Hrothgar

Parallel fitting and Markov Chain Monte Carlo of CPU-intensive functions
Version 2.3, updated 20 July 2014

Andisheh Mahdavi (andisheh.mahdavi@gmail.com)

This manual is for Hrothgar (version 2.3, 20 July 2014), which provides parallel fitting and Markov Chain Monte Carlo of CPU-intensive functions.

Table of Contents

1 Invocation

```
#include <hrothgar.h>

void hrothgar_init_pars(struct hrothgar_setup *setup,
              int ntotparams, char **paramnames,
              double *pmin, double *pmax, double *initvals,
              int *dofit, char **pcomment);


void hrothgar_init_string_pars(struct hrothgar_setup *setup,
              int ntotstringparams, char **stringparamnames,
              char **stringparamvalues, char **pcomment);

int hrothgar_init(struct hrothgar_setup *setup, int argc,
              char **argv);

void hrothgar(unsigned long ndata, double *x,
              double *data, double *error,
              int (*get_model)(double *x,
                               double *params,
                               double *model,
                               double *errors,
                               double *logprior,
                               unsigned long *ndata,
                               void *dataptr),
              void *dataptr,
              struct hrothgar_setup *setup);

void hrothgar_statonly(double (*get_stat)(double *params,
                              int np, void *dataptr),
                              void *dataptr,
                   struct hrothgar_setup *setup)


<linked program> [OPTIONS] [[input-file] output-file]
```

2 Description

Hrothgar is a library that carries out nonlinear fitting of data to functions of arbitrarily many variables. The fitting can be carried out either on a single machine, or on a computer cluster using a Message Passing Interface (MPI) library. Multi-CPU and multicore machines can take advantage of OpenMP. There are two modes: Levenberg-Marquardt (LM) mode, or Markov-Chain-Montecarlo (MCMC) mode.

In LM Mode, which assumes uncorrelated errors, the calling program provides the data, model, errors and the fitting function `f(a,x)`, where `a` are the N free parameters and `x` are the M data points. Starting from an initial vector in the N dimensional minimization space, Hrothgar calculates the MxN Jacobian of `f(a,x)` numerically and proceeds downhill using a Levenberg-Marquardt alogrithm. Hrothgar can do this will conducting random walks uphill. The program concludes by saving the best-fit parameters, the associated errors, and graphical images of the N(N-1)/2 separate pairwise joint probability distributions of the best-fit parameters.

In MCMC mode, instead of `f(a,x)`, the calling program can simply provide -2 ln P (equal to the chi square for normally distributed variables), and Hrothgar will conduct an MCMC chain analysis with importance sampling and a robust exploration of the topology. The accompanying program `mcmcprob` allows for visualization of this topology and calculation of error parameters.

Any program which is usable in LM mode (i.e., contains uncorrelated errors AND is able to provide all data and errors to Hrothgar) can also be run in MCMC mode.

2.1 Installation and SCIUTILS package

In addition to a POSIX environment Hrothgar also requires the GNU Scientific Library (GSL), the compression library zlib, and the CFITSIO library to be installed. Plotting MCMC chains requires the pgplot library. These packages maybe installed through your system's package manager (Linux) or through macports (Mac OS X).

Included with Hrothgar is the SCIUTILS package also by A. Mahdavi. This package will compile along with Hrothgar; see the sciutils subdirectory and sciutils.pdf for details.

On a Linux system the full list of required packages is

```
openmpi-bin
texinfo
zlib1g-dev
libcfitsio3-dev
libfftw3-dev
libgsl0-dev
libopenmpi-dev
pgplot5
libreadline6-dev
pdfjam
```

these can all be installed via apt-get on a system like Ubuntu. For a Macintosh system, the required Macports packages are

```
openmpi
```

```
gsl
zlib
cfitsio
pgplot
ghostscript
pdfjam
```

For installation to /usr/local, installation proceeds via standard

```
./configure; make; sudo make install
```

Those without root access may install to another location via

```
./configure --prefix=/other/place; make; make install
```

2.2 Initialization

Hrothgar should be called from another program that provides the function to be minimized as well as its parameters. There are three steps involved in initialization.

