional -Wconversion
  -Wshadow -Wpointer-arith -Wcast-qual -Wcast-align -Wwrite-strings
  -Wstrict-prototypes -fshort-enums -fno-common -Wmissing-prototypes
  -Wnested-externs -Dinline= -g -O4"
```

checkergcc でスタックおよびヒープで問題が生じないかどうかを確認する。これはメモリ・チェック・ツールの中でもベストのものである。checkergcc が使えない場合は、ヒープのチェックには Electric Fence が使える。やらないよりはよい。

メモリアクセスのチェックには、valgrind という新しいツールもある。これも checkergcc 同様に使うべきである。

実装したライブラリは C++ コンパイラ (g++) でもコンパイルできるようにせねばならない。ANSI C で書いていれば、これは大した問題にはならないだろう。

3.36 スレッド・セーフ

GSL はスレッド・セーフなプログラムで使えなければならない。すべての関数がスレッド・セーフで、その意味で静的変数を使うべきではない。

すべの関数が完璧にスレッド・セーフでなければならない、というわけではないが、スレッド・セーフかどうかはっきりしないようなものは避けなければならない。たとえば、ライブラリ全体の挙動を制御するための大域変数 (範囲確認の On/Off、致命的エラー時に呼ばれる関数など) はあってよい。これらの変数はプログラム側から直接参照、操作できるため、マルチ・スレッドのプログラムを書いているときに、複数のスレッドで操作してしまうことが好ましくないことは明白である。

スレッドを明示的にサポートする機能 (たとえば排他制御機構など) を実装する必要はないが、マルチ・スレッド・プログラムから GSL ルーチンを呼べなくなってしまうようなことは避けなければならない。

3.37 著作権について

- GSL のためのコードを書いた人は、そのコードが GNU General Public License (GPL) にしたがって公開されることを了承しなければならない。場合によっては、雇用者からコードの著作権を放棄する旨の文書を書いてもらう必要があるかもしれない。
- コードとアルゴリズムについて、著作権を GSL プロジェクトに対して明確に示しておかねばならない。
- コードの著作権は、コーディングした人が保持したままにしてもいいし、FSF に譲渡しても構わない。

  以下に、GPL における標準的な著作権放棄の文例を載せておく (詳しくは GPL そのものを参照されたい)。状況に合わせて修正したり詳細を定めたりする必要があるだろう。

  ```
  Yoyodyne, Inc., hereby disclaims all copyright interest in the software
  'GNU Scientific Library - Legendre Functions' (routines for computing
  legendre functions numerically in C) written by James Hacker.
  (当社 Yoyodyne, Inc. は、James Hacker が書いたソフトウェア 'GNU Scientific
  Library - ルジャンドル関数' (ルジャンドル関数の値を数値的に計算する C 言語
  で書かれたルーチン) の著作権を放棄する。)

  <signature of Ty Coon>, 1 April 1989 (社長の署名と日付)
  Ty Coon, President of Vice (社長の名前と肩書き)
  ```

- フリーでないコードを使う、あるいは実装し直すようなことをしてはならない。

  分かりやすい例では、*Numerical Recipes* や *ACM TOMS* (ACM Transacntions on Mathematical Software) に載っているコードなどである。

  Numerical Recipes は厳しいライセンスの元での利用のみが許可されており、フリーなソフトウェアではない。この本の内容、載っているコード (関数名、変数名、数式の順序など) の著作権は版元の Cambridge University Press が保持している。どういう形であっても、GSL のコードは Numerical Recipes にならったり、それを元にしたりしてはならない。

TOMS に発表されたアルゴリズムはオンラインで公開されて入るが、著作権フリーではない。ACM は独自の非商用のライセンスをそれに適用しているが、そのライセンスは GPL と互換ではない。詳細は ACM Transactions on Mathematical Software に毎号記載されている。また ACM のウェブページでも確認できる。