First the program needs to register the default fit parameters via a call to `hrothgar_init_pars`, providing the total number of fit parameters `(ntotparams)`, the names of the parameters `(paramnames)`, the minimum and maximum allowed values of the parameters `(pmin,pmax)`, their default values `(initvals)`, their default fitting states `(dofit)`, and optionally a comment describing the variables `(pcomment)`. The `setup` variable needs to be declared, but can otherwise be safely ignored by the calling program. The development files for the readline utility are highly recommended.

Next, the program call `hrothgar_init` to parse the command line arguments. Upon success, `hrothgar_init` will return the rank of the node assigned to it by MPI (or 0 in a single CPU setting).

Finally, the main program makes one of two calls:

1. In case of diagonal (uncorrelated) errors, the full Hrothgar functionality, including both Levenberg-Marquardt involves a call outright call to `hrothgar` with the number of data points `ndata`, abscissae `x`, the actual `data` and `errors`, and fitting function `get_model`. An auxilliary pointer `dataptr` can be used to convey auxilliary information necessary for the fit; `dataptr` will be passed straight to `get_model` without modification.
2. The user may also choose to only supply only -2 ln P, in which case a call to `hrothgar_statonly` should be made instead. This only requires the fitting function `get_stat` to be supplied `dataptr` will be passed straight to `get_stat` without modification.

Following the call to Hrothgar, everything is handled automatically.

Optionally, non-fittable string parameters may be given to Hrothgar by calling the `hrothgar_init_stringpars()` function prior to calling `hrothgar_init`. The number of of string parameters is `nstringpars`, and the character matrices `stringparname`, `stringparval`, and `stringparcomment` are the names of the string parameter names, default values, and comments, respectively. String parameters may be altered via the command line and may be useful for the calling program's internal initialization.

2.3 Configuration files

When run with the `--defaults` argument, the main program will carry out a fit with the default input parameters discussed above. However, it is useful to store all the parameters in a configuration file. For this purpose, the `-d` command line parameter outputs the default configuration file (generated using the default variables passed to `hrothgar_init_pars` and `hrothgar_init_stringpars`) to standard output.

If `input-file`, is specified, its contents override the default settings, except for `pmin` and `pmax`, which may never be overridden. In addition, the `-p` command line parameter can override the default settings (see *Index* below).

If `output-file` is specified, the results of the minimization run are stored in a similar output configuration file, with the fittable parameters at the best-fit minimum. In addition, any changes in input parameters expressed via the `-p` command line parameter are reflected in `output-file`.

Any characters in a line following and including a pound sign ("#") are treated as comments . Information from the hrothgar run, including the random number seed, the date and time, the best-fit statistic, and 1D confidence intervals for the free parameters, are appended to `output-file` as comments.

The configuration file has two types of entries; normal, fittable parameter entries, and non-fittable, string parameter entries. A normal line has five whitespace-delimited elements. The first element contains the (unique) name of the fittable parameter; the second element contains the starting value of the parameter; the third element is either a zero or or one, indicating whether the parameter is to be fit (0) or frozen (1). The next two numbers are optional and specify the minimum and maximum allowed value for the parameter, e.g.:

```
slope     0.3 0 -5 5
intercept 7.9 1
```

In the above example, there are two parameters, "slope" and "intercept." Slope is a free parameter, and it will be constrained to lie between -5 and 5. Intercept is a frozen parameter, and it will not be varied during the fit. Suppose however we use the command line string `-p intercept~`; in that case, interecept will be thawed (i.e. fit), and the minimum and maximum allowed values will be the values set by `pmin` and `pmax`.

The second type of allowed parameter is a string parameter. These parameters have no influence on the fit, and their values will simply be copied from the input file to the output file, unless `-p` is used on the command line. Example:

```
logfile simpleline.log # File name to use for logging
```

After launch, operation is automatic and requires no user interaction. Three basic modes of operation are available.

2.4 Levenberg-Marquardt Minimization

This is the first possible mode of operation. The merit function is evaluated at the values specified in `input-file`, and the Levenberg-Marquardt steeping routine steps downhill until convergence. Optionally with the `-P` parameter, a powell-style minimization is also attempted (available only with multiple CPUs). Optionally with the `-f` parameter, a number of uphill steps are taken following each convergence. This to help increase the likelihood of convergence to a global minimum.

Hrothgar utilizes bounded Levenberg-Marquardt minimization. This means that each fit parameter is always bounded by a global upper and lower limit, specified by `pmin` and `pmax`. The limits specified in `input-file` may never exceed these limits.

The bounded minimization is achieved through the following transformation:

```
y = arcsin((2 x - xmax - xmin)/(xmax - xmin)).
```

With the above transformation, the Levenberg-Marquardt routine can operate in an unbounded and continuously differentiable space.

Levenberg-Marquardt errors are derived assuming that the minimum is well approximated by a suitably-dimensional paraboloid. This is frequently not the case, in which case MCMC chains are a better choice.

2.5 MCMC chain

Instead of minimization, the `-m <N>` command line parameter prompts Hrothgar to attempt a Markov Chain Monte Carlo procedure, generating one or more chains of total length at least `<N>` times the number of free parameters. For example, if there are 5 free parameters in the model, the total length of all chains will be at least 5`<N>`, if not more. Without MPI, only one chain is run.

With MPI, several chains (equal to the number of MPI nodes engaged) are run and checked for convergence. The convergence test is that the means of all the free parameters for each chain lie within one standard deviation of the means of all the other chains. Again, under the example of 5 free parameters, if the chains converge, Hrothgar will run to at least $5<N>$ total chain length; if they do not converge, Hrothgar will run to at most $50<N>$ total chain length, or until the chains converge.

See the example below for a description of how data in the chains is stored and see the visualization section for how display and use it.

3 Example

3.1 Fitting a Gaussian

I now present a worked example, which may also be used as a test suite for Hrothgar, or even a template using which users may build their own fitting routines. The following program generates a number of Gaussian deviates, bins them into a group size specified by the user, and fits a Gaussian to the result. It is in effect "testing" the accuracy of the GSL Gaussian distribution.