使ってよいのは、フリーなライセンスで公開されていることがはっきり分かっているコードだけである。あるコードに、ライセンスのことが何も書いてなかったとしても、それはそのコードが著作権フリーであるということにはならない。フリーであることが明示されていなければならない。はっきりしない場合は、そのコードを書いた人に確認しなければならない。

- 数値解析の古い書籍に載っているアルゴリズムを参照するのは問題ない、と私は考える (BJG. コードが既存のもののコピーではなく、独立に実装されたものである場合に限る。たとえば ACM TOMS に載っている論文を読んで、それを参考しながらもコードは新たに、独自に書くといった場合である)。

3.38 非 UNIX 環境との互換性

GSL を非 UNIX 環境への移植は、検討する価値のあることである。DOS のことは無視していいだろう。windows95/windowsNT に移植することだけを考えればよい (ファイル名が長くなってもいいだろうから)。

一方で、積極的に非 UNIX なシステムに移植を進めるべきだ、というわけではない。

もっともよいのは、たとえば「絶対に必要という訳でなければ XYZ 機能を使ってはならい」というような、移植のためのガイドラインを作ることだろう。そうすれば、Windows の利用者が移植作業を行えるだろう。

3.39 他のライブラリとの互換性

GSL の開発においては、他の数値計算ライブラリとの互換性は考慮していない。

しかし現実的に他のライブラリ、たとえば Numerical Recipes などは幅広く利用されている。それらのライブラリ中の関数を簡単に置き換えられるような形のコードがあれば、便利であろう。そういったものが作られた場合、それは GSL とは別に管理、リリースされるラッパーとなるだろう。

それとは別に、BSD の数学ライブラリと、`expm1`、`log1p`、`hypot` いったその関数との互換性の問題もある。このライブラリは、ほとんどの (すべての、ではないが) システムで利用できる。

この場合、システムのベンダーが提供するライブラリよりも優れたコードを書くのが理想である。(たとえば、log1p では Intel x86 の CPU 命令を使っている)。また GSL では移植性を確保するために `gsl_hypot` などの関数を実装している。これは必要に応じて、autoconf によって自動的にシステムの hypot の予備として使われるようになる。詳しくは '`gsl/complex/math.c`' の中の `gsl_hypot` の定義の部分と、'`configure.in`' と '`config.h.in`' の該当部分を参照されたい。

3.40 並列化

GSL の開発においては、並列実行のサポートは考慮していない。並列計算のためには設計から全く違ったアプローチを取る必要があり、並列実行しないプログラムにとっては、実行時に無用のオーバーヘッドを生じるからである。

3.41 精度

数値の精度が限られていることによる切り捨てなどを行うときに、計算精度からなんからの値を決めて判断に使うことになるだろうが、そのときは GSL_DBL_EPSILON と GSL_DBL_MIN、またはこれらの組み合わせやこれらのベキ乗などでその値を決めるものとする。これにより、計算精度が異なるプラットフォームへの移植が容易になる。

3.42 その他

変数名に l を使ってはならない。これは数字の 1 と見分けにくいからである (昔の Fortran プログラムではよく用いられているようだが)。

最後に - 修正の必要なルーチンがたくさんあるより、完璧なルーチンが 1 つだけしかない方がマシである。

4 関連文献

4.1 数値計算一般

- *Numerical Computation* (2 Volumes) by C.W. Ueberhuber, Springer 1997, ISBN 3540620583 (Vol 1) and ISBN 3540620575 (Vol 2).
- *Accuracy and Stability of Numerical Algorithms* by N.J. Higham, SIAM, ISBN 0898715210.
- *Sources and Development of Mathematical Software* edited by W.R. Cowell, Prentice Hall, ISBN 0138235015.
- *A Survey of Numerical Mathematics (2 vols)* by D.M. Young and R.T. Gregory, ISBN 0486656918, ISBN 0486656926.
- *Methods and Programs for Mathematical Functions* by Stephen L. Moshier, Hard to find (ISBN 13578980X or 0135789982, possibly others).
- *Numerical Methods That Work* by Forman S. Acton, ISBN 0883854503.