3.2 Annotated Code

The contents of hrothgar_test.c:

```
#include <gsl/gsl_rng.h>
#include <gsl/gsl_randist.h>
#include <gsl/gsl_sort.h>
#include <hrothgar.h>
#include <math.h>

// HAVE_MPI is automatically defined in the build process for Hrothgar.
#ifdef HAVE_MPI
#include <mpi.h>
#endif

// This function simply returns a Gaussian distribution for
// ndata values of x, storing them in model.  par[0] is the mean,
// par[1] is the sigma, and par[2] is the normalization of the
// gaussian.
// The user can use this function as a template for building fits.

int gaussian_model(double *x, double *pars, double *model,
                   double *error, double *logprior,
                   unsigned long ndata, void *extra){

  double diff;
  unsigned long i;

  for (i = 0; i < ndata; ++i) {
    diff = (x[i]-pars[0])/pars[1];
    model[i] = pars[2]*exp(-diff*diff/2.)/(pars[1]*2.5066283);
  }

  return 0;

}
```

```
/* double gaussian_model_stat(double *pars, int npar, void *extra) */
/* { */
/*   double diff; */
/*   diff = pow((1 - pars[0]),2.)+pow((2-pars[1]),2.)+pow((0.5-pars[2]),2.); */
/*   return diff; */
/*  } */

int main(int argc, char *argv[]) {

  unsigned long i,j;
  double tempmean;

  // This is a required declaration.
  struct hrothgar_setup setup;

  // There are 3 total fit parameters:  the mean, sigma, and
  // normalization of the Gaussian
  // Parameter names as they should appear in the default config file
  static char *paramnames[] = { "mean", "sigma", "norm" };

  // Which parameters should be frozen?  None, in our case, we are
  // fitting them all.  This is a default only, and can be overridden.
  static int frozen[] = { 0, 0, 0 };

  // Optional comments describing what each parameter means:
  static char *comments[] = { "Mean", "Sigma",
                              "Normalization" };

  // Use remote initial values for the minimization.
  static double initvalues[] = { -1, 3., 23.  };

  // Minimum and maximum values allowed for minimization
  static double parmin[] = { -10., 0.001, 0.0001 };
  static double parmax[] = { 10., 100., 10000.  };

  // String parameters.  These are parameters not used during
  // the fit.  In this case, we'll use string parameters to
  // generate our synthetic data.

  char *stringparname[] = { "simmean", "simsig", "nsims" };

  // By default, we'll simulate a centered Gaussian with unit sigma
  // Note that the parameters need to be string valued here.
  // By default we are doing 1000 simulations.  On a modern system,
  // you will need 30000000 (3e7) simulations or more before you will
  // see a big slowdown on the fit.
  char *stringparval[] = { "0.", "1.", "1000" };
```

```
char *stringparcomments[] = { "Simulated mean",
                              "Simulated sigma",
                              "Number of Simulations"};


/******************** Program starts here **********************/

// Initialize the string parameters.  Pointers to them are stored
// within the setup structure.
hrothgar_init_stringpars(&setup,3,stringparname,stringparval,
                         stringparcomments);

// Initialize the fit parameters
hrothgar_init_pars(&setup,3,paramnames,parmin,parmax,
                   initvalues,frozen,comments);

// Initialize the hrothgar core.
// Note that any command line modifications to the string
// parameters will be written to stringparval at this point
if (hrothgar_init(&setup,argc,argv) < 0) { return; }

// Get the mean and sigma of the Gaussian to simulate, as well
// as the total number of simulations to perform.
double simmean = atof(stringparval[0]);
double simsig = atof(stringparval[1]);
unsigned long nsim = atol(stringparval[2]);

// Use 30 data points per bin
if (nsim < 150) BYE("Too few simulations.");
unsigned long nbins = nsim/30;
printf("Init %d\n",setup.node);


// Allocate room for the binned data.
double *x = (double *)malloc(nbins*sizeof(double));
double *y = (double *)malloc(nbins*sizeof(double));
double *ye = (double *)malloc(nbins*sizeof(double));

// Now it is time to generate the data.  For simiplicity, we'll have
// only the master MPI node generate the data.  If we're not running
// MPI, the node number will be 0 by default anyway.

if (setup.node == 0) {

  // At this point, hrothgar_init has already conveniently
  // initialized and seeded a random number generator for us.
  gsl_rng *rng = setup.generator;
```

```
    double *data = (double *)malloc(nsim*sizeof(double));

    // Generate the Gaussians
    for (i = 0; i < nsim; ++i)
      data[i] = simmean+gsl_ran_gaussian(rng,simsig);

    // Sort them
    gsl_sort(data,1,nsim);

    // Bin them into a histogram of width 30
    j = 0;
    tempmean = 0.;

    for (i = 0; i < nsim; ++i) {
      tempmean += data[i];

      if (i % 30 == 29) {
        x[j] = tempmean/30.;
        tempmean = 0.;
        y[j] = 30./(nsim*(data[i]-data[i-29]));
        ye[j] = 5.5/(nsim*(data[i]-data[i-29]));
        ++j;
      }
    }
    free(data);
  }


    // Conditional MPI section
#ifdef HAVE_MPI
    // Send all the slaves the binned data.
  printf("Trying %d\n",setup.node);
    MPI_Bcast(x,nbins,MPI_DOUBLE,0,MPI_COMM_WORLD);
    MPI_Bcast(y,nbins,MPI_DOUBLE,0,MPI_COMM_WORLD);
    MPI_Bcast(ye,nbins,MPI_DOUBLE,0,MPI_COMM_WORLD);
#endif
    printf("OK %d\n",setup.node);
  // Run the fit
  hrothgar(nbins,x,y,ye,gaussian_model,NULL,&setup);
  //hrothgar_statonly(gaussian_model_stat,NULL,&setup);

}
```

3.3 Running the Test Suite

To run the test suite following installation, simply run

```
hrothgar_test -D
```

This should measure the mean of the Gaussian to be near 0; the sigma and the normalization should be near 1.

To examine the format of the default configuration file and save the results of the fit:

```
hrothgar_test -d > input.cfg
hrothgar_test -Xtf50 input.cfg output.cfg
```

In the above example, Hrothgar will perform an uphill step and reminimize the merit function, repeating the procedure 50 times and outputting timing statistics. The Gaussian-approximated errors are stored in the output configuration file.

3.4 Parallel Runs (MPI and OpenMP)

Hrothgar is capable of running in either or both of the MPI and OpenMP parallel environments.

Hrothgar does not require any special treatment when running in an MPI context. It will automatically realize that it is in a parallel environment and take advantage of its capability. If users wish to limit the number of CPUs that hrothgar uses, they can specify `-n` on the command line.

In minimization mode, Hrothgar uses MPI primarily for calculating the Jacobian of the merit function. For a fit with m parameters, this requires $2m$ calculations of the merit function. Thus, during minimization Hrothgar can take full advantage of at most $2m$+1 CPUs, hence its attractiveness for high-dimensional problems.

In MCMC mode, each MPI slave node runs a different MCMC chain, which are checked for convergence by the master mode.

To test the performance of your MPI setup relative to the single CPU setup:

```
mpirun -np 4 hrothgar_test -Xtm300000 input.cfg output.cfg
```

The output of this program is stored in binary files called output.cfg.??.mcmc, which can be viewed using the included mcmcprob utility. See the visualizations section below.

Users with capable compilers may use OpenMP within their merit functions; Hrothgar does not use OpenMP capabilities in a way that interferes with this. Thus OpenMP can be used together with MPI for even larger speed gains.

3.5 Visualizing MCMC chains

The included mcmcmprob utility can be used to extract statistics from the mcmc chains, as well as to plot them if the optional pgplot package is available. The syntax is

```
mcmcprob [FILE ... [+ FILE... ] [-i | -d | -p filename | -a ] BN cmds...
```

Here FILE indicates a *mcmc output by Hrothgar; all files mentioned together are plotted using the same symbols, but encountering a "+" in the file list causes the symbol to be switched, so that different data sets can be contrasted.

Burnin is the number of chain events to ignore at the begining of the chain prior to starting the statistic.

One of the four available commands can be selected. -i enters an interactive mode, which requires pgplot. In interactive mode, a simple command line interface is available. To plot

variable against variable, type "c1 c2" which will generate a plot of the chain points in the c1-c2 space. Typing "c1 c2 c3" will generate three separate plots of the chain points in c1-c2, c1-c3, and c2-c3 space; and so on. Typing "l" will list all fields available. Two special fields are also available: "ord," which retains memory of the order in which each chain point was selected, and "chisq," which retains - 2 ln P. These can also be plotted against any of the other fields. Preceding a series of fields with "c", as in "c col1 col2" will generate contours of 68% and 95% confidence instead of plotting the MCMC points. "q" quites. In the above example "cmds" can be any command that is valid in interactive mode.

The command -p tells pgplot to create a postscript file called filename, rather than printing the output to the screen.

The command -d simply prints out the marginalized 1D error bars in the listed quantities of c1 c2. Each of c1 and c2 is treated as a regular expression and any matching columns will be printed out. The command -a outputs more detailed statistics. The -d and -a commands are the only ones available if pgplot is not selected.

For example, following the above MCMC run, the following command

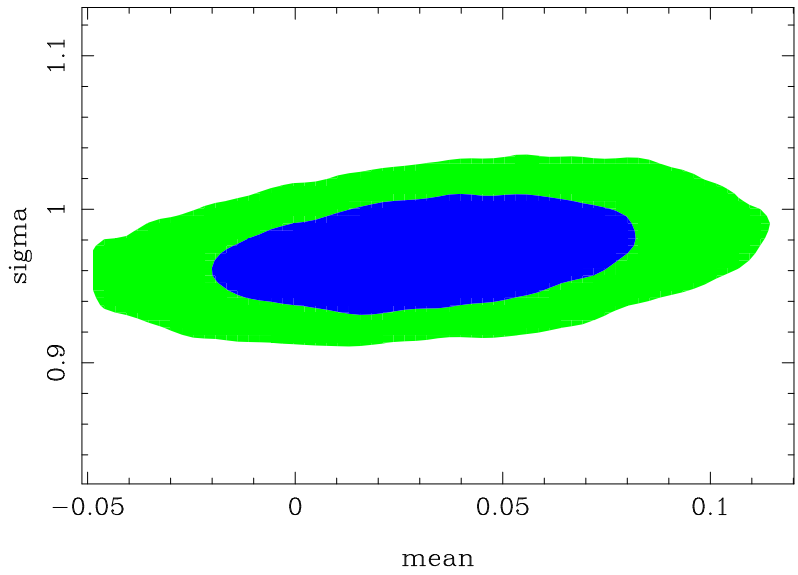
```
mcmcprob output.cfg.*mcmc -d 0 mean sigma
```

will yield an output similar to

```
mean                        -1.392447E-03         1.927497E-03
sigma                        9.982855E-01         1.351426E-03
```

```
mcmcprob output.cfg.*mcmc -p hrothgar-example.ps 0 c mean sigma
```

Will yield something like attached image (with 'c' meaning "contour").