- *Real Computing Made Real: Preventing Errors in Scientific and Engineering Calculations* by Forman S. Acton, ISBN 0486442217.

4.2 参考文献

- *Handbook of Mathematical Functions* edited by Abramowitz & Stegun, Dover, ISBN 0486612724.
- *The Art of Computer Programming* (3rd Edition, 3 Volumes) by D. Knuth, Addison Wesley, ISBN 0201485419 (邦訳がある).

4.3 専門書

- *Matrix Computations* (3rd Ed) by G.H. Golub, C.F. Van Loan, Johns Hopkins University Press 1996, ISBN 0801854148.
- *LAPACK Users' Guide* (3rd Edition), SIAM 1999, ISBN 0898714478.
- *Treatise on the Theory of Bessel Functions 2ND Edition* by G N Watson, ISBN 0521483913.
- *Higher Transcendental Functions satisfying nonhomogenous linear differential equations* by A W Babister, ISBN 1114401773.

複製と再配布

The subroutines and source code in the *GNU Scientific Library* package are "free"; this means that everyone is free to use them and free to redistribute them on a free basis. The *GNU Scientific Library*-related programs are not in the public domain; they are copyrighted and there are restrictions on their distribution, but these restrictions 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 these programs that they might get from you. (*GNU Scientific Library* パッケージに含まれるサブルーチンとソースコードは「フリー」である。これは、万人がこれを自由に使うことができて、これを自由に再配布することができる、という意味である。*GNU Scientific Library* とこれに関連するプログラムは、著作権フリーではない。著作権は維持されていて、その配布にも条件があるが、その条件は善意の協力者たちが行う作業の妨げとならないように設定されている。他の者がこれらのプログラムを入手しようとすることを、妨げようとしてはならない。)

Specifically, we want to make sure that you have the right to give away copies of the programs that relate to *GNU Scientific Library*, that you receive 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. (特に、*GNU Scientific Library* と関連するプログラムを所持している者はこれを他者に与える権利があり、入手を望む者には入手する権利があり、入手したプログラムを改編する、あるいは一部を取り出して他のプログラムに利用する権利があり、これらの権利を知る権利があることを明確にしておきたい。)

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 the *GNU Scientific Library*-related code, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must tell them their rights. (これらの権利が万人にあることを明確にするため、誰もこれらの権利を他者から奪うことができないものとする。たとえば *GNU Scientific Library* またはこれに関連するコードを公開する時、それを見る人にはこれらの権利を付与せねばならない。その人たちがそのコードを入手できるようにせねばならない。そしてこれらの権利のことを通知せねばならない。)

Also, for our own protection, we must make certain that everyone finds out that there is no warranty for the programs that relate to *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. (また *GNU Scientific Library* の開発者を守るため、*GNU Scientific Library* と関連するプログラムには何の保証もないこととする。誰かがこれらのプログラムを改変して他の人に渡したとき、それを受け取った人には、受け取ったものが *GNU Scientific Library* の開発者が公開しようとしているものではなく、改変した者によって生じた問題は *GNU Scientific Library* の開発者とは無関係であることが分かるようにしてほしい。)

The precise conditions of the licenses for the programs currently being distributed that relate to *GNU Scientific Library* are found in the General Public Licenses that accompany them. (公開条件は詳しくは GNU General Public License で規定されており、その文面は公開されているソフトウェアに添付されている。)

GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not

have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

   It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

   A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

   B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

   C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

   D. Preserve all the copyright notices of the Document.

   E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

   F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

   G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the

compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (C) *year* *your name*.
> Permission is granted to copy, distribute and/or modify
> this document under the terms of the GNU Free
> Documentation License, Version 1.3 or any later version
> published by the Free Software Foundation; with no
> Invariant Sections, no Front-Cover Texts, and no
> Back-Cover Texts. A copy of the license is included in
> the section entitled ‘‘GNU Free Documentation License’’.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being *list their*
> *titles*, with the Front-Cover Texts being *list*, and
> with the Back-Cover Texts being *list*.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.