4 Index of command line parameters

The default values of the command line parameters may be seen by specifying `--help [parameter]` on the command line.

`-C, --conf` *level*

Specify confidence level at which 1D errors are reported.

`-D, --defaults`

Run without any configuration files, using internal default parameters. If the configuration files are specified, -D does nothing.

`-E, --evalonly`

Evaluate the merit function at the initial vector and quit—do not carry out any fitting or error analysis.

`-H, --hardlimits`

Ignore the limits set in the input parameter file; use hard limits on the parameter values as specified by the calling program.

`-L, --license`

Display the terms for copying, modifying and redistributing this program.

`-R, --remember`

Normally, if the user modifies the fit/frozen state of parameters via the `--parameter` command line option, the new state is recorded in the output file. With `--remember`, the old state is recorded instead. This is useful in conjunction with the `%` modifier to `--parameter`.

`-S, --seed` *number*

The random number seed is normally read from /dev/random. This uses the user specified value instead. Random numbers are used in the Tree MCMC mode, as well as in the uphill steps taken using the `--floataround` option. Implies `--predictable` in Tree MCMC mode.

`-X, --overwrite`

Overwrite all existing files, rather than exiting with an error condition.

`-a, --inputaccuracy` *eps*

Specify the fractional accuracy of the fitting function provided by the calling program. This is used in estimating the error on the numerical derivatives.

`-c, --covaronly`

Do not conduct minimization. If the covariance matrix exists on the disk already, read it in and use it to output confidence contours. If not, or if `--ignorecovar` is specified, calculate the covariance matrix using the initial conditions.

`-d, --dumpconfig`

Write the default configuration file to standard output.

`-e, --eps` *eps*

Specify the fractional tolerance for convergence.

`-g, --gaussstep` *[sigma]*

For MCMC mode only. Initial sigma of Gaussian by which to step the parameters fractionally.

`-h, --help` *[parameters]*

Summarize the available options. If *parameters* are specified, show their default value.

`-m, --mcmc` *number*

Do not minimize. Instead, generate a Tree Markov Chain Monte Carlo Chain of length *number*.

`-p, --parameter` *name*[%=~@][*value*]

This powerful option allows manipulation of the fit or frozen parameters, and may be specified as many times as needed. The parameter *name* is selected. The operator '`@`' freezes it at *value*, or at its initial value if *value* is not specified. The operator '`~`' does the opposite: it causes *name* to be fit, starting with *value*, or with its initial value if *value* is not specified. The operators '`=`' respects the initial fit/frozen state of *name*, but sets its initial value to *value*. Finally, the '`%`' operator causes *name* to be fit, but freezes all other parameters not previously modified by '`%`'.

Examples:

`-p slope=2` sets *slope* to 2, but does not change its fit/frozen status

`-p slope~` causes *slope* to be fit starting at its initial value, even if it was frozen by default

`-p slope@3` freezes *slope* at the value 3.

`-p slope%2 -p intercept%` freezes all parameters except *slope* and *intercept*; *slope* is fit with an initial value of 2, while *intercept* is fit starting at its default initial value.

`-q, --quiet`

Try to output as little text as possible.

`-t, --timer`

Measure Hrothgar's performance (the time it takes for each Levenberg-Marquardt step). Works with the clustered as well as the single CPU modes.

`-v, --verbose`

Show lots of diagnostic output.

5 Acknowledgments

Hrothgar was made possible through the generous support of Arif Babul, Henk Hoekstra, and John Criswick. Partial support provided by NASA through Chandra award No. AR0-11016A, issued by the Chandra X-ray Observatory Center, which is operated by the Smithsonian Astrophysical Observatory for and on behalf of NASA under contract NAS8-03060. AM was also supported through NASA ADAP grant 11-ADAP11-0270.

6 GNU Free Documentation License

Version 1.2, November 2002

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 non-commercially. 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.

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 for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

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